2.3. Image Formats

August 28, 2024

Jarrian Vince G. Gojar
Instructor I
College of Information and Communications Technology, Sorsogon State University, Philippines

1 Image Formats

Storing an image digitally can be a rather simple concept in which each pixel is stored in three bytes (one each for color channel). However, there are many different ways to store images digitally, and the format in which an image is stored can have a significant impact on the image's quality and file size. For example, the `JPEG` format is a lossy format, meaning that some image quality is lost when the image is compressed and saved, while the `PNG` format is a lossless format, meaning that no image quality is lost when the image is saved.

A common smartphone can capture an image with a resolution of `4160x2340` pixels, which is over 9 million pixels. Furthermore, this image is a color image, meaning that each pixel is represented by three bytes (one each for the red, green, and blue color channels). This means that a single image can require over 29 million bytes (or 29 megabytes) of storage space. If images were stored in this fashion, they would take up a significant amount of storage space, and it would be difficult to share them over the internet. Therefore, images are typically compressed before they are stored or shared. Greater compression can lead to a smaller file size, but it can also lead to a loss of image quality.

Common Types of Image Formats:

1. Bitmap (BMP)
2. JPEG (Joint Photographic Experts Group)
3. GIF (Graphics Interchange Format)
4. TIFF (Tagged Image File Format)
5. PNG (Portable Network Graphics)

Read More:

- Image File Formats by Adobe
- Image File Formats by Wikipedia

2 Setup

[ ]:
```
%pip install opencv-python opencv-contrib-python pillow numpy matplotlib
```

3 Initial Setup

[ ]:
```
# Import required packages:
import os
import cv2
import PIL
import numpy as np
import matplotlib.pyplot as plt

# Read an image
image_path = '../../assets/images/bg_jellyfish.png'
original_image = cv2.imread(image_path)
original_image = cv2.cvtColor(original_image, cv2.COLOR_BGR2RGB)

# Display the image
plt.figure("Original Image")
plt.imshow(original_image)
plt.axis('off')
plt.title(f'(a) Original Image', y=-0.1)
plt.show()
```

(a) Original Image

4 Bitmaps

A bitmap is a way to represent a grid of pixels.For grayscale images, each pixel is represented by one byte, where 0 is black and 255 is white. For color images, each pixel is represented by three bytes, where each byte represents the intensity of red, green, or blue.

Read More:

- Bitmap

5 JPEG (Joint Photographic Experts Group)

`JPEG` is a lossy compression algorithm that is widely used for photographic images. The `JPEG` standard defines how an image is compressed into a stream of bytes and how the image is decompressed back into an image. The `JPEG` standard is widely used for storing and transmitting images on the internet.

This format sacrifices some image quality to achieve high compression ratios. Since most photographs do not contain exceedingly sharp edges, the loss of quality is usually not noticeable.

The `JPEG` compression converts the image into the `YCbCr` color space and then applies the Discrete Cosine Transform (`DCT`) to the image. The `DCT` is a mathematical technique that converts an image into a set of frequencies. The `DCT` is similar to the Fourier Transform, but it is more efficient for images.

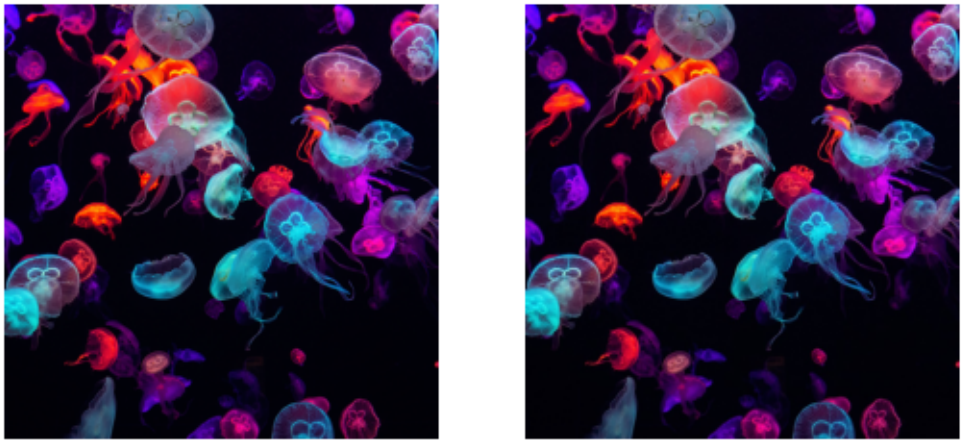
Read More:

- `JPEG` - Wikipedia
- `JPEG` Standard

[ ]:
```
# Convert the image to JPEG format
jpeg_image_path = '../../assets/images/bg_jellyfish.jpg'

# Save the image in JPEG format and in RGB color space
cv2.imwrite(jpeg_image_path, cv2.cvtColor(original_image, cv2.COLOR_RGB2BGR),␣
 ↪[cv2.IMWRITE_JPEG_QUALITY, 90])

# Read the JPEG image
jpeg_image = cv2.imread(jpeg_image_path)
jpeg_image = cv2.cvtColor(jpeg_image, cv2.COLOR_BGR2RGB)

# Compare the original(PNG) and JPEG images
print('PNG vs JPEG Image')
plt.figure("PNG vs JPEG Image")

plt.subplot(1, 2, 1)
plt.imshow(original_image)
plt.title(f'(a) Original Image (PNG)', y=-0.15)
plt.axis('off')
```

```
plt.subplot(1, 2, 2)
plt.imshow(jpeg_image)
plt.title(f'(b) JPEG Image', y=-0.15)
plt.axis('off')

plt.show()

print("PNG Details")
print("Resolution:", f"{original_image.shape[1]}x{original_image.shape[0]}")
print("Size (KB):", f"{os.path.getsize(image_path) / 1024:.2f} KB")
print("Size (MB):", f"{os.path.getsize(image_path) / (1024 * 1024):.2f} MB")

print("\nJPEG Details")
print("Resolution:", f"{jpeg_image.shape[1]}x{jpeg_image.shape[0]}")
print("Size (KB):", f"{os.path.getsize(jpeg_image_path) / 1024:.2f} KB")
print("Size (MB):", f"{os.path.getsize(jpeg_image_path) / (1024 * 1024):.2f}␣
 ↪MB")
```

PNG vs JPEG Image



(a) Original Image (PNG)

(b) JPEG Image

```
PNG Details
Resolution: 872x872
Size (KB): 1377.51 KB
Size (MB): 1.35 MB

JPEG Details
Resolution: 872x872
Size (KB): 166.31 KB
Size (MB): 0.16 MB
```

The code above shows the comparison between the original PNG image and the JPEG image. The

JPEG image is compressed using the JPEG algorithm, which results in a smaller file size. The JPEG image is visually similar to the original PNG image, but it has a smaller file size.

6 GIF (Graphics Interchange Format)

GIF is a bitmap image format that was developed by a team at the online services provider CompuServe led by American computer scientist Steve Wilhite on June 15, 1987. It has since come into widespread usage on the World Wide Web due to its wide support and portability.

GIF uses a palette to store color information, and it supports animations. A look-up table of 256 colors (RGB) is used to represent the image. GIF images are compressed using the Lempel-Ziv-Welch (LZW) lossless data compression technique to reduce the file size without losing the image quality.

GIF has far too few colors to be a good format for photographs, therefore a GIF compression will estimate the color palette of the image and reduce the number of colors in the image to 256 or less. Although GIF is not suitable for photographs, it is an excellent format for images with large areas of solid color, like logos, icons, and simple drawings.

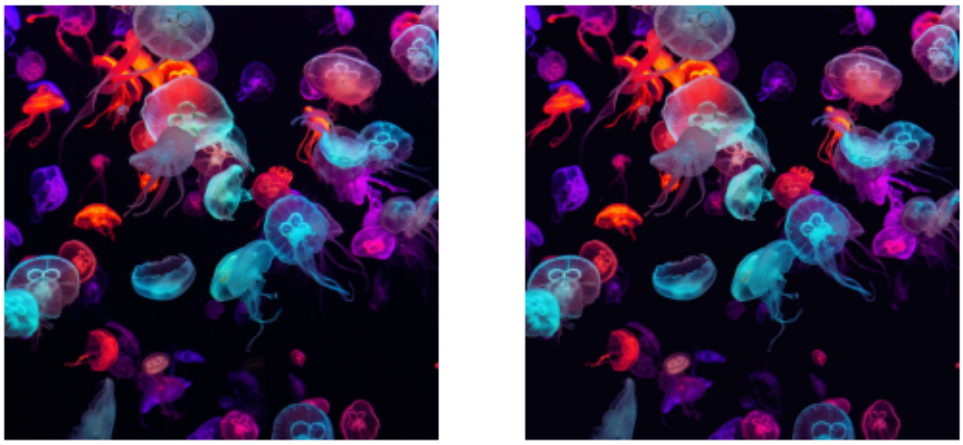
Read More:

- GIF - Wikipedia

[ ]:
```
# Read an image
image_PIL = PIL.Image.open(image_path)

# Convert the image to GIF format
gif_image_path = '../../assets/images/bg_jellyfish.gif'

# Save the image in GIF format
image_PIL.save(gif_image_path, format='GIF')

# Read the GIF image
gif_image = PIL.Image.open(gif_image_path)

# Compare the original(PNG) and GIF images
print('PNG vs GIF Image')

plt.figure("PNG vs GIF Image")

plt.subplot(1, 2, 1)
plt.imshow(original_image)
plt.title(f'(a) Original Image (PNG)', y=-0.15)
plt.axis('off')

plt.subplot(1, 2, 2)
plt.imshow(gif_image)
plt.title(f'(b) GIF Image', y=-0.15)
plt.axis('off')
```

```
plt.show()

print("PNG Details")
print("Resolution:", f"{original_image.shape[1]}x{original_image.shape[0]}")
print("Size (KB):", f"{os.path.getsize(image_path) / 1024:.2f} KB")
print("Size (MB):", f"{os.path.getsize(image_path) / (1024 * 1024):.2f} MB")

print("\nGIF Details")
print("Resolution:", f"{gif_image.size[0]}x{gif_image.size[1]}")
print("Size (KB):", f"{os.path.getsize(gif_image_path) / 1024:.2f} KB")
print("Size (MB):", f"{os.path.getsize(gif_image_path) / (1024 * 1024):.2f} MB")
```

PNG vs GIF Image



(a) Original Image (PNG)

(b) GIF Image

```
PNG Details
Resolution: 872x872
Size (KB): 1377.51 KB
Size (MB): 1.35 MB

GIF Details
Resolution: 872x872
Size (KB): 224.42 KB
Size (MB): 0.22 MB
```

The above code demonstrates how to convert an image to GIF format. The image is read using the PIL library, and the image is saved in GIF format using the `save()` method. The GIF image is then read and displayed using the `imshow()` function of the matplotlib library.

As shown in the output, the GIF image has a smaller file size compared to the original PNG image. However, some details are lost in the GIF image due to the reduction in the number of colors.

7 TIFF (Tagged Image File Format)

The `TIFF` format is a flexible format that normally saves 8 or 16 bits per color (red, green, blue) for a total of 24 or 48 bits. `TIFF`s also support layers, alpha channels, and annotations. The format is widely supported by image-manipulation applications, by publishing and page layout applications, and by scanning, faxing, word processing, optical character recognition, and other applications.

`TIFF` format can store data with or without compression. If the data is not compressed, then `TIFF` files can be larger than bitmap files. Scanner software often saves scanned images as `TIFF` files which allows the user to view the raw image data in the file. This is actually useful to companies which will modify the image by inserting new data fields into the image file. Standard `TIFF` files will not be able to display the new data fields and may not be able to display the image at all.

Read More:

- `TIFF` (Tagged Image File Format)

8 PNG (Portable Network Graphics)

The `PNG` format was designed for transmitting images over the internet, thus gaining its name "Portable Network Graphics". It is a lossless compression which means that it creates files that are larger than `JPEGs`, but it does not lose any information in the process. `PNGs` are ideal for images that require transparency and for images that need to be edited and saved multiple times.

Unlike `JPEGs`, `PNGs` support transparency. This means that you can have an image with a transparent background that allows you to place the image on top of another image or background. This is useful for logos and other images that require a transparent background.

PNG supports RGBA color space, which means that it can store an additional channel for transparency. This channel is called the Alpha channel and it stores information about how opaque each pixel is. This allows for images to have a transparent background.

Read More:

- Wikipedia - Portable Network Graphics

[ ]:
```
# Read an image with transparency
transparent_image_path = '../../assets/images/nb_rainbow_smoke.png'

# Read the image
transparent_image = PIL.Image.open(transparent_image_path)

# Divide the image into RGB and Alpha channels
r, g, b, a = transparent_image.split()

# Display the image
plt.figure("RGBA Channels")

plt.subplot(2, 3, 1)
plt.imshow(transparent_image)
plt.title(f'(a) Original Image', y=-0.25)
```

```
plt.axis('off')

plt.subplot(2, 3, 2)
plt.imshow(r, cmap='Reds')
plt.title(f'(b) Red Channel', y=-0.25)
plt.axis('off')

plt.subplot(2, 3, 3)
plt.imshow(g, cmap='Greens')
plt.title(f'(c) Green Channel', y=-0.25)
plt.axis('off')

plt.subplot(2, 3, 5)
plt.imshow(b, cmap='Blues')
plt.title(f'(d) Blue Channel', y=-0.25)
plt.axis('off')

plt.subplot(2, 3, 6)
plt.imshow(a, cmap='Greys')
plt.title(f'(e) Alpha Channel', y=-0.25)
plt.axis('off')

plt.show()
```

The code above shows the `RGBA` channels of an image with transparency - `PNG` format. The image is divided into four channels: Red, Green, Blue, and Alpha. The Alpha channel stores information about how opaque each pixel is. This allows for images to have a transparent background.

9 Summary

There are many different ways to store images digitally, and the format in which an image is stored can have a significant impact on the image's quality and file size. Common image formats include `Bitmap (BMP)`, `JPEG`, `GIF`, `TIFF`, and `PNG`. Each format has its own characteristics and use cases. For example, `JPEG` is a lossy format that is widely used for photographic images, while `PNG` is a lossless format that is ideal for images that require transparency. Understanding the differences between image formats can help you choose the right format for your specific needs. It is also important to note that the choice of image format can affect the file size of the image. Lossy formats like `JPEG` can achieve high compression ratios, resulting in smaller file sizes, but at the cost of some image quality. Lossless formats like `PNG` preserve image quality but result in larger file sizes. When choosing an image format, it is important to consider the trade-offs between image quality and file size to ensure that the image meets your specific requirements.

10 References

- Thomas G. (2022). Graphic Designing: A Step-by-Step Guide (Advanced). Larsen & Keller. ISBN: 978-1-64172-536-1
- Singh M. (2022). Computer Graphics and Multimedia. Random Publications LLP. ISBN: 978-93-93884-95-4
- Singh M. (2022). Computer Graphics Science. Random Publications LLP. ISBN: 978-93-93884-03-9
- Singh M. (2022). Computer Graphics Software. Random Publications LLP. ISBN: 9789393884114
- Tyagi, V. (2021). Understanding Digital Image Processing. CRC Press.
- Ikeuchi, K. (Ed.). (2021). Computer Vision: A Reference Guide (2nd ed.). Springer.
- Bhuyan, M. K. (2020). Computer Vision and Image Processing. CRC Press.
- Howse, J., & Minichino, J. (2020). Learning OpenCV 4 Computer Vision with Python 3: Get to grips with tools, techniques, and algorithms for computer vision and machine learning. Packt Publishing Ltd.
- Kinser, J. M. (2019). Image Operators: Image Processing in Python. CRC Press.