<< Back to MAIN notebook

3. Image formats

There can be some differences between the image data loaded to your code and the same data once it is saved to your storage drive. Image data loaded to your code is kept in RAM memory as an N-dimensional array of pixel values (usually 3-dimensional array of 8bit unsigned integers with pixel values between 0 and 255, or 64bit floats with values between 0.0 to 1.0).

However, image formats used for storing the image data on your storage drive (HDD/SSD) rarely save raw pixel values as that could require unreasonable amounts of storage space. Instead, a compression (i.e., size-reduction) algorithm is applied to the raw data to reduce the storage space requirements. Compression algorithms are generally divided into two groups:

1. **Lossless compression** - no information from the image is lost when it is stored to drive. Once stored, image can be retrieved from drive to reconstruct the identical initial N-dimensional array, i.e., original image.
2. **Lossy compression** - a certain amount of information is lost when image is stored in order to reduce its size in storage. This compression type is commonly used with photographic data such as UAV imagery.

In the following chapters, a number of lossless and lossy formats will be presented along with their pros and cons, to allow the user to consider the balance between image quality and computational complexity of applying image velocimetry workflow, including image enhancement algorithms.

> File size comparisons in this notebook are made by using the 4K resolution images found in the `./4K` folder. Other notebooks use reduced size images (1080p) for efficiency reasons.

Contents

In [1]:
```
# Necessary libraries
import matplotlib.pyplot as plt
import cv2

# Use [%matplotlib widget] inside JupyterLab,
# and [%matplotlib notebook] for Jupyter Notebook
```

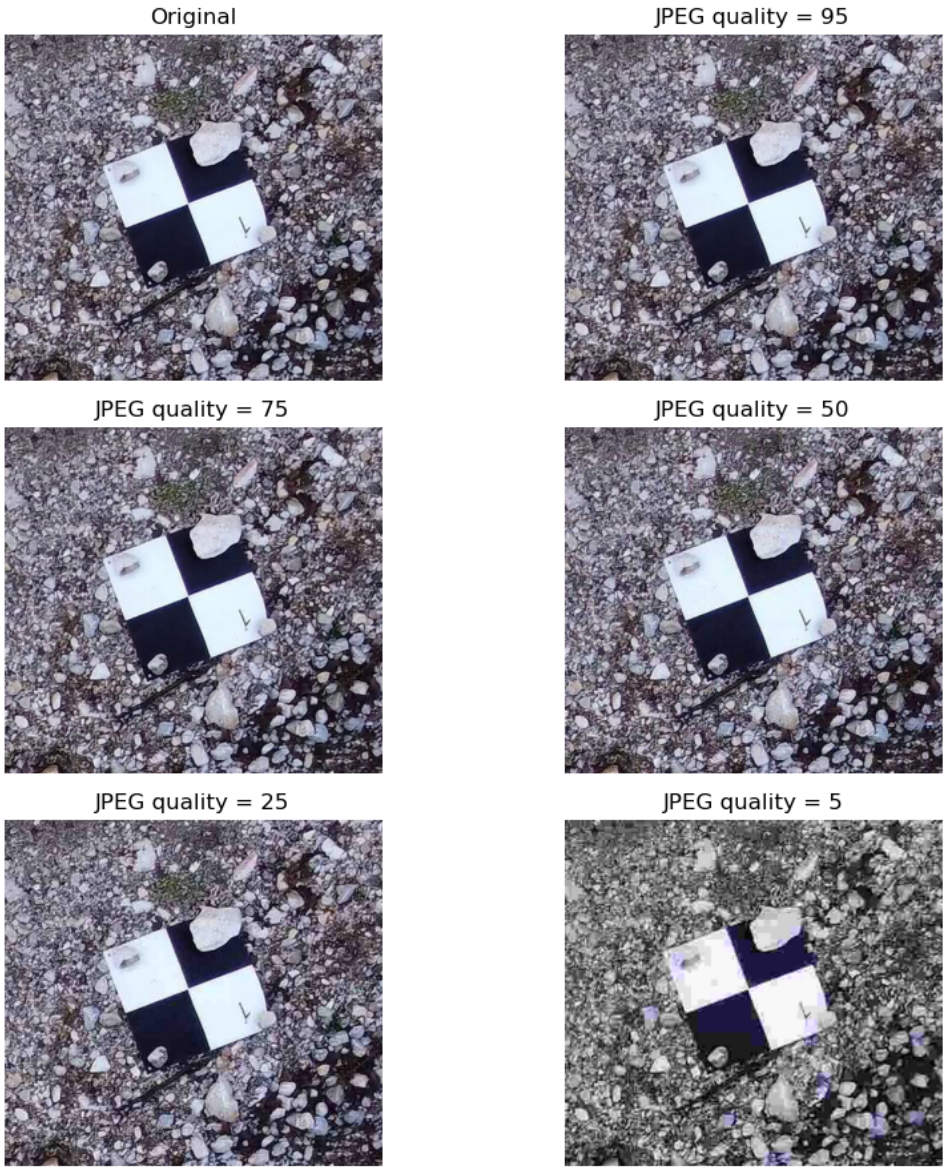
3.1 JP(E)G

JPEG or JPG abbreviates "Joint Photographic Experts Group", and is the most widely used image format. The compression algorithm is **lossy** but can achieve high compression ratio (easily around 10:1). However, there are some variants of JPEG standard which achieve lossless compression, but are rarely used relative to other modern lossless formats.

Consider the following example of a ground control point #1 (top-left) from Image 5. We can vary the JPEG image quality and compare the resulting images (default JPEG quality in OpenCV library is 95):

In [2]:
```
# Read image and crop to center of marker
img_jpeg = cv2.imread('./figures/Fig02.jpg')

# JPEG quality values to plot
image_qual = [95, 75, 50, 25, 5]

# Initiate a 3x2 plot
fig, ax = plt.subplots(nrows=3, ncols=2, figsize=(9.8, 9.0))

# Original image
ax[0][0].imshow(cv2.cvtColor(img_jpeg, cv2.COLOR_BGR2RGB))
ax[0][0].set_title('Original')
ax[0][0].axis('off')

for i, qual in enumerate(image_qual):
    # Encode using JPEG in memory, then decode to numpy array
    _, img_jpeg_enc = cv2.imencode('.jpg', img_jpeg, [int(cv2.IMWRITE_JPEG_QUALITY), qual])
    img_jpeg_dec = cv2.imdecode(img_jpeg_enc, 1)

    # Plot in 2-column figure
    ax[(i+1)//2][(i+1)%2].imshow(cv2.cvtColor(img_jpeg_dec, cv2.COLOR_BGR2RGB))
    ax[(i+1)//2][(i+1)%2].set_title('JPEG quality = {}'.format(qual))

# Hide all figure axes
[a.axis('off') for a in ax.reshape(-1)]

# Show the image
plt.tight_layout()
plt.show()
```

Figure



Images with quality values between 95 and 25 appear to be quite the same until we zoom in on the marker center (using the "zoom to window" option in the interactive figure above). At that zoom level, the differences in colors and edge aliasing degrade noticeably with decreasing JPEG quality. More specifically, there is a noticeable visual noise with lower JPEG quality. If we write the compressed images to disk, we can compare their file sizes:

JPEG quality [-]	File size [KB]	Relative size [%]
100	159.0	100
95	113.0	71.1
90	86.9	54.7
75	60.9	38.3
50	48.3	30.4

JPEG quality [-]	File size [KB]	Relative size [%]
25	**31.5**	**19.8**
5	**11.5**	**7.2**

JPEG format was developed specifically for handling photographic images, such as the ones that are used in UAV velocimetry. For that reason, it is the most commonly used format for such purposes, but the user should be aware that repeated compression of image data has a cumulative effect - the data loss will keep increasing if we:

1. Compress the images when we extract them from the video,
2. Compress the images again when we apply geometric correction techniques,
3. Compress the images again when we stabilize the image sequence,
4. Compress the images again when we orthorectify images,
5. ...

The aforementioned workflow should be avoided **as much as possible** to preserve as much original visual information and avoid adding noise to our images. Some preprocessing tools for UAV velocimetry, such as the **SSIMS tool** algorithmically bundle certain steps together (steps 2-4) to reduce the total number of image compression iterations.

If JPEG format is used, it is advised to keep the JPEG quality at most at 95 (default in OpenCV library) since higher values produce significantly larger file sizes with no noticeable quality improvements. In fact, some lossless image formats may even produce smaller file sizes than JPEG with quality of 100. Lower boundary of acceptable JPEG quality is often cited to be between 75 and 80.

> OpenCV uses a well established `libjpeg` library for JPEG compression. However, some libraries such as the `MozJPEG` have been proven to be more efficient for JPEG compression, but are not implemented in the OpenCV and are not used as often.

3.2 PNG

Second most popular image format is PNG, which stands for "Portable Network Graphics". PNG is a lossless format, meaning no information is lost when saving an image to storage drive. Additionally, PNG supports transparency using an additional Alpha channel (4th channel apart from the usual RGB/BGR).

Unlike with JPEG format, when using OpenCV library to save a PNG image, user can only select the **compression level**. While both higher and lower compression level retain all image information, higher compression level will result in a smaller file size but longer time needed to decode the image from storage drive to memory (and vice versa).

Keep in mind that PNG images have larger file sizes than JPEG with little to no noticeable effect of in image quality, so this format is rarely used for UAV velocimetry purposes. To provide a perspective, the PNG file size of the original image from the previous subsection is **557KB with compression level 0**

(350% increase relative to JPEG with quality of 100) and **342KB with compression level 9** (215% increase relative to JPEG with quality of 100).

> Interestingly, the lowest file size is not always achieved with the highest PNG compression level - the same image saved with **level 1 has a file size of 335KB**.

3.3 WebP

The youngest of the image formats described in this report is WebP developed by Google for web images and graphics. Unlike JPEG or PNG, it can be both lossless and lossy depending on the selected compression strength. As claimed by Google, "*[lossless WebP images are] 26% smaller in size compared to PNGs*" and "*[lossy WebP images are] 25-34% smaller than comparable JPEG images at equivalent SSIM quality index*". Their use for computer vision tasks such as velocimetry is rare, possibly due to the late introduction of WebP support on PCs. OpenCV library supports WebP since 2013, but the initial support was rather buggy until versions 4.x.

However, comparing lossless WebP with PNG and lossy WebP with JPEG reveals a significant advantage of this new format for general purpose computing. Lossless WebP version of the original image from subsection 3.1 with image quality set to 100 (default in OpenCV) yields a file size of 123KB which is only 36.7% of the PNG image with compression level 1. Furthermore, this is a smaller file size than with JPEG image with quality set to 100 (by 22.6%), and is only 8.8% larger than JPEG with quality of 95 (default in OpenCV).

We can also compare lossy WebP compression with the popular JPEG format with the code below. It is noticeable that for similar file sizes (WebP file sizes are actually somewhat lower than JPEG for the same quality parameter, up until the value of around 35), lossy WebP produces significantly lower amounts of compression artifacts even for quality of 5 which produces a file size of just 19KB.

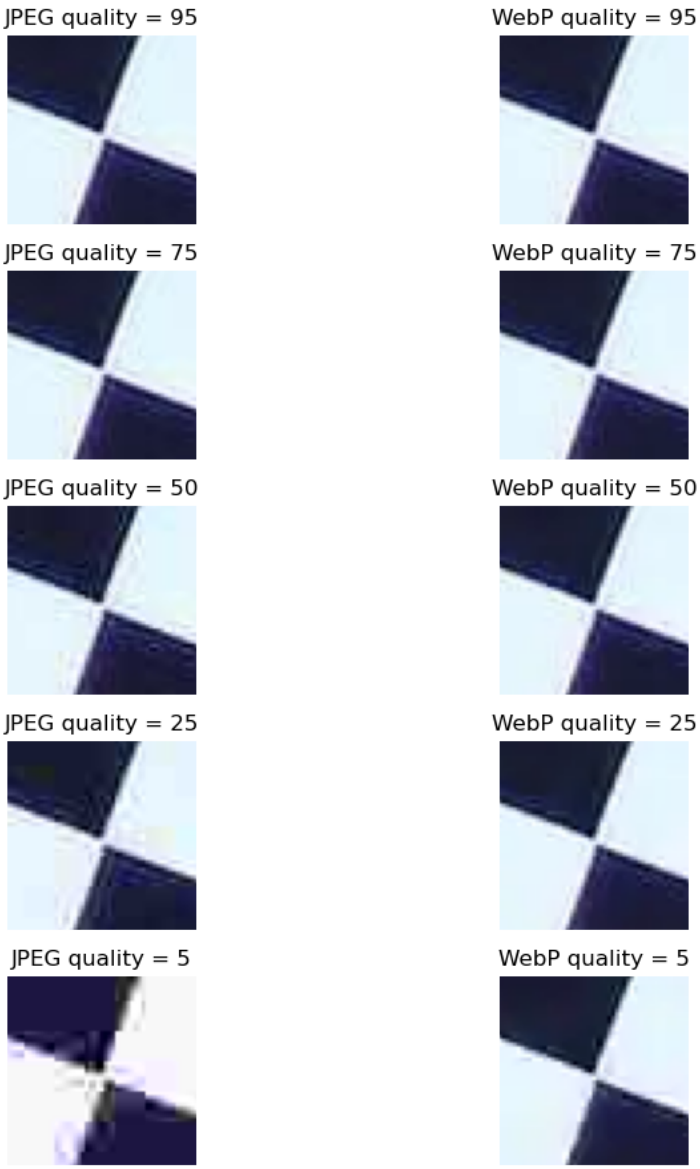
All the above makes the WebP format a strong contender for replacing JPEG as the go-to format for image velocimetry, assuming that necessary libraries for handling such files are installed.

> Keep in mind that not all image viewing software supports WebP format.

In [3]:
```
# WebP image quality to plot
webp_qual = [95, 75, 50, 25, 5]

fig, ax = plt.subplots(nrows=len(webp_qual), ncols=2, figsize=(9.8, 9))

for i, qual in enumerate(webp_qual):
    # Encode using JPEG in memory, then decode to numpy array
    _, img_jpeg_enc = cv2.imencode('.jpg', img_jpeg, [int(cv2.IMWRITE_JPEG_QUALITY), qual])
    img_jpeg_dec = cv2.imdecode(img_jpeg_enc, 1)

    # Encode using WebP in memory, then decode to numpy array
    _, img_webp_enc = cv2.imencode('.webp', img_jpeg, [int(cv2.IMWRITE_WEBP_QUALITY), qual])
    img_webp_dec = cv2.imdecode(img_webp_enc, 1)

    ax[i][0].imshow(cv2.cvtColor(img_jpeg_dec, cv2.COLOR_BGR2RGB))
    ax[i][0].set_title('JPEG, quality = {}'.format(qual))
    ax[i][0].axis('off')
```

```
    ax[i][1].imshow(cv2.cvtColor(img_webp_dec, cv2.COLOR_BGR2RGB))
    ax[i][1].set_title('WebP quality = {}'.format(qual))
    ax[i][1].axis('off')

# Zoom to GCP marker center for a better view
for a in ax.reshape(-1):
    a.set_xlim([195, 245])
    a.set_ylim([180, 230])

plt.tight_layout()
plt.show()
```

Figure



3.4 TIF(F)

TIFF is an abbreviation of "Tagged Image File Format", and is less commonly used for image velocimetry. While TIFF format can be either lossless or lossy, in practice it is almost exclusively used for lossless image storage without compression, which results in very large file sizes - there is an option of LZW (Lempel–Ziv–Welch) compression for TIFF but it is not supported on all devices. For example,

uncompressed TIFF file size of Image 3 is 23.8MB but LZW-compressed TIFF is only 11.6MB (48.7% reduction). However, the same image as compressed JPEG (quality 95) is just 2.2MB (1072% and 523% reduction, respectively). Lossless PNG file size, on the other hand is close to LZW TIFF with 10.7MB.

> LZW is the default compression method for TIFF in OpenCV library.

For the aforementioned reasons, TIFF is generally avoided for image velocimetry purposes, but it can rival PNG if LZW compression is used.

However, TIFF format supports various features not common with some other formats like 16bit images, image layering/multipage images, transparency, standardized metadata assignment, etc. Because of these additional capabilities, TIFF format is often used to represent results of survey/photogrammetry analyses such as digital elevation models (DEMs).

3.5 BMP

BMP or bitmap (or "bump") is a lossless image format commonly used on Windows devices. While such images can be compressed using a lossless algorithm, this is uncommon and they are usually found uncompressed making their file size substantial. Actually, BMP image file size should be exactly the same as of the uncompressed TIFF variant, which severely limits its usability for image velocimetry purposes.

3.6 GIF

While rarely used for color images because it only supports 256 colors per image, GIF format (or "Graphical Interchange Format") can be safely used for lossless compression and storage of 8bit grayscale images - which are commonly used for image velocimetry purposes. Like the TIFF format, it used LZW compression algorithm.

Continue to next chapter: Image colorspaces >>

or

<< Back to MAIN notebook