CE 712: Digital Image Processing for Remotely Sensed Data

Tutorial 3 - Image Handling

Submission: 28th August, 2025

Email: ce712.lab@gmail.com

The objectives of this tutorial are to:

- Demonstrate different types of image resolutions,
- Illustrate the effect of changing pixel density
- Compute reflectance using Landsat-9 data.

Overview of Resolutions

1. Spatial resolution:

   the size of a single pixel on the ground. High spatial resolution in imaging is the ability to distinguish fine details in an image, hence a greater level of detail and sharpness compared to images with lower spatial resolution.

2. Temporal resolution:

   the frequency at which data is acquired or captured for the same location over time. In Landsat, it is the time taken for a satellite to revisit a specific area and collect new data.

3. Spectral resolution:

   refers to the specific wavelengths, or bands, that a sensor can detect. For Landsat satellites, the spectral specifications can be accessed here https://landsat.gsfc.nasa.gov/satellites/landsat-9/landsat-9-instruments/landsat-9-spectral-specifications/

4. Radiometric resolution:

   is the sensor's ability to distinguish between slight differences in intensity. This is determined by the number of bits used to store the data for each pixel. A higher radiometric resolution means the sensor can record a wider range of brightness levels.

Task 1:

To study the resolution of a jpeg./png image, and alter the pixel density via up sampling and down sampling.

Step 1: Import libraries

```
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt
```

Step 2: read the image

```
image_path = r'E:\CE712_2025\tutorial3_image_handling\final\lake.jpg'
img = Image.open(image_path)
img_array = np.array(img)
```

Step 3: determine the radiometric and spectral resolution

```
mode = img.mode
bit_depth = img_array.dtype.itemsize * 8
print(bit_depth)

# 4. Spectral resolution (number of bands)
if len(img_array.shape) == 2:
    num_bands = 1  # grayscale
else:
    num_bands = img_array.shape[2]  # e.g., 3 for RGB
print("Spectral Resolution:", num_bands, "band(s)")
```

```
8
Spectral Resolution: 3 band(s)
```

Step 4: up sampling and down sampling using bilinear interpolation

```
#change the pixel density --> upsampling and downsampling the image
downscaled = img.resize((img.width // 10, img.height // 10), Image.BILINEAR)

# 3. Upscale back (double resolution)
upscaled = downscaled.resize((img.width * 10, img.height * 10), Image.BILINEAR)
```

Step 5: plot the final results for analysis

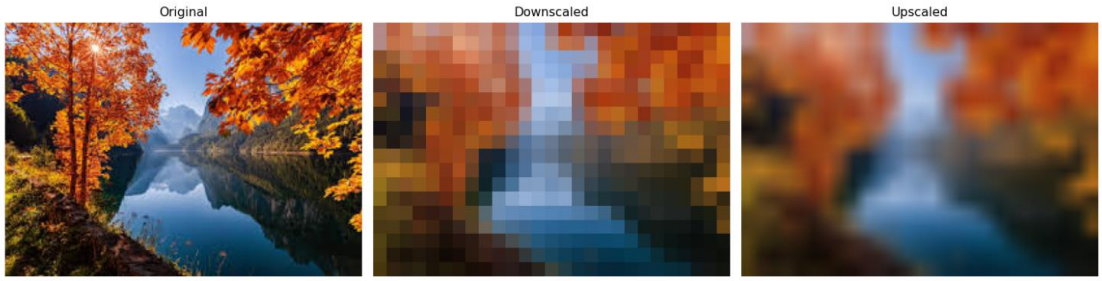
```
# 4. Display the three images side by side
fig, axes = plt.subplots(1, 3, figsize=(15, 5))

axes[0].imshow(img, cmap='gray')
axes[0].set_title('Original')
axes[0].axis('off')

axes[1].imshow(downscaled, cmap='gray')
axes[1].set_title('Downscaled')
axes[1].axis('off')

axes[2].imshow(upscaled, cmap='gray')
axes[2].set_title('Upscaled')
axes[2].axis('off')

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




```
print(img.size)
```

```
(275, 183)
```

```
print(downscaled.size)
```

```
(27, 18)
```

```
print(upscaled.size)
```

```
(2750, 1830)
```

Perform the same task with a satellite (Landsat) image

Step 1: Import libraries

```
import rasterio
import matplotlib.pyplot as plt
from rasterio.enums import Resampling
```

Step 2: define the image path (replace with your file path)

```
# Define the path to your input image
image_path = r'E:\CE712_2025\tutorial3_image_handling\final\clipped1.tif'
```

Step 3: compute spatial, radiometric and spectral resolution

```
with rasterio.open(image_path) as src:
    # Read the original image data from the first band
    original = src.read(1)

    # Print basic image properties
    print("--- Original Image Properties ---")
    print(f"Spatial resolution: {src.res[0]} m x {src.res[1]} m per pixel")
    print(f"Radiometric: dtype = {src.dtypes[0]}")
    print(f"Dimensions: {src.height} rows x {src.width} columns")
```

```
--- Original Image Properties ---
Spatial resolution: 30.0 m x 30.0 m per pixel
Radiometric: dtype = uint16
Dimensions: 134 rows x 249 columns
```

Step 4: perform up sampling and down sampling to alter pixel density of the image (experiment with different numbers to observe changes)

```
with rasterio.open(image_path) as src:
    # Read the original image data
    original = src.read(1)

    # Downscale the image to 1/20th of its original resolution
    # We use integer division // to ensure the new dimensions are integers
    downscaled = src.read(
        out_shape=(1, src.height // 20, src.width // 20),
        resampling=Resampling.bilinear
    )[0]

    # Upscale the image to 20 times its original resolution
    # This increases the pixel count
    upscaled = src.read(
        out_shape=(1, src.height * 20, src.width * 20),
        resampling=Resampling.bilinear
    )[0]
```

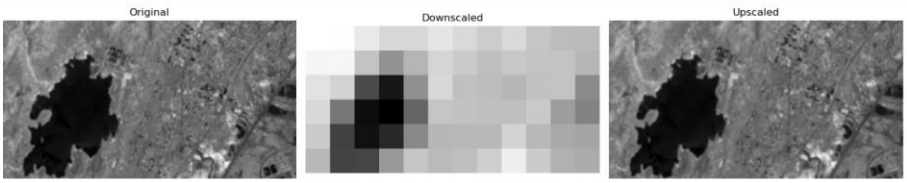
```
# Display the three images side-by-side for comparison
fig, axes = plt.subplots(1, 3, figsize=(15, 5))

axes[0].imshow(original, cmap='gray')
axes[0].set_title('Original')
axes[0].axis('off')

axes[1].imshow(downscaled, cmap='gray')
axes[1].set_title('Downscaled')
axes[1].axis('off')

axes[2].imshow(upscaled, cmap='gray')
axes[2].set_title('Upscaled')
axes[2].axis('off')

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



Step 5: Numerically show the dimensions of resulting images

```
# Print the dimensions to numerically show the change
print(f"Original: {original.shape[0]} rows x {original.shape[1]} columns")
print(f"Downscaled: {downscaled.shape[0]} rows x {downscaled.shape[1]} columns")
print(f"Upscaled: {upscaled.shape[0]} rows x {upscaled.shape[1]} columns")
```

```
Original: 134 rows x 249 columns
Downscaled: 6 rows x 12 columns
Upscaled: 2680 rows x 4980 columns
```

Question:

1. Visually compare the upscaled image and original image. In few lines, explain why increase in number of pixels has not resulted in increase of visual clarity.

Task 2: Analysing Surface Reflectance using Landsat Image Handling

The TOA (Top of Atmosphere) reflectance formula converts digital numbers (DNs) from satellite imagery to reflectance values. The basic formula involves using radiometric calibration coefficients (multiplicative and additive factors) from the image metadata and the scene centre sun elevation angle.

$$\text{Reflectance} = \frac{(\text{Multiplier} \times \text{DN}) + \text{Addend}}{\sin(\text{solar elevation angle})}$$

Similarly, radiance can be determined using the following formula

$$\text{Radiance} = (\text{Multiplier X DN}) + \text{Addend}$$

Step 1: import libraries

```
#import libraries
import rasterio
import matplotlib.pyplot as plt
import numpy as np
```

Step 2: define the image path (replace with your file path) and read the file

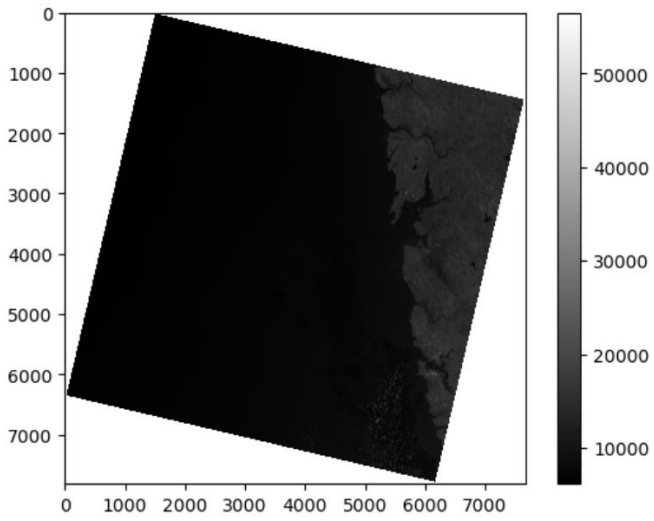
```
# 1) Open the file
input_dn_image_path = r"E:\CE712_2025\tutorial3_image_handling\final\LC09_L1TP_148047_20250418_20250418_02_T1_B5.tif"
```

```
with rasterio.open(input_dn_image_path) as src:
    dn_image = src.read(1).astype(float)
    no_data_value = src.nodata
    if no_data_value is not None:
        dn_image = np.ma.masked_equal(dn_image, no_data_value)
```

Step 3: plot the Landsat image (Band 5 – NIR is used here)

```
#plot the full landsat image (NIR Band)
plt.imshow(dn_image, cmap = 'gray')
plt.colorbar()
```

```
<matplotlib.colorbar.Colorbar at 0x194866fa3f0>
```



Step 4: Crop the image as per your area of interest

```
nir_cropped = dn_image[3300:3800,5800:6200]
```

Step 5: for the cropped image, compute radiance and reflectance

```
#define constants for reflectance calculation
mult_ref = 2.0000E-05 #gain
add_ref = -0.100000 #offset
angle_degrees = 65.51727527
angle_rad = np.deg2rad(angle_degrees)
```

```
#calculate reflectance
ref_image = (mult_ref * nir_cropped + add_ref) / np.sin(angle_rad)
ref_image[ref_image < 0] = np.nan
```

```
#define constants for radiance calculation
mult_rad = 6.0737E-03
add_rad = -30.36854
```

```
#calculate radiance
radiance_image5 = mult_rad * nir_cropped + add_rad
radiance_image5[radiance_image5 < 0] = np.nan
```

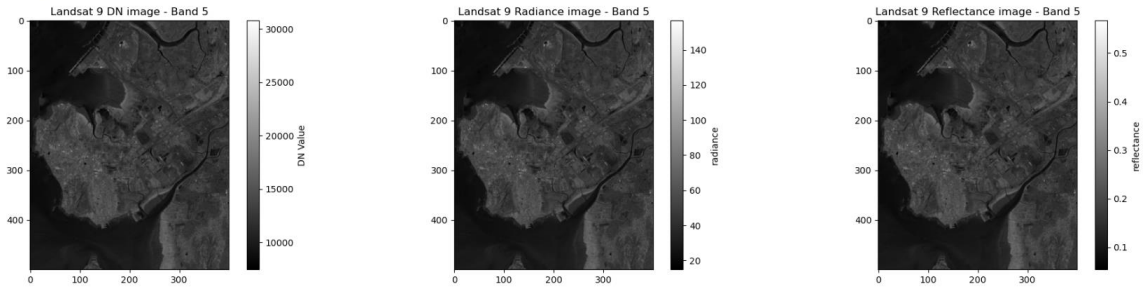
Step 6: plot the DN, radiance and reflectance images

```
fig, (ax1, ax2, ax3) = plt.subplots(1,3, figsize=(24,5))
dn_plot5 = ax1.imshow(nir_cropped, cmap='gray')
ax1.set_title('Landsat 9 DN image - Band 5')
fig.colorbar(dn_plot5, ax=ax1, label='DN Value')

radiance_plot5 = ax2.imshow(radiance_image5, cmap='gray')
ax2.set_title('Landsat 9 Radiance image - Band 5')
fig.colorbar(radiance_plot5, ax=ax2, label='radiance')

reflectance_plot5 = ax3.imshow(ref_image, cmap='gray')
ax3.set_title('Landsat 9 Reflectance image - Band 5')
fig.colorbar(reflectance_plot5, ax=ax3, label='reflectance')
```

```
<matplotlib.colorbar.Colorbar at 0x1948e7c6540>
```



Assignment

1. Compute the surface reflectance for all bands of Landsat-9 and compare the results across these bands.
2. Select a homogenous region of your interest from Landsat 9 image.
3. Plot the spectral reflectance curve of this region across all the given bands