# Correlation Assignment

Austin Gardiner

October 3, 2025

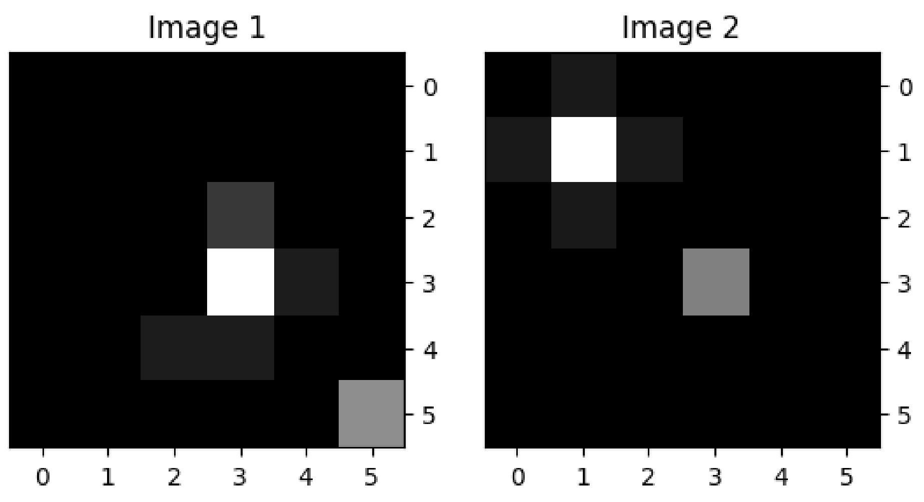

Figure 1: Original images

## 1 How do you interpret C(r,s)

Comparing the images, it can be seen that the main pattern is moving up and to the left by approximately 2 pixels. Looking at the correlation matrix calculated by the code, the biggest correlation value is found at (r,s) = (-2,-2). This means that a negative value of r means the particles are moving to the left (positive = right). For s, a negative value of s means the particles are moving up (positive = down).

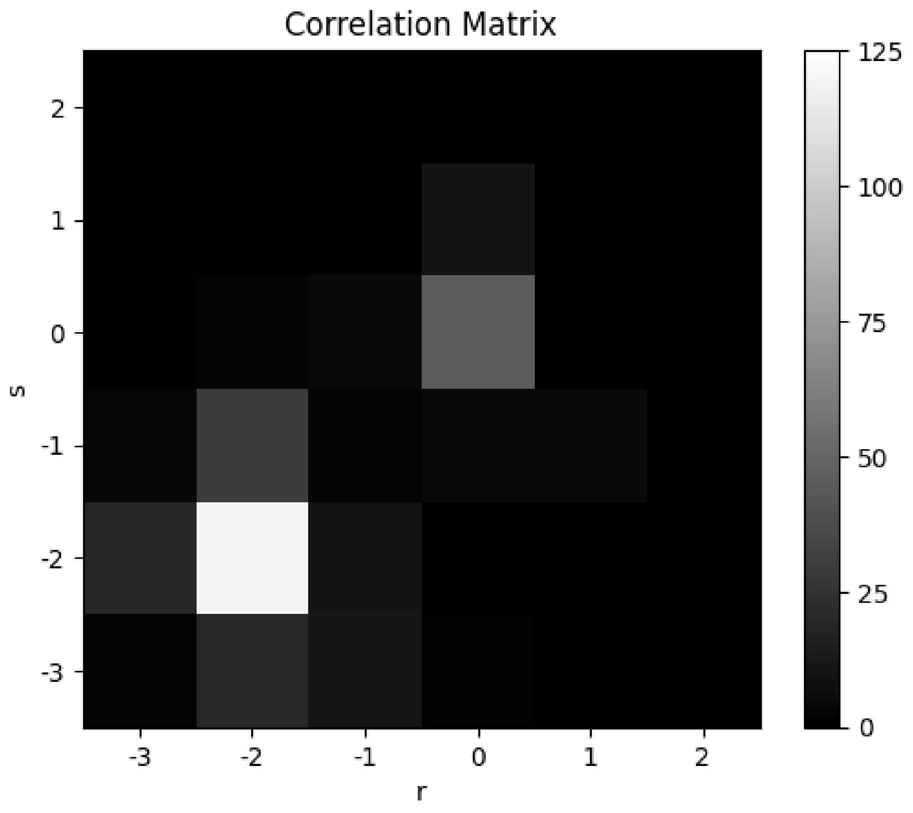

Figure 2: Correlation results

## 2 What direction and how far do you estimate the displacement to be without subpixel estimation (i.e. to the nearest pixel)?

Without subpixel estimation, the image is moving up and to the left by two pixels. This can be seen in Fig. 2 or by comparing the values in the correlation matrix printed below. The printed correlation matrix has (r,s) = (-3,-3) in the top left corner, with r increasing to the right and s increasing as you move down.

```
Correlation Matrix:
[[  2.  20.  11.   1.   0.   0.]
 [ 19. 119.  10.   0.   0.   0.]
 [  3.  29.   2.   5.   5.   0.]
 [  0.   2.   5.  45.   0.   0.]
 [  0.   0.   0.  10.   0.   0.]
 [  0.   0.   0.   0.   0.   0.]]
```

## 3 If $D_I$ is 6, as in this case, how big must $IA_1$ and $IA_2$ be?

With $D_I = 6$ and $\frac{D_I}{2} = 3$, that leaves r and s as [-3, -2, -1, 0, 1, 2]. Using the maximum and minimum values of r and s, that means there need to be (3 + 6 + 2) = 11 horizontal pixels and (3 + 6 + 2) = 11 vertical pixels at minimum. An 11x11 image would be 121 total pixels.

## 4 Code

```
import numpy as np
import matplotlib.pyplot as plt

# Define the two images as 6x6 matrices
A1 = [[0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0], [0, 0, 0, 2, 0, 0], [0, 0, 0, 9, 1, 0], [0, 0, 1, 1, 0, 0]
A2 = [[0, 1, 0, 0, 0, 0], [1, 10, 1, 0, 0, 0], [0, 1, 0, 0, 0, 0], [0, 0, 0, 5, 0, 0], [0, 0, 0, 0, 0,

# Convert lists to numpy arrays for easier manipulation
A1 = np.array(A1)
A2 = np.array(A2)

def calculate_correlation(image1, image2):
    # Assuming image1 and image2 are square matrices of the same size
    # Get the dimensions of the images
    D_I = image1.shape[0]
    r = np.arange(-D_I // 2, D_I // 2)
    s = np.arange(-D_I // 2, D_I // 2)

    # Initialize the correlation matrix
    C = np.zeros((D_I, D_I))

    # Calculate the correlation
    for i in range(D_I):
        for j in range(D_I):
            if image1[i][j] != 0:
                for r_index, v in enumerate(r):
                    if (i + v) >= 0 and (i + v) < D_I:
                        for u_index, u in enumerate(s):
                            if (j + u) >= 0 and (j + u) < D_I:
                                C[r_index][u_index] += image1[i][j] * image2[i + v][j + u]
    return C

# Calculate the correlation matrix
correlation_matrix = calculate_correlation(A1, A2)
print("Correlation Matrix:")
print(correlation_matrix)

# Plot the images
plt.subplot(1, 2, 1)
plt.title("Image 1")
plt.imshow(A1, cmap='gray')
plt.xticks(np.arange(A1.shape[0]))
plt.gca().yaxis.set_ticks_position('right')
plt.subplot(1, 2, 2)
plt.title("Image 2")
plt.imshow(A2, cmap='gray')
plt.xticks(np.arange(A1.shape[0]))
plt.gca().yaxis.set_ticks_position('right')
plt.show()

# Get the dimensions of the correlation matrix
D = correlation_matrix.shape[0]
center = D // 2
ticks = np.arange(D) - center

# Plot the correlation matrix with centered axes and labels
plt.figure()
plt.title("Correlation Matrix")
im = plt.imshow(correlation_matrix, cmap='gray', origin='lower', vmin=0, vmax=125)
# Adjust ticks to be centered around 0
plt.xticks(np.arange(D), ticks)
plt.yticks(np.arange(D), ticks)
plt.gca().yaxis.set_ticks_position('right')
plt.gca().yaxis.set_label_position('right')
plt.xlabel('r')
plt.ylabel('s')
# Colorbar with ticks at 0,25,...,125
plt.colorbar(im, fraction=0.046, pad=0.12, ticks=np.arange(0,126,25))
plt.show()
```