

Report Computer Vision

Rawan Farouk 16001938

October 25, 2024

1 Canny Edge Detection

I chose the Canny Edge Detection method. Canny is a popular multi-stage edge detection algorithm known for its ability to detect a wide range of edges in images with high accuracy.

1.1 Why I chose it?

One of the primary reasons for selecting the Canny method is its ability to minimize false edges while preserving true edges, which provides a clear and refined detection of the edges in complex landscapes, such as the given grayscale image. Additionally, Canny offers adjustable thresholds that allow for precise control over the sensitivity of the edge detection process, making it versatile for various image types.

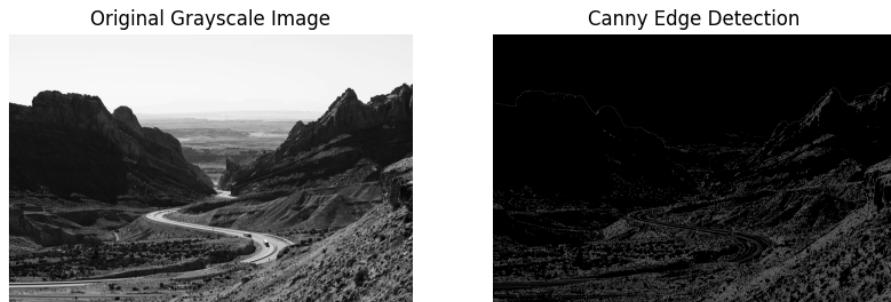


Figure 1: Original vs Canny Edge Detection

2 Types of Noises - Average Filtering

2.1 Salt and Pepper Noise

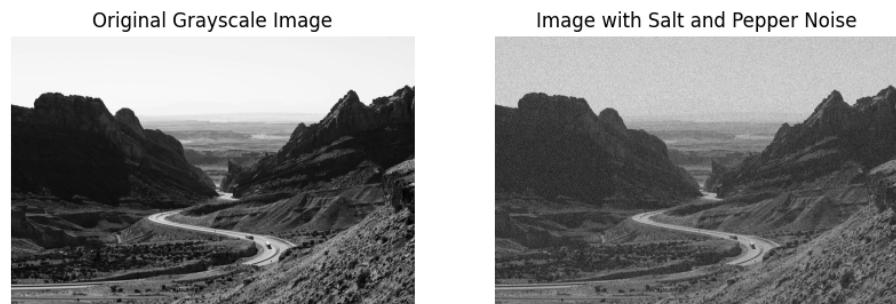


Figure 2: Original vs Salt and Pepper Noise

2.1.1 Average Filter 3x3 - 5x5 - 9x9

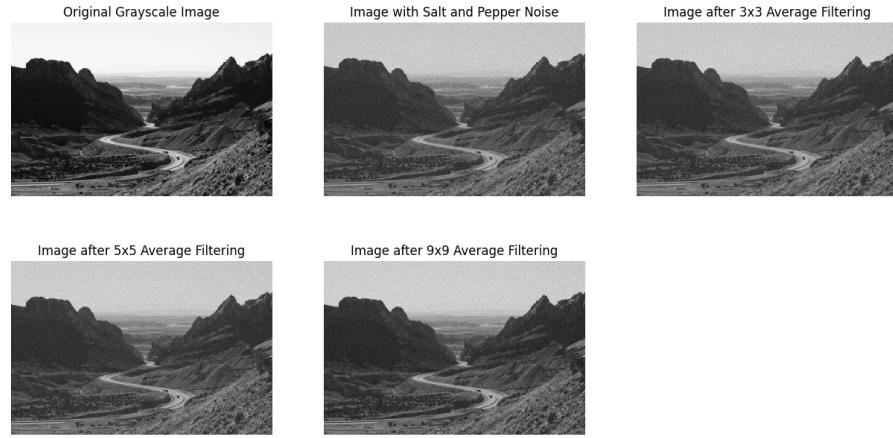


Figure 3: Original vs Salt and Pepper Noise vs Average Filtering

The effectiveness of noise reduction and image quality vary with different filter sizes (3x3, 5x5, and 9x9) used for Salt and Pepper noise reduction. The 3x3 filter provides minimal noise reduction but retains most of the image detail, making it suitable for areas requiring fine details, though it allows some noise to remain. The 5x5 filter offers a balance between noise suppression and detail preservation, producing smoother images with moderate noise removal. The 9x9 filter effectively reduces noise but significantly blurs fine details, making it less ideal for images where maintaining detail is important. Smaller filters preserve detail, while larger filters are better for noise removal.

2.2 Gaussian Noise

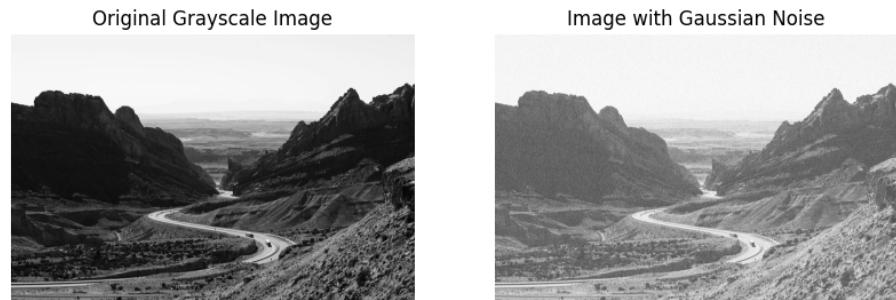


Figure 4: Original vs Guassian Noise

2.2.1 Average Filter 3x3 - 5x5 - 9x9



Figure 5: Original vs Gaussian Noise vs Average Filtering

In cases where an image is affected by Gaussian noise, the size of the filter significantly impacts noise reduction and detail preservation. A 3x3 filter offers minimal noise reduction while preserving fine details, allowing some noise to remain visible. This is useful when retaining image details is more important than removing noise. The 5x5 filter strikes a balance between reducing noise and preserving detail, producing a smoother image but introducing slight blurring. The 9x9 filter removes the most noise but at the cost of losing fine details due to increased blurring. It is most suitable when noise reduction is prioritized over maintaining image features.

2.3 Poisson Noise

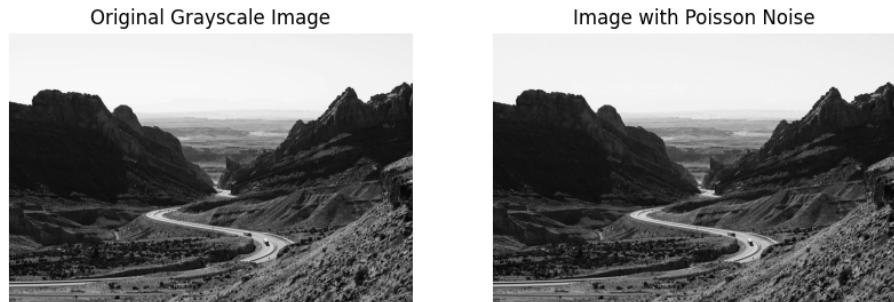


Figure 6: Original vs Poisson Noise

2.3.1 Average Filter 3x3 - 5x5 - 9x9



Figure 7: Original vs Poisson Noise vs Average Filtering

When dealing with Poisson noise, the performance of average filters (3x3, 5x5, 9x9) varies with their size. The 3x3 filter provides minimal noise reduction, preserving fine details while leaving noticeable noise in high-intensity areas. It offers a balance between smoothing and detail retention. The 5x5 filter reduces Poisson noise more effectively, resulting in a smoother image, though some blurring of finer details starts to appear. The 9x9 filter is the most effective for noise reduction, significantly smoothing the image but causing substantial blurring, particularly in areas with sharp edges or fine textures. As the filter size increases, noise reduction improves, but at the cost of image sharpness and detail.

2.4 Random Noise



Figure 8: Original vs Random Noise

2.4.1 Average Filter 3x3 - 5x5 - 9x9

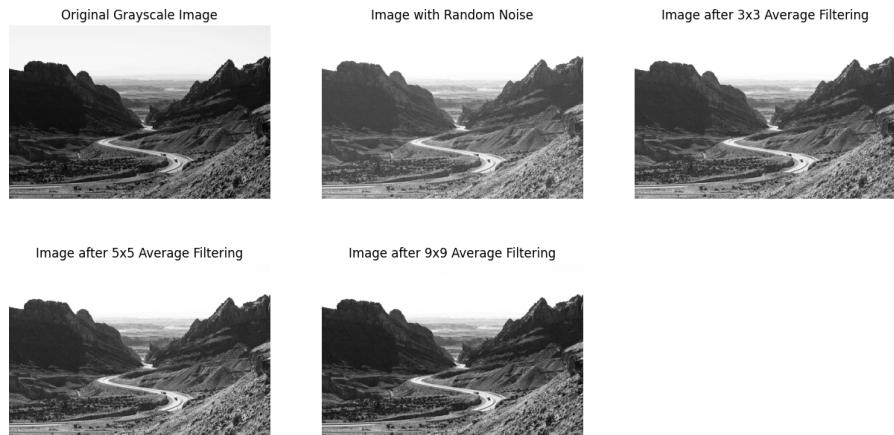


Figure 9: Original vs Random Noise vs Average Filtering

When Random Noise is applied to an image, the effectiveness of average filters depends on their size. The 3x3 filter offers minimal smoothing, reducing some noise while preserving fine details, but leaving noticeable noise. This filter is ideal for preserving details when noise reduction is less critical. The 5x5 filter achieves a better balance between noise reduction and detail retention, smoothing out more noise while maintaining reasonable image sharpness. The 9x9 filter is the most effective at removing random noise, producing a much smoother image, but with considerable blurring of fine details. Larger filters reduce more noise but sacrifice image sharpness and detail.

3 Analysis of the Effects of Different Noises and Filter Sizes on Image Quality

When the various types of noise application—Salt and Pepper, Gaussian, Poisson, and Random—are applied to the image, each affects the quality of the image differently; then the application of averaging filtering with variable filter size again changes the result.

1. **Salt and Pepper Noise:** It is a kind of noise that adds sharp black and white pixels randomly in the image. It creates huge contrasts, hence affecting the quality of the image very much.
 - **3x3 filter:** This cleans off some of the noise but still retains much of the details, so that most of the noise artifacts remain visible.
 - **5x5 filter:** It smoothes out more of the noise but maintains reasonable sharpness; it balances noise reduction and detail pretty well.
 - **9x9 filter:** Noise is largely removed, but there is marked blurring of the finer details, which makes the image overly smooth.
2. **Gaussian Noise:** Happens to be less heavy than Salt and Pepper noise. The noise generally spreads in the image in a uniform manner, providing a grainy look to the image.
 - **3x3 filter:** A small smoothing out of noise and lots of the image details remain preserved.
 - **5x5 filter:** Smoothing out of noise is way more. The image is cleaner but at the cost of minor loss in details.
 - **9x9 filter:** This filter removes most of the Gaussian noise but, in turn, severely causes edge and fine structure blurring.
3. **Poisson Noise:** This noise, since it depends on the intensity of a pixel, adds small variations that increase in higher intensities. The noise will appear fewer in darker regions and more conspicuous in brighter regions.
 - **3x3 filter:** Grossly less noise reduction, leaving most of the Poisson noise still visible.
 - **5x5 filter:** Balances noise removal and detail preservation, notably reducing the noise in bright areas.
 - **9x9 filter:** Removes most of the Poisson noise at the cost of image details, giving a very smooth yet somewhat blurred image.

4. **Random Noise:** Random noise introduces uniformly distributed noise that appears chaotic across the image. It impacts all regions of the image equally.

- **3x3 filter:** Has minimal smoothing with quite obvious random noise.
- **5x5 filter:** Removes most of the noise without eliminating moderate details. It's a good balance between noise removal and sharpness.
- **9x9 filter:** Removes nearly all the random noise but results in a heavy blur on the image, quite noticeable in regions of fine detail.

4 Overall Conclusions on the Impact of Different Noise Types and Filtering Techniques

Besides, the influence of various types of noise on image quality can be drastically different: Salt and Pepper noise results in sharp pixel-level distortions, Gaussian and Poisson noise cause more evenly distributed variations, and Random noise affects the whole image chaotically.

Average filtering is quite an efficient method of noise reduction; however, its efficiency directly depends on the size of the filter applied:

- With a small size, it provides basic noise reduction but maintains image details and can leave some noise artifacts.
- A medium size guarantees a good balance between noise reduction and keeping image sharpness, making this type of filter suitable for most applications.
- Large filters (9x9) are quite effective in removing noise, but this is often at the cost of losing a lot of image details and hence yielding very smooth, blurred images.

In summary, the nature of noise and size of filter selected should be compatible with the requirement on image quality. Smaller filters may be advisable when the preservation of detail is critical, while maximum noise reduction may call for larger filters, though at the expense of fine detail reduction. Different types of noise may also require different filtering strategies to achieve optimal results.

A Code Listing

Here is the code as part of the appendix:

```
File Edit Selection View Go Run Terminal Help ↵ → cv task 1
Settings code_1.py ✘
◆ code_1.py > ...
1 import cv2
2 import numpy as np
3 import random
4 from matplotlib import pyplot as plt
5
6 # Use the absolute path to the image
7 image_path = 'C:/Users/user/Desktop/cv task 1/gray.jpg'
8
9 image_cv = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
10
11 # Apply Canny edge detection
12 edges = cv2.Canny(image_cv, 100, 200)
13
14 # Plot the original image and the result of the edge detection
15 plt.figure(figsize=(10, 5))
16
17 # Original grayscale image
18 plt.subplot(1, 2, 1)
19 plt.imshow(image_cv, cmap='gray')
20 plt.title('Original Grayscale Image')
21 plt.axis('off')
22
23 # Canny edge detection result
24 plt.subplot(1, 2, 2)
25 plt.imshow(edges, cmap='gray')
26 plt.title('Canny Edge Detection')
27 plt.axis('off')
28
29 # Display the results
30 plt.show()
31
32 # Function to add Salt and Pepper noise
33 def add_salt_and_pepper_noise(image, salt_prob, pepper_prob):
34     noisy_image = np.copy(image)
35     total_pixels = image.size
36
37     # Salt noise
38     num_salt = np.ceil(salt_prob * total_pixels)
39     coords = [np.random.randint(0, i - 1, int(num_salt)) for i in image.shape]
40     noisy_image[coords[0], coords[1]] = 255
41
42     # Pepper noise
43     num_pepper = np.ceil(pepper_prob * total_pixels)
44     coords = [np.random.randint(0, i - 1, int(num_pepper)) for i in image.shape]
45     noisy_image[coords[0], coords[1]] = 0
46
47     return noisy_image
48
49 # Add Salt and Pepper noise to the image
50 salt_prob = 0.2 # Probability of salt noise
51 pepper_prob = 0.2 # Probability of pepper noise
52 noisy_image = add_salt_and_pepper_noise(image_cv, salt_prob, pepper_prob)
53
54 # Apply average filtering with different filter sizes
55 filtered_image_3x3 = cv2.blur(noisy_image, (3, 3))
56 filtered_image_5x5 = cv2.blur(noisy_image, (5, 5))
57 filtered_image_9x9 = cv2.blur(noisy_image, (9, 9))
58
59 # Plot 1: Original vs Noisy image
60 plt.figure(figsize=(10, 5))
61
62 # Original grayscale image
63 plt.subplot(1, 2, 1)
64 plt.imshow(image_cv, cmap='gray')
65 plt.title('Original Grayscale Image')
66 plt.axis('off')
67
68 # Noisy image with Salt and Pepper noise
69 plt.subplot(1, 2, 2)
70 plt.imshow(noisy_image, cmap='gray')
71 plt.title('Noisy Image with Salt and Pepper noise')
72 plt.axis('off')
73
74 # Plot 2: Filtered images
75 plt.figure(figsize=(10, 3))
76 plt.subplot(1, 3, 1)
77 plt.imshow(filtered_image_3x3, cmap='gray')
78 plt.title('Filtered Image (3x3)')
79 plt.axis('off')
80
81 plt.subplot(1, 3, 2)
82 plt.imshow(filtered_image_5x5, cmap='gray')
83 plt.title('Filtered Image (5x5)')
84 plt.axis('off')
85
86 plt.subplot(1, 3, 3)
87 plt.imshow(filtered_image_9x9, cmap='gray')
88 plt.title('Filtered Image (9x9)')
89 plt.axis('off')
90
91 # Display the results
92 plt.show()

Ln 71, Col 46 Spaces: 4 UTF-8 CR/LF { Python 3.7.4 64-bit (3.7.4:pyenv) ⚡ Go Live 🔍 AI Code Chat Q
Type here to search 🌱 🌐 🌐 🌐 🌐 🌐 🌐 🌐 🌐 🌐 ENG 4:39 PM Wird in Kürze gestartet 10/25/2024
```

```
File Edit Selection View Go Run Terminal Help ↵ → cv task 1
Settings code_1.py ✘
◆ code_1.py > ...
31
32 # Function to add Salt and Pepper noise
33 def add_salt_and_pepper_noise(image, salt_prob, pepper_prob):
34     noisy_image = np.copy(image)
35     total_pixels = image.size
36
37     # Salt noise
38     num_salt = np.ceil(salt_prob * total_pixels)
39     coords = [np.random.randint(0, i - 1, int(num_salt)) for i in image.shape]
40     noisy_image[coords[0], coords[1]] = 255
41
42     # Pepper noise
43     num_pepper = np.ceil(pepper_prob * total_pixels)
44     coords = [np.random.randint(0, i - 1, int(num_pepper)) for i in image.shape]
45     noisy_image[coords[0], coords[1]] = 0
46
47     return noisy_image
48
49 # Add Salt and Pepper noise to the image
50 salt_prob = 0.2 # Probability of salt noise
51 pepper_prob = 0.2 # Probability of pepper noise
52 noisy_image = add_salt_and_pepper_noise(image_cv, salt_prob, pepper_prob)
53
54 # Apply average filtering with different filter sizes
55 filtered_image_3x3 = cv2.blur(noisy_image, (3, 3))
56 filtered_image_5x5 = cv2.blur(noisy_image, (5, 5))
57 filtered_image_9x9 = cv2.blur(noisy_image, (9, 9))
58
59 # Plot 1: Original vs Noisy image
60 plt.figure(figsize=(10, 5))
61
62 # Original grayscale image
63 plt.subplot(1, 2, 1)
64 plt.imshow(image_cv, cmap='gray')
65 plt.title('Original Grayscale Image')
66 plt.axis('off')
67
68 # Noisy image with Salt and Pepper noise
69 plt.subplot(1, 2, 2)
70 plt.imshow(noisy_image, cmap='gray')
71 plt.title('Noisy Image with Salt and Pepper noise')
72 plt.axis('off')
73
74 # Plot 2: Filtered images
75 plt.figure(figsize=(10, 3))
76 plt.subplot(1, 3, 1)
77 plt.imshow(filtered_image_3x3, cmap='gray')
78 plt.title('Filtered Image (3x3)')
79 plt.axis('off')
80
81 plt.subplot(1, 3, 2)
82 plt.imshow(filtered_image_5x5, cmap='gray')
83 plt.title('Filtered Image (5x5)')
84 plt.axis('off')
85
86 plt.subplot(1, 3, 3)
87 plt.imshow(filtered_image_9x9, cmap='gray')
88 plt.title('Filtered Image (9x9)')
89 plt.axis('off')
90
91 # Display the results
92 plt.show()

Ln 71, Col 46 Spaces: 4 UTF-8 CR/LF { Python 3.7.4 64-bit (3.7.4:pyenv) ⚡ Go Live 🔍 AI Code Chat Q
Type here to search 🌱 🌐 🌐 🌐 🌐 🌐 🌐 🌐 🌐 🌐 🌐 🌐 ENG 4:40 PM Wird in Kürze gestartet 10/25/2024
```

```
File Edit Selection View Go Run Terminal Help cv task 1
code_1.py > ...
68 # Noisy image with Salt and Pepper noise
69 plt.subplot(1, 2, 2)
70 plt.imshow(noisy_image, cmap='gray')
71 plt.title('Image with Salt and Pepper Noise')
72 plt.axis('off')
73
74 # Show the first plot
75 plt.show()
76
77 # Plot 2: Original vs Noisy vs Filtered images
78 plt.figure(figsize=(20, 10))
79
80 # Original grayscale image
81 plt.subplot(2, 3, 1)
82 plt.imshow(image_cv, cmap='gray')
83 plt.title('Original Grayscale Image')
84 plt.axis('off')
85
86 # Noisy image with Salt and Pepper noise
87 plt.subplot(2, 3, 2)
88 plt.imshow(noisy_image, cmap='gray')
89 plt.title('Image with Salt and Pepper Noise')
90 plt.axis('off')
91
92 # Filtered image with 3x3 average filtering
93 plt.subplot(2, 3, 3)
94 plt.imshow(filtered_image_3x3, cmap='gray')
95 plt.title('Image after 3x3 Average Filtering')
96 plt.axis('off')
97
98 # Filtered image with 5x5 average filtering
99 plt.subplot(2, 3, 4)
100 plt.imshow(filtered_image_5x5, cmap='gray')
101 plt.title('Image after 5x5 Average Filtering')
102 plt.axis('off')
103
104 # Filtered image with 9x9 average filtering
105 plt.subplot(2, 3, 5)
106 plt.imshow(filtered_image_9x9, cmap='gray')
107 plt.title('Image after 9x9 Average Filtering')
108 plt.axis('off')
109
110 # Show the second plot
111 plt.show()
112
113
114
115 # Function to add Gaussian noise
116 def add_gaussian_noise(image, mean=0, var=0.01):
117     row, col = image.shape
118     sigma = var**0.5 # Standard deviation
119     gaussian = np.random.normal(mean, sigma, (row, col)) # Gaussian noise
120     noisy_image = image + gaussian * 255 # Scale to image pixel range
121     noisy_image = np.clip(noisy_image, 0, 255) # Clip pixel values to stay within valid range
122     return noisy_image.astype(np.uint8)
123
124 # Add Gaussian noise to the image
125 mean = 0.3 # Mean of the Gaussian distribution
126 var = 0.1 # Variance of the Gaussian noise
127 noisy_image = add_gaussian_noise(image_cv, mean, var)
128
129 # Apply average filtering with different filter sizes
130 filtered_image_3x3 = cv2.blur(noisy_image, (3, 3))
131 filtered_image_5x5 = cv2.blur(noisy_image, (5, 5))
132 filtered_image_9x9 = cv2.blur(noisy_image, (9, 9))
133
134 # Plot 1: Original vs Noisy image
135 plt.figure(figsize=(10, 5))
136
137 # Original grayscale image
138 plt.subplot(1, 2, 1)
139 plt.imshow(image_cv, cmap='gray')
140 plt.title('Original Grayscale Image')

In 71, Col 46 Spaces: 4 UTF-8 CRLF {} Python 3.7.4 64-bit (3.7.4: pyenv) ⚡ Go Live ⚡ AI Code Chat
Wird in Kürze gestartet... ENG 4:40 PM 10/25/2024
```

```
File Edit Selection View Go Run Terminal Help cv task 1
code_1.py > ...
105 # Filtered image with 9x9 average filtering
106 plt.imshow(filtered_image_9x9, cmap='gray')
107 plt.title('Image after 9x9 Average Filtering')
108 plt.axis('off')
109
110 # Show the second plot
111 plt.show()
112
113
114
115 # Function to add Gaussian noise
116 def add_gaussian_noise(image, mean=0, var=0.01):
117     row, col = image.shape
118     sigma = var**0.5 # Standard deviation
119     gaussian = np.random.normal(mean, sigma, (row, col)) # Gaussian noise
120     noisy_image = image + gaussian * 255 # Scale to image pixel range
121     noisy_image = np.clip(noisy_image, 0, 255) # Clip pixel values to stay within valid range
122     return noisy_image.astype(np.uint8)
123
124 # Add Gaussian noise to the image
125 mean = 0.3 # Mean of the Gaussian distribution
126 var = 0.1 # Variance of the Gaussian noise
127 noisy_image = add_gaussian_noise(image_cv, mean, var)
128
129 # Apply average filtering with different filter sizes
130 filtered_image_3x3 = cv2.blur(noisy_image, (3, 3))
131 filtered_image_5x5 = cv2.blur(noisy_image, (5, 5))
132 filtered_image_9x9 = cv2.blur(noisy_image, (9, 9))
133
134 # Plot 1: Original vs Noisy image
135 plt.figure(figsize=(10, 5))
136
137 # Original grayscale image
138 plt.subplot(1, 2, 1)
139 plt.imshow(image_cv, cmap='gray')
140 plt.title('Original Grayscale Image')

In 71, Col 46 Spaces: 4 UTF-8 CRLF {} Python 3.7.4 64-bit (3.7.4: pyenv) ⚡ Go Live ⚡ AI Code Chat
Wird in Kürze gestartet... ENG 4:41 PM 10/25/2024
```

The screenshot shows a Jupyter Notebook interface with the following code:

```
File Edit Selection View Go Run Terminal Help ← → ⌘ cv task 1
```

```
Settings code_1.py X
```

```
code_1.py > ...
```

```
137 # Original grayscale image
138 plt.subplot(1, 2, 1)
139 plt.imshow(image_cv, cmap='gray')
140 plt.title('Original Grayscale Image')
141 plt.axis('off')
142
143 # Noisy image with Gaussian noise
144 plt.subplot(1, 2, 2)
145 plt.imshow(noisy_image, cmap='gray')
146 plt.title('Image with Gaussian Noise')
147 plt.axis('off')
148
149 # Show the first plot (Original vs Noisy)
150 plt.show()
151
152 # Plot 2: Original vs Noisy vs Filtered images
153 plt.figure(figsize=(20, 10))
154
155 # Original grayscale image
156 plt.subplot(2, 3, 1)
157 plt.imshow(image_cv, cmap='gray')
158 plt.title('Original Grayscale Image')
159 plt.axis('off')
160
161 # Noisy image with Gaussian noise
162 plt.subplot(2, 3, 2)
163 plt.imshow(noisy_image, cmap='gray')
164 plt.title('Image with Gaussian Noise')
165 plt.axis('off')
166
167 # Filtered image with 3x3 average filtering
168 plt.subplot(2, 3, 3)
169 plt.imshow(filtered_image_3x3, cmap='gray')
170 plt.title('Image after 3x3 Average Filtering')
171 plt.axis('off')
172
173 # Filtered image with 5x5 average filtering
```

At the bottom, there are several status indicators: In 71, Col 46, Spaces: 4, UTF-8, CR/LF, Python 3.7.4 64-bit (3.7.4: pyenv), Go Live, Al Code Chat, and a search bar.

The screenshot shows a Jupyter Notebook interface with the following code:

```
File Edit Selection View Go Run Terminal Help ← → 🔍 cv task 1
```

```
code_1.py X
```

```
code_1.py >...
```

```
173 # Filtered image with 5x5 average filtering
174 plt.subplot(2, 3, 4)
175 plt.imshow(filtered_image_5x5, cmap='gray')
176 plt.title('Image after 5x5 Average Filtering')
177 plt.axis('off')
178
179 # Filtered image with 9x9 average filtering
180 plt.subplot(2, 3, 5)
181 plt.imshow(filtered_image_9x9, cmap='gray')
182 plt.title('Image after 9x9 Average Filtering')
183 plt.axis('off')
184
185 # show the second plot (Original vs Noisy vs Filtered)
186 plt.show()
187
188
189
190 # Function to add Poisson noise
191 def add_poisson_noise(image):
192     noisy_image = np.copy(image).astype(np.float32)
193
194     # Normalize the image to [0, 1] range
195     noisy_image = noisy_image / 255.0
196
197     # Apply Poisson noise and multiply the noise effect to make it more visible
198     noisy_image = np.random.poisson(noisy_image * 255) / 255.0 * 2 # Multiply by a factor to increase noise visibility
199
200     # Scale back to the [0, 255] range and convert to uint8
201     noisy_image = np.clip(noisy_image * 255, 0, 255).astype(np.uint8)
202
203     return noisy_image
204
205
206 # Add Poisson noise to the image
207 noisy_image = add_poisson_noise(image_cv)
208
209 # Apply average filtering with different filter sizes
```

At the bottom, there are tabs for 'Add Log', 'Select Postgres Server', 'Search Error', 'Share Code Link', and a search bar. The status bar shows 'In 71, Col 46' and other system information.

```
File Edit Selection View Go Run Terminal Help cv task 1
code_1.py > ...
209 # Apply average filtering with different filter sizes
210 filtered_image_3x3 = cv2.blur(noisy_image, (3, 3))
211 filtered_image_5x5 = cv2.blur(noisy_image, (5, 5))
212 filtered_image_9x9 = cv2.blur(noisy_image, (9, 9))
213
214 # Plot 1: Original vs Noisy image
215 plt.figure(figsize=(10, 5))
216
217 # Original grayscale image
218 plt.subplot(1, 2, 1)
219 plt.imshow(image_cv, cmap='gray')
220 plt.title('Original Grayscale Image')
221 plt.axis('off')
222
223 # Noisy image with Poisson noise
224 plt.subplot(1, 2, 2)
225 plt.imshow(noisy_image, cmap='gray')
226 plt.title('Image with Poisson Noise')
227 plt.axis('off')
228
229 # Show the first plot (Original vs Noisy)
230 plt.show()
231
232 # Plot 2: Original vs Noisy vs Filtered images
233 plt.figure(figsize=(20, 10))
234
235 # Original grayscale image
236 plt.subplot(2, 3, 1)
237 plt.imshow(image_cv, cmap='gray')
238 plt.title('Original Grayscale Image')
239 plt.axis('off')
240
241 # Noisy image with Poisson noise
242 plt.subplot(2, 3, 2)
243 plt.imshow(noisy_image, cmap='gray')
244 plt.title('Image with Poisson Noise')
245 plt.axis('off')
246
247 # Filtered image with 3x3 average filtering
248 plt.subplot(2, 3, 3)
249 plt.imshow(filtered_image_3x3, cmap='gray')
250 plt.title('Image after 3x3 Average Filtering')
251 plt.axis('off')
252
253 # Filtered image with 5x5 average filtering
254 plt.subplot(2, 3, 4)
255 plt.imshow(filtered_image_5x5, cmap='gray')
256 plt.title('Image after 5x5 Average Filtering')
257 plt.axis('off')
258
259 # Filtered image with 9x9 average filtering
260 plt.subplot(2, 3, 5)
261 plt.imshow(filtered_image_9x9, cmap='gray')
262 plt.title('Image after 9x9 Average Filtering')
263 plt.axis('off')
264
265 # Show the second plot (Original vs Noisy vs Filtered)
266 plt.show()
267
268
269 # Function to add Random noise
270 def add_random_noise(image, noise_factor=0.05):
271     noisy_image = np.copy(image).astype(np.float32)
272
273     # Generate random noise in the range [0, 255]
274     random_noise = np.random.uniform(0, 255, image.shape)
275
276     # Scale the random noise by a factor and add it to the image
277     noisy_image = noisy_image + noise_factor * random_noise
278
279     # Clip the image to stay within valid pixel range [0, 255]
280     noisy_image = np.clip(noisy_image, 0, 255).astype(np.uint8)
281
282     return noisy_image
283
```

In 71, Col 46 Spaces: 4 UTF-8 CR/LF {} Python 3.7.4 64-bit (3.7.4: pyenv) ⚡ Go Live ⚡ AI Code Chat

18°C Sonnig 4:41 PM ENG 10/25/2024

```
File Edit Selection View Go Run Terminal Help cv task 1
code_1.py > ...
240
247 # Filtered image with 3x3 average filtering
248 plt.subplot(2, 3, 3)
249 plt.imshow(filtered_image_3x3, cmap='gray')
250 plt.title('Image after 3x3 Average Filtering')
251 plt.axis('off')
252
253 # Filtered image with 5x5 average filtering
254 plt.subplot(2, 3, 4)
255 plt.imshow(filtered_image_5x5, cmap='gray')
256 plt.title('Image after 5x5 Average Filtering')
257 plt.axis('off')
258
259 # Filtered image with 9x9 average filtering
260 plt.subplot(2, 3, 5)
261 plt.imshow(filtered_image_9x9, cmap='gray')
262 plt.title('Image after 9x9 Average Filtering')
263 plt.axis('off')
264
265 # Show the second plot (Original vs Noisy vs Filtered)
266 plt.show()
267
268
269 # Function to add Random noise
270 def add_random_noise(image, noise_factor=0.05):
271     noisy_image = np.copy(image).astype(np.float32)
272
273     # Generate random noise in the range [0, 255]
274     random_noise = np.random.uniform(0, 255, image.shape)
275
276     # Scale the random noise by a factor and add it to the image
277     noisy_image = noisy_image + noise_factor * random_noise
278
279     # Clip the image to stay within valid pixel range [0, 255]
280     noisy_image = np.clip(noisy_image, 0, 255).astype(np.uint8)
281
282     return noisy_image
283
```

In 71, Col 46 Spaces: 4 UTF-8 CR/LF {} Python 3.7.4 64-bit (3.7.4: pyenv) ⚡ Go Live ⚡ AI Code Chat

18°C Sonnig 4:42 PM ENG 10/25/2024

```
File Edit Selection View Go Run Terminal Help ↵ → ○ cv task 1
Settings code_1.py X
285
286 # Add Random noise to the image
287 noise_factor = 0.4 # Adjust this value to control the amount of noise
288 noisy_image = add_random_noise(image_cv, noise_factor)
289
290 # Apply average filtering with different filter sizes
291 filtered_image_3x3 = cv2.blur(noisy_image, (3, 3))
292 filtered_image_5x5 = cv2.blur(noisy_image, (5, 5))
293 filtered_image_9x9 = cv2.blur(noisy_image, (9, 9))
294
295 # Plot 1: Original vs Noisy image
296 plt.figure(figsize=(10, 5))
297
298 # Original grayscale image
299 plt.subplot(1, 2, 1)
300 plt.imshow(image_cv, cmap='gray')
301 plt.title('Original Grayscale Image')
302 plt.axis('off')
303
304 # Noisy image with Random noise
305 plt.subplot(1, 2, 2)
306 plt.imshow(noisy_image, cmap='gray')
307 plt.title('Image with Random Noise')
308 plt.axis('off')
309
310 # Show the first plot (Original vs Noisy)
311 plt.show()
312
313 # Plot 2: Original vs Noisy vs Filtered images
314 plt.figure(figsize=(20, 10))
315
316 # Original grayscale image
317 plt.subplot(2, 3, 1)
318 plt.imshow(image_cv, cmap='gray')
319 plt.title('Original Grayscale Image')
320 plt.axis('off')
321
322 # Noisy image with Random noise
323 plt.subplot(2, 3, 2)
324 plt.imshow(noisy_image, cmap='gray')
325 plt.title('Image with Random Noise')
326 plt.axis('off')
327
328 # Filtered image with 3x3 average filtering
329 plt.subplot(2, 3, 3)
330 plt.imshow(filtered_image_3x3, cmap='gray')
331 plt.title('Image after 3x3 Average Filtering')
332 plt.axis('off')
333
334 # Filtered image with 5x5 average filtering
335 plt.subplot(2, 3, 4)
336 plt.imshow(filtered_image_5x5, cmap='gray')
337 plt.title('Image after 5x5 Average Filtering')
338 plt.axis('off')
339
340 # Filtered image with 9x9 average filtering
341 plt.subplot(2, 3, 5)
342 plt.imshow(filtered_image_9x9, cmap='gray')
343 plt.title('Image after 9x9 Average Filtering')
344 plt.axis('off')
345
346 # Show the second plot (Original vs Noisy vs Filtered)
347 plt.show()
348
```

In 71, Col 46 Spaces: 4 UTF-8 CRLF {} Python 3.7.4 64-bit (3.7.4: pyenv) ⚡ Go Live ⚡ AI Code Chat

18°C Sonnig 4:42 PM ENG 10/25/2024

```
File Edit Selection View Go Run Terminal Help ↵ → ○ cv task 1
Settings code_1.py X
314 plt.figure(figsize=(20, 10))
315
316 # Original grayscale image
317 plt.subplot(2, 3, 1)
318 plt.imshow(image_cv, cmap='gray')
319 plt.title('Original Grayscale Image')
320 plt.axis('off')
321
322 # Noisy image with Random noise
323 plt.subplot(2, 3, 2)
324 plt.imshow(noisy_image, cmap='gray')
325 plt.title('Image with Random Noise')
326 plt.axis('off')
327
328 # Filtered image with 3x3 average filtering
329 plt.subplot(2, 3, 3)
330 plt.imshow(filtered_image_3x3, cmap='gray')
331 plt.title('Image after 3x3 Average Filtering')
332 plt.axis('off')
333
334 # Filtered image with 5x5 average filtering
335 plt.subplot(2, 3, 4)
336 plt.imshow(filtered_image_5x5, cmap='gray')
337 plt.title('Image after 5x5 Average Filtering')
338 plt.axis('off')
339
340 # Filtered image with 9x9 average filtering
341 plt.subplot(2, 3, 5)
342 plt.imshow(filtered_image_9x9, cmap='gray')
343 plt.title('Image after 9x9 Average Filtering')
344 plt.axis('off')
345
346 # Show the second plot (Original vs Noisy vs Filtered)
347 plt.show()
348
```

In 71, Col 46 Spaces: 4 UTF-8 CRLF {} Python 3.7.4 64-bit (3.7.4: pyenv) ⚡ Go Live ⚡ AI Code Chat

Nahe beim Rekord 4:42 PM ENG 10/25/2024