Pencil Sketch Image Processing

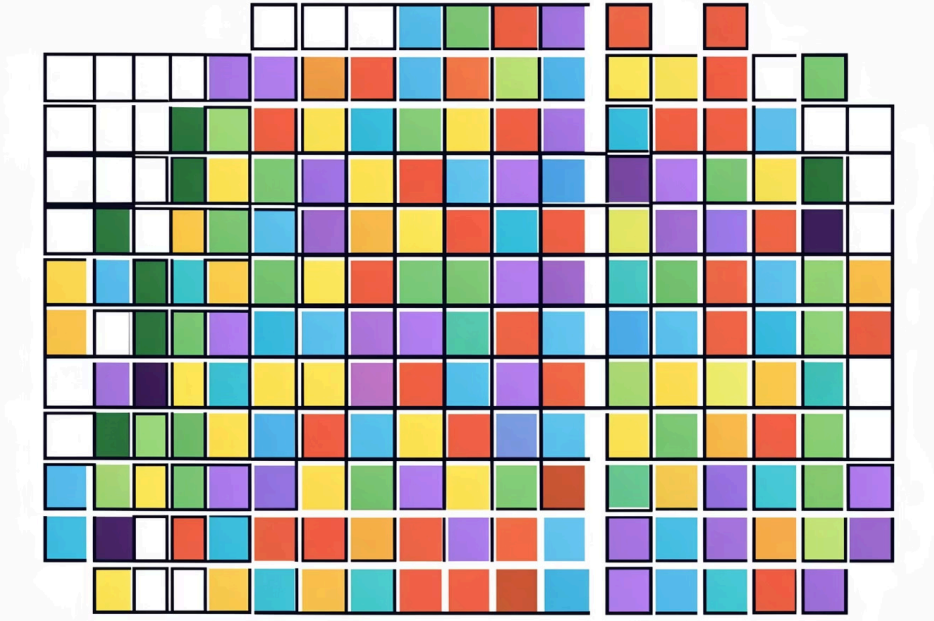
A Digital Image Processing Semester Project

Student Names: Ayesha & Iman

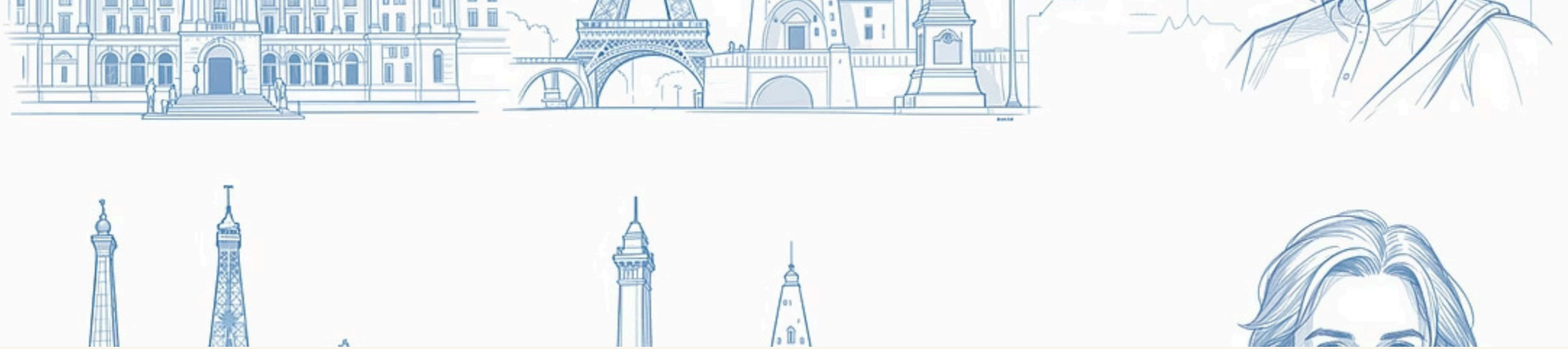
Roll Numbers: 100042 & 100049

Institution: Govt Graduate College Burewala

Digital Image Processing Overview



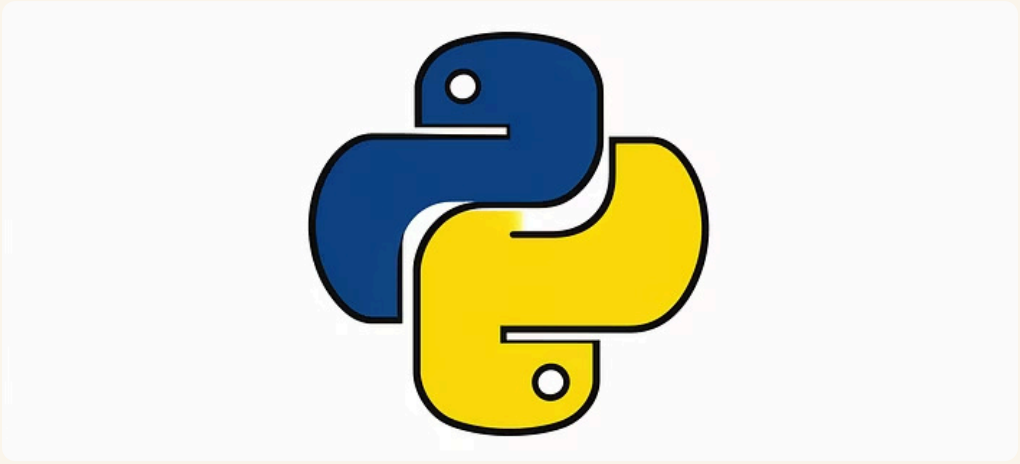
Digital Image Processing uses computers to improve images and extract useful information, and is widely used in medical, security, and artistic applications.

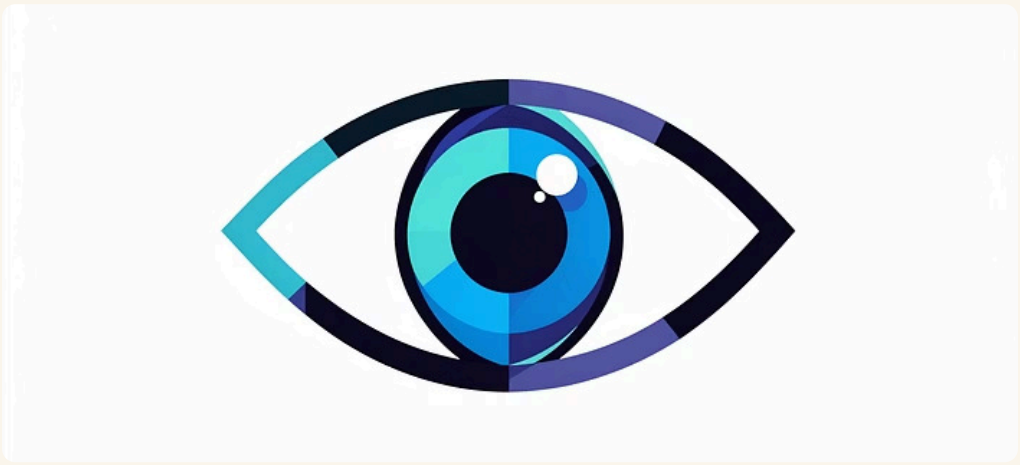


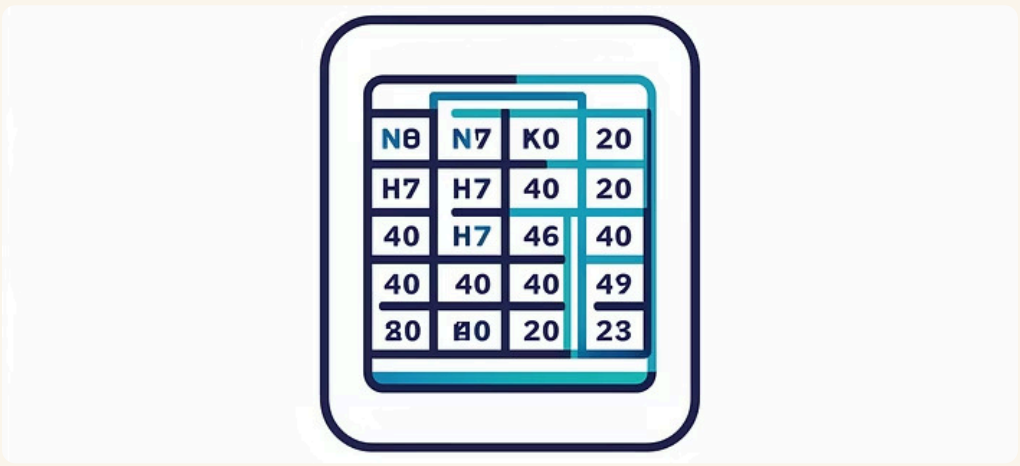
Project Objectives

1	2	3
Transform Images	**Apply Algorithms**	**Demonstrate Skills**
Convert colorful images into realistic pencil sketches using computer vision techniques	Implement edge detection and image filtering algorithms to extract structural features	This project demonstrates practical skills in digital image processing using Python.

Tools and Technologies







1

Python 3.x

Main programming language with lots of libraries for image processing.

2

OpenCV

Library for image processing and computer vision tasks.

3

NumPy

Handles arrays and math operations efficiently.

4

Matplotlib

Displays images and plots results.

5

Jupyter Notebook

Interactive environment for coding and visualizing results

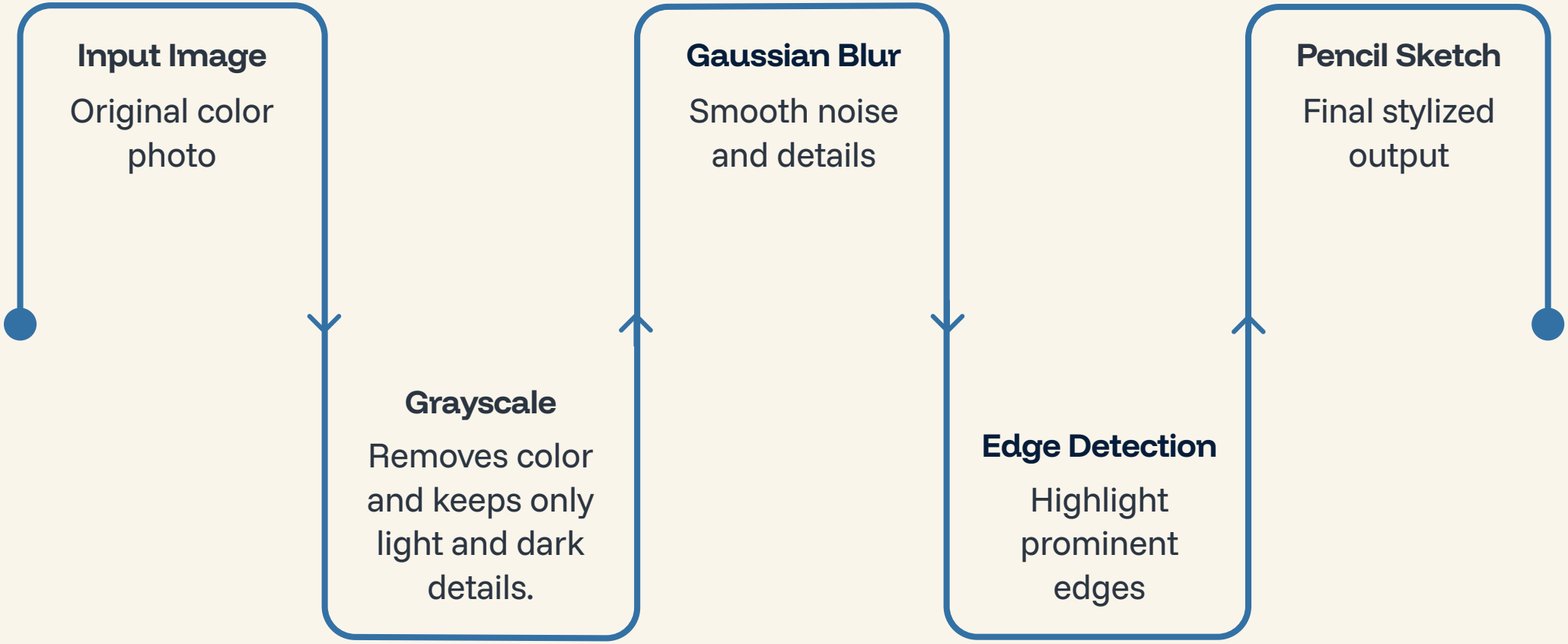
Dataset Description

The dataset has **color photos** of cats.

The images are chosen to **show how the algorithm works on different types of pictures and lighting**.

Format: JPEG and PNG (RGB color).

Methodology Overview



This project converts a normal image into a pencil sketch by following **step-by-step image processing techniques**, mentioned above.

Grayscale Conversion

The first step converts the color image into **grayscale** by removing color and keeping only **light and dark (brightness) details**.

Code :

```
// gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

Images aftter grayscale coversion

Gaussian Blur Application

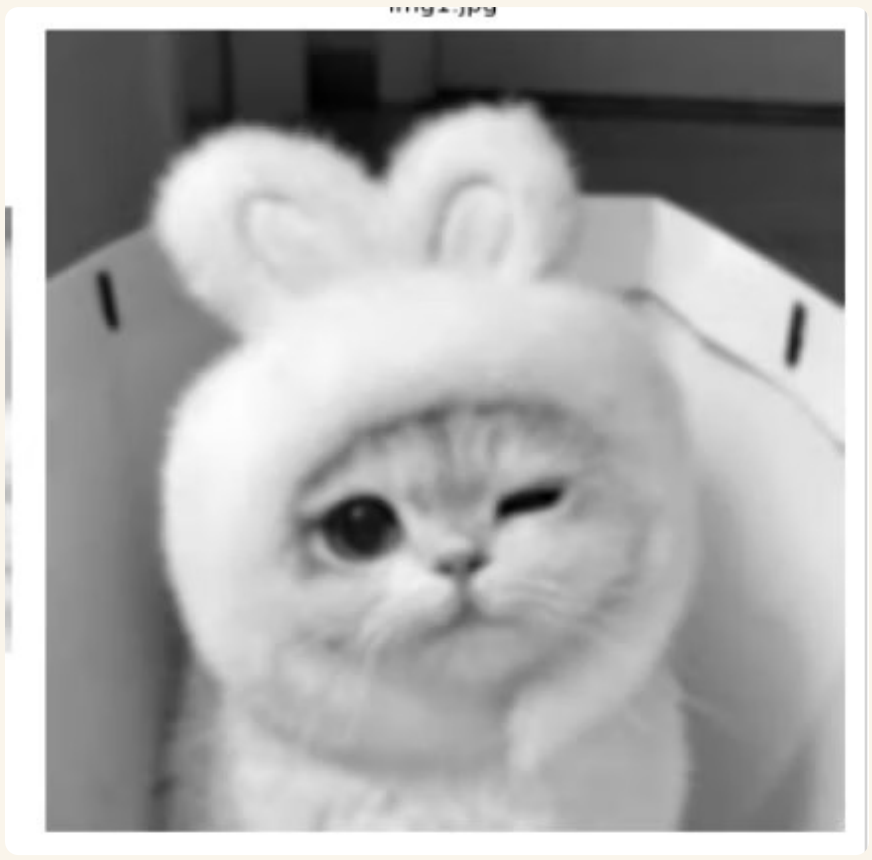
Gaussian blur is used to **smooth the grayscale image** and **remove noise** before detecting edges.
This step helps avoid **false or unwanted edges** caused by small image details.

What it does:

- Softens the image by reducing sharp noise
- Makes edge detection more accurate

Code:

```
blur = cv2.GaussianBlur(gray, (5, 5), 0)
```



Canny Edge Detection Algorithm

Canny edge detection is used to **find clear outlines** in the image, which later form the pencil sketch.

- **Gradient calculation:** Finds areas where brightness changes sharply (possible edges).
- **Non-maximum suppression:** Makes edges **thin and sharp** by removing extra pixels.
- **Double threshold:** Separates **strong edges**, **weak edges**, and **non-edges**.
- **Edge tracking:** Keeps only real edges by connecting weak edges to strong ones.

Code:

```
edges = cv2.Canny(blur, 60, 100)
```

Results and Conclusion

Project Outcomes

This project successfully converts **color images into pencil sketches** while keeping important shapes and edges clear.

The results look **realistic and artistic** for different types of images.

Key Achievements

- Clear and accurate **edge detection**
- **Sketch-like appearance** similar to hand drawing
- **Fast processing** using OpenCV
- **Well-structured code** that is easy to reuse

Future Enhancements

- Add **color pencil sketch** effects
- Improve sketches using **texture effects**
- Apply the method to **real-time video**
- Build a **web-based interactive application**

Code:

```
sketch = cv2.bitwise_not(edges)
```

Presentation Summary

- Converts color images into **pencil sketches** using Python & OpenCV
- Steps: **Grayscale → Gaussian Blur → Canny Edge Detection → Edge Inversion**
- Works on **different types of images**, producing **clear and artistic sketches**
- Can be extended to **color sketches, video processing, or web apps**