**IBM CLOUD DEVELOPMENT PROJECT**

**A blue and white striped logo

Description automatically generated**

**COLLEGE CODE: 4224**

**UNIVERSITY COLLEGE OF ENGINEERING TINDIVANAM**

**MELPAKKAM 604001.**

**IMAGE RECOGNITION WITH IBM CLOUD VISUAL RECOGNITION**

**PROJECT TITLE FOR IMAGE RECOGNITION IS TO**

**IMAGE EDGE DETECTION**

**SUBMITTED BY**

**G M PRANAV**

**B SHAHID AFRIDI**

**G DEVA KUMAR**

**S SUDHAGARAN**

**R MANIGANDAN**

**INTRODUCTION:**

Image edge detection is a fundamental technique in computer vision, and its application in cloud development has opened new horizons for a wide range of industries. By identifying and highlighting the edges of objects within an image, this process allows for enhanced feature extraction, object recognition, and image segmentation. In the context of cloud application development, integrating edge detection algorithms into cloud-based systems provides scalability, accessibility, and real-time processing capabilities. This introduction sets the stage for exploring how image edge detection can empower cloud applications by improving image analysis, security, and automation, among other use cases.

**ABSTRACT:**

Image recognition, a vital field in computer vision, involves the automated identification and classification of objects, scenes, or patterns within digital images or video frames. It employs complex algorithms and deep learning models to extract meaningful features and make sense of visual data. This abstract conceptually outlines the fundamental principles, methodologies, and applications of image recognition, highlighting its significance in various domains like healthcare, autonomous vehicles, security, and more. It also underscores the ongoing advancements and challenges in this evolving field.

|  |  |
| --- | --- |
| **CHAPTER** | **CONTENT** |
| 1 | INTRODUCTION OF IMAGE RECOGNITION |
| 2 | Applications of Image Recognition  Usage of Image Edge Detection  Edge Detection  Submission  Documentation  Code  Model  Result  Program  Output |
| 3 | CONCLUSION |

**INRODUCTION:**

Image recognition, a subset of computer vision, is a fascinating and rapidly evolving field that empowers computers to interpret and understand visual data. In this context, Python, with its rich ecosystem of libraries and tools, plays a pivotal role in developing robust image recognition systems.

Image recognition, often referred to as image classification, involves teaching computers to recognize and categorize objects, scenes, or patterns within digital images or video frames. It enables machines to replicate the human ability to understand visual information, opening doors to a multitude of applications across various industries.

**APPLICATIONS OF IMAGE RECOGNITION:**

The applications of image recognition using Python are diverse and continue to expand.

They include:

Object Detection:

Identifying and locating objects within images or video streams.

Facial Recognition:

Verifying or identifying individuals based on facial features.

Medical Imaging:

Assisting in the diagnosis and analysis of medical images like X-rays and MRIs.

Autonomous Vehicles:

Enabling vehicles to detect and react to their surroundings.

Retail and E-commerce:

Enhancing customer experiences through product recognition.

Security and Surveillance:

Monitoring and analysing security camera footage.

Image recognition using Python is an exciting and transformative field that harnesses the power of deep learning and Python's versatility to enable machines to understand and interact with visual information, leading to numerous innovative applications across industries. This introduction serves as a gateway to explore the vast possibilities within this dynamic realm.

**USAGE OF IMAGE EDGE DETECTION:**

Image edge detection is a fundamental technique in computer vision and image processing used to identify boundaries within an image. Here are some common applications and uses of image edge detection:

1. Object Detection:

Edge detection can help identify objects within an image by highlighting their contours or edges. This is often a crucial step in more complex object detection algorithms.

2. Image Segmentation:

Edge detection is used to partition an image into regions or segments based on the abrupt changes in intensity or color. This is useful for tasks like medical image analysis and scene understanding.

3. Feature Extraction:

Edges represent significant features in an image. They can be used as features for further analysis, such as character recognition in OCR

(Optical Character Recognition) systems.

4. Robotics and Autonomous Navigation:

Robots and autonomous vehicles use edge detection to perceive their

environment and make decisions based on the detected edges, like identifying lanes on a road.

5. Image Enhancement:

Edge detection can enhance the visual appearance of an image by highlighting details and boundaries. This is useful in applications like image sharpening.

6. Medical Imaging:

Edge detection helps in identifying boundaries of organs, tumors, and other structures in medical images like X-rays and MRIs.

7. Quality Control:

In manufacturing, edge detection can be used to inspect products for defects by analyzing the edges and shapes of components.

8. Video Compression:

Edge information is often used in video compression techniques to reduce data redundancy and improve compression efficiency.

9. Artistic Filters:

Edge detection is used in image and video processing software to create artistic effects, like edge-enhancing filters that make images look like sketches.

10. Security and Surveillance:

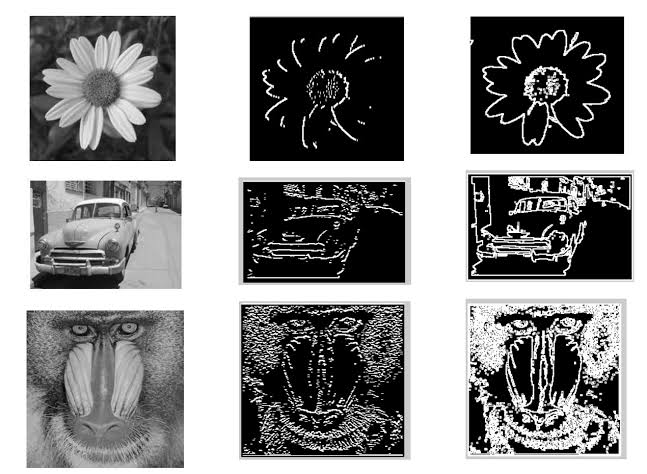
Edge detection can be used in surveillance systems to detect motion by identifying changes in object outlines.

11. Photography:

Photographers sometimes use edge detection to create unique effects or to emphasize certain aspects of an image.

Different edge detection algorithms, such as the Canny edge detector, Sobel operator, and the Laplacian of Gaussian (LoG), are employed

based on the specific requirements of the application. These algorithms work by identifying areas of rapid intensity change in an image, which often correspond to edges or boundaries between objects or regions.



**EDGE DETECTION:**

Edge detection helps segment images to allow for data extraction. An edge in an image is essentially a discontinuity (or a sharp change) in the pixel intensity values of an image. You must have witnessed edge detection at play in software like Kingsoft WPS or your own smartphone scanners and, therefore, should be familiar with its significance.

**SUBMISSION:**

To submit an image recognition project:

**DOCUMENTATION:**

Prepare a report detailing your process, including data sources, model architecture, training details, and results.

**CODE:**

Share the code (Python, for instance) used for training and testing the model.

**MODEL:**

Submit the trained model file or details on how to access it.

**RESULT:**

Present the accuracy metrics, validation, and testing results

Share insights, limitations, and potential improvements.

For more specific instructions or a template tailored to a particular platform or competition, please provide more details about the submission requirements.

**PROGRAM:**

import cv2

# Read the original image

img = cv2.imread('vijay.jpeg')

# Display original image

cv2.imshow('Original', img)

cv2.waitKey(0)

# Convert to graycsale

img\_gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

# Blur the image for better edge detection

img\_blur = cv2.GaussianBlur(img\_gray, (3,3), 0)

# Sobel Edge Detection

sobelx = cv2.Sobel(src=img\_blur, ddepth=cv2.CV\_64F, dx=1, dy=0, ksize=5) # Sobel Edge Detection on the X axis

sobely = cv2.Sobel(src=img\_blur, ddepth=cv2.CV\_64F, dx=0, dy=1,

ksize=5) # Sobel Edge Detection on the Y axis

sobelxy = cv2.Sobel(src=img\_blur, ddepth=cv2.CV\_64F, dx=1, dy=1, ksize=5) # Combined X and Y Sobel Edge Detection

# Display Sobel Edge Detection Images

cv2.imshow('Sobel X', sobelx)

cv2.waitKey(0)

cv2.imshow('Sobel Y', sobely)

cv2.waitKey(0)

cv2.imshow('Sobel X Y using Sobel() function', sobelxy)

cv2.waitKey(0)

# Canny Edge Detection

edges = cv2.Canny(image=img\_blur, threshold1=100, threshold2=200) # Canny Edge Detection

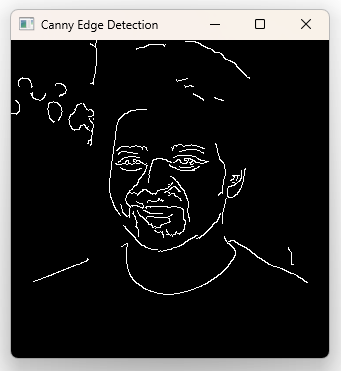
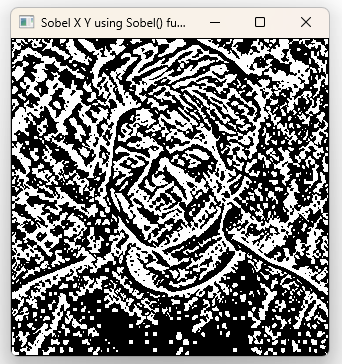
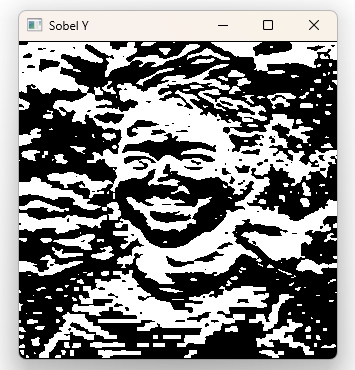
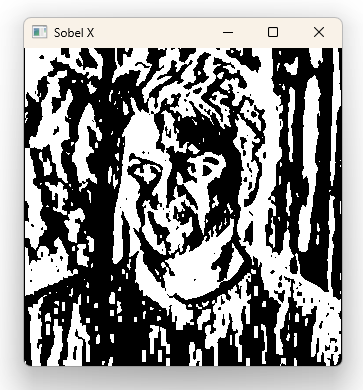
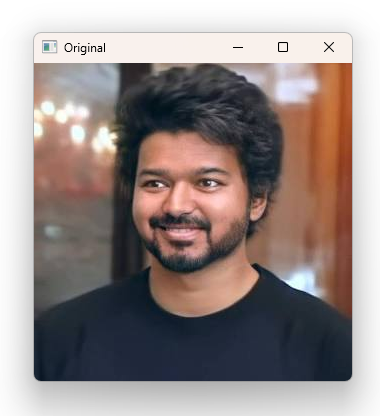
# Display Canny Edge Detection Image

cv2.imshow('Canny Edge Detection', edges)

cv2.waitKey(0)

cv2.destroyAllWindows()

**OUTPUT:**

****

**CONCLUSION:**

Image edge detection is a fundamental technique in computer vision and image processing. It plays a crucial role in identifying object boundaries and extracting important features from images. Various edge detection algorithms, such as the Sobel, Canny, and Prewitt operators, offer different trade-offs in terms of accuracy and computational complexity. The choice of algorithm depends on the specific application and

requirements. Overall, edge detection is a key step in many image

analysis tasks, including object recognition, image segmentation, and more, making it an essential tool in the field of computer vision.