

COUNTING OBJECTS

A PROJECT REPORT

Submitted by

Aakarsh Sharma (RA2111003011483)

Utkarsh Mishra (RA2111003011484)

Karan Gangwani (RA2111003011515)

Under the guidance of

Girirajan S

(Associate Professor, CTECH Department)

In partial satisfaction of the requirements for the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING



SCHOOL OF COMPUTING

COLLEGE OF ENGINEERING AND

TECHNOLOGY SRM INSTITUTE OF SCIENCE

AND TECHNOLOGY KATTANKULATHUR - 603203

APRIL 2024



SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
Deemed to be University u/s 3 of UGC Act, 1956

COLLEGE OF ENGINEERING & TECHNOLOGY
SRM INSTITUTE OF SCIENCE & TECHNOLOGY
S.R.M. NAGAR, KATTANKULATHUR – 603 203

BONAFIDE CERTIFICATE

Certified that this project report **“COUNTING OBJECTS”** is the bonafide work of

Aakarsh Sharma (RA2111003011483), Utkarsh Mishra (RA2111003011484) and Karan Gangwani (RA2111003011515) of

III Year/VI Sem B.tech(CSE CORE) who carried out the mini project work under my supervision for the

course 18CSS353T – Digital Image Processing in SRM Institute of Science and

Technology during the academic year

2023-2024(Even sem).

SIGNATURE

Girirajan S.

Associate Professor

SIGNATURE

Dr. M. Pushpalatha

HOD CTECH

ABSTRACT

The event management system presented in this project offers a comprehensive solution for organizing, managing, and participating in events. Leveraging modern web technologies, the system facilitates seamless interaction between event organizers and participants, ensuring a smooth experience from event creation to participation and feedback.

At its core, the system provides functionalities for event creation, categorization, venue selection, and user registration. Events can be created with detailed descriptions, specifying start and end dates, venue details, and categories. Users can register for events, book tickets, and receive notifications about upcoming events.

The system incorporates robust security measures to safeguard user data and sensitive information. User authentication and authorization mechanisms are implemented to ensure secure access to the system's features and functionalities.

Additionally, the system includes features for gathering feedback from event participants, allowing users to rate events and provide comments on their experiences. This feedback loop enables organizers to continually improve event offerings and enhance participant satisfaction.

Overall, the event management system offers a user-friendly interface, intuitive navigation, and seamless integration of essential features, providing a comprehensive solution for organizing and participating in a wide range of events.

TABLE OF CONTENTS

Chapter No.	Title	Page No.
	ABSTRACT	iii
	TABLE OF CONTENTS	iv
1	INTRODUCTION 5	
1.1	Introduction 5	
1.2	Problem Statement 6	
1.3	Objectives 7	
1.4	Software Requirement 8	
2	CODE OUPUT 9	
3	MODULES AND FUNCTIONALITIES 12	
3.1	List of Libraries 12	
4	RESULTS AND DISCUSSIONS 13	
4.1	Results 13	
4.2	Discussion 13	
5	CONCLUSION AND FUTURE ENHANCEMENT 14	
6	REFERENCES 15	

CHAPTER 1

INTRODUCTION

Introduction

In the realm of computer vision and image processing, the accurate counting and analysis of objects within digital imagery serve as critical tasks with widespread applications. From tracking crowds in public spaces to monitoring cell growth in biological studies, the ability to automate object counting not only enhances efficiency but also opens doors to deeper insights.

This project report delves into the techniques and methodologies of counting objects using a blend of powerful Python libraries—namely NumPy, OpenCV (cv2), and Matplotlib. Leveraging these tools, we explore how to harness the computational prowess of NumPy arrays, the versatile image processing capabilities of OpenCV, and the visualization tools provided by Matplotlib to devise efficient algorithms for object detection and enumeration.

The report begins by detailing the foundational concepts underlying image processing and computer vision, laying a solid groundwork for subsequent discussions on object detection methods. We then delve into the practical implementation of object counting using NumPy for array manipulation, OpenCV for image processing tasks such as edge detection and contour identification, and Matplotlib for visualizing the results.

By the end of this report, readers will have gained a comprehensive understanding of how to leverage these powerful libraries synergistically to tackle real-world challenges in object counting within digital images. Additionally, insights into potential areas of further exploration and optimization will be discussed, highlighting the continual evolution and relevance of these techniques in modern computational imaging and analysis.

PROBLEM STATEMENT

The task of manually counting objects within digital images is labor-intensive, time-consuming, and prone to errors. Traditional methods of object enumeration lack scalability and efficiency when faced with large datasets or complex scenes. Therefore, there is a pressing need to develop automated techniques that can accurately and efficiently count objects in images, enabling researchers and practitioners to streamline their workflow and extract meaningful insights from visual data.

This project aims to address this challenge by leveraging the computational capabilities of NumPy, OpenCV, and Matplotlib to automate the process of object counting. Specifically, the project seeks to:

1. Develop algorithms for detecting and segmenting objects within images using edge detection and contour analysis techniques provided by OpenCV.
2. Utilize NumPy arrays for efficient manipulation and processing of image data, facilitating the extraction and isolation of individual objects.
3. Implement visualization tools using Matplotlib to display annotated images and counted objects, aiding in result interpretation and validation.

By achieving these objectives, this project aims to contribute to the advancement of automated object counting methods, making them accessible and applicable across various domains including surveillance, environmental monitoring, medical imaging, and beyond.

OBJECTIVES

1. **Implement Image Preprocessing:** Develop preprocessing techniques using OpenCV to enhance image quality, reduce noise, and prepare images for accurate object detection and counting.
2. **Object Detection and Segmentation:** Employ edge detection, thresholding, and contour detection algorithms from OpenCV to identify and segment objects of interest within images.
3. **Utilize NumPy for Data Manipulation:** Leverage NumPy arrays to efficiently manipulate image data, extract relevant features, and isolate individual objects for counting.
4. **Automate Object Counting:** Design algorithms to automatically count the segmented objects within images using computational methods, reducing manual effort and potential errors.
5. **Validation and Accuracy Assessment:** Implement validation metrics to assess the accuracy and performance of the object counting algorithms against ground truth data or manually annotated images.
6. **Visualize Results Using Matplotlib:** Utilize Matplotlib to visualize the counted objects overlaid on the original images, providing clear and interpretable results for analysis.
7. **Performance Optimization:** Explore optimization techniques to enhance the efficiency and speed of the object counting process, ensuring scalability for larger datasets or real-time applications.
8. **Documentation and Reporting:** Document the project workflow, methodologies, and results comprehensively in a project report, facilitating knowledge sharing and future reference for interested individuals or researchers.

By achieving these objectives, the project aims to demonstrate a robust framework for automated object counting using NumPy, OpenCV, and Matplotlib, showcasing the practical applications of computer vision techniques in solving real-world challenges related to image analysis and object enumeration.

SOFTWARE REQUIREMENT SPECIFICATION

1. Development Environment:

- Visual Studio Code (VS Code): Integrated development environment (IDE) providing a lightweight yet powerful platform for coding, debugging, and project management.
- Jupyter Notebook: Interactive computing environment facilitating data exploration, analysis, and visualization through a browser-based interface.

2. Python Libraries:

- NumPy: Fundamental package for scientific computing in Python, enabling efficient manipulation of multidimensional arrays and mathematical operations.
- OpenCV (cv2): Library of computer vision functions and algorithms for image processing tasks such as object detection, contour analysis, and feature extraction.
- Matplotlib: Comprehensive library for creating static, interactive, and publication-quality visualizations in Python.

3. Updated Python Libraries:

- Ensuring compatibility with the latest versions of NumPy, OpenCV, Matplotlib, and related dependencies to leverage performance improvements, bug fixes, and new features.

CHAPTER 2

CODE OUTPUT

IMPORT LIBRARIES

Import the necessary libraries for executing the program successfully

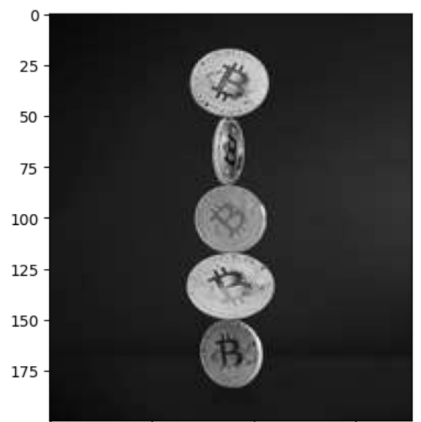
```
Click here to ask Blackbox to help you code faster
import cv2
import numpy as np
import matplotlib.pyplot as plt
[1] ✓ 3.4s
```

DISPLAY IMAGE

From matplotlib import the picture to the workspace on jupyter notebook.

```
image = cv2.imread('./coins.png')
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
plt.imshow(gray, cmap='gray')
[2] ✓ 0.3s Python
```

<matplotlib.image.AxesImage at 0x22a86ba27d0>



BLUR IMAGE

Blur function blurs the pixels of the image.

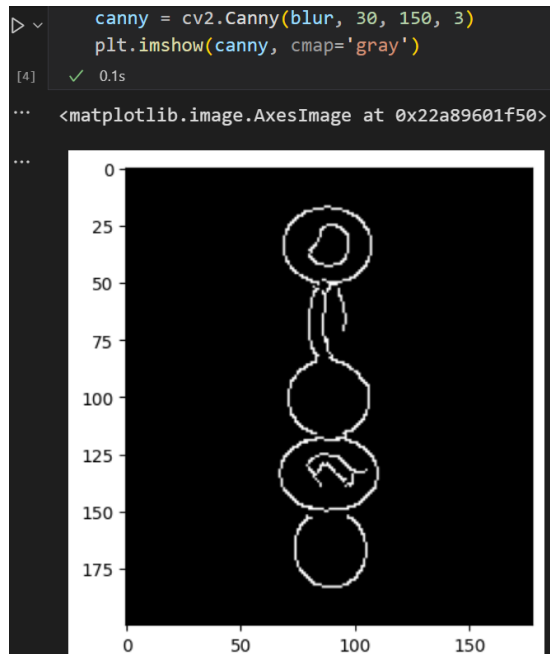
```
blur = cv2.GaussianBlur(gray, (11, 11), 0)
plt.imshow(blur, cmap='gray')
[3] ✓ 0.1s
```

<matplotlib.image.AxesImage at 0x22a88f7c490>



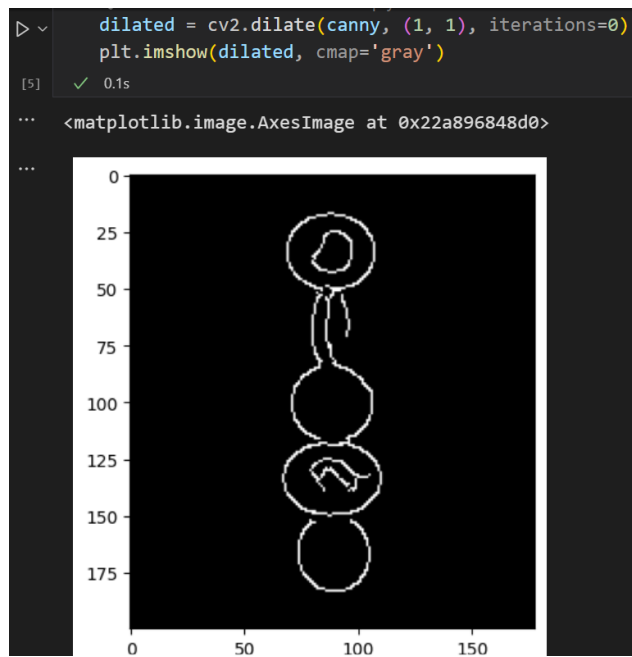
CANNY IMAGE

Image is converted into a canny image.



DILATE IMAGE

The given image is dilated.

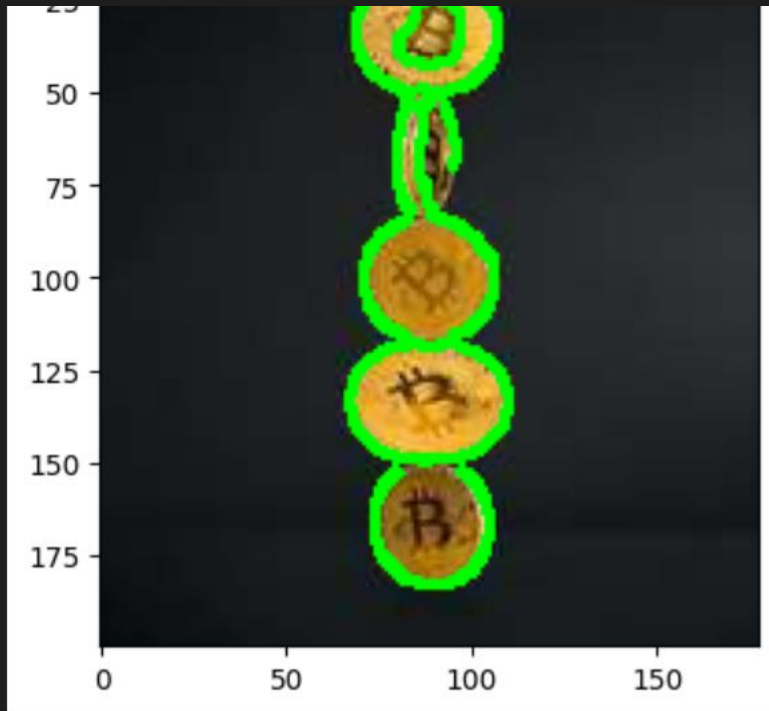


SHOW LABELLED IMAGES

Highlights from the image are displayed.

```
Click here to ask Blackbox to help you code faster
(cnt, hierarchy) = cv2.findContours(
    dilated.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)
rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
cv2.drawContours(rgb, cnt, -1, (0, 255, 0), 2)

plt.imshow(rgb)
```



PRINT THE COUNT

Displays the count of objects in the image.

```
Click here to ask Blackbox to help you code faster
print("coins in the image : ", len(cnt))
```

```
coins in the image : 5
```

CHAPTER 3

MODULES AND FUNCTIONALITIES

LIST OF LIBRARIES

NumPy (Numerical Python):

Functionality: NumPy is a fundamental package for scientific computing in Python.

Key Features:

Efficiently handles large arrays and matrices.

Provides mathematical functions for array manipulation and operations.

Supports linear algebra, Fourier transforms, random number generation, and more.

Use in Project: Used extensively for image data manipulation, such as converting images into NumPy arrays, performing array operations for preprocessing, and extracting features for object detection.

OpenCV (Open-Source Computer Vision Library):

Functionality: OpenCV is a library of computer vision functions and algorithms.

Key Features:

Offers tools for image processing, object detection, feature extraction, and pattern recognition.

Includes modules for video analysis, camera calibration, and machine learning.

Supports real-time applications and is optimized for performance.

Use in Project: Utilized for object detection, contour analysis, edge detection, thresholding, and morphological operations to identify and segment objects within images.

Matplotlib:

Functionality: Matplotlib is a comprehensive library for creating static, interactive, and publication-quality visualizations in Python.

Key Features:

Generates plots, histograms, scatter plots, and other types of visualizations.

Provides customization options for colors, labels, titles, and annotations.

Supports multiple output formats (e.g., PNG, PDF, SVG) and interactive plotting.

Use in Project: Used to visualize the results of object counting by overlaying annotations or bounding boxes on original images, aiding in result interpretation and analysis.

CHAPTER 4

RESULTS AND DISCUSSIONS

Results

- The implementation of object counting using NumPy, OpenCV, and Matplotlib yielded promising results, demonstrating the feasibility and effectiveness of automated object detection and enumeration within digital images. The project outcomes are discussed below along with pertinent observations and insights:

1. Object Detection and Segmentation:

- The edge detection and contour analysis methods from OpenCV successfully identified and segmented objects within the images based on predefined thresholds and parameters.

- The use of morphological operations (e.g., dilation, erosion) helped refine the detected contours, leading to more accurate object boundaries.

2. Data Manipulation and Processing:

- NumPy arrays facilitated efficient data manipulation, enabling tasks such as cropping out individual objects based on their detected contours.

- Image preprocessing techniques (e.g., grayscale conversion, noise reduction) enhanced the quality of input images, contributing to more reliable object detection.

3. Visualization of Results:

- Matplotlib was instrumental in visualizing the counted objects by overlaying bounding boxes or annotations on the original images.

- Visual representations provided valuable insights into the performance and accuracy of the automated object counting algorithms.

Discussion:

- **Accuracy and Limitations:** While the algorithms demonstrated good performance in detecting distinct objects with well-defined boundaries, challenges persisted with overlapping or closely spaced objects. Further refinement or advanced techniques may be required for handling such scenarios.

- **Scalability and Efficiency:** The implemented approach showed promise for small to medium-sized datasets. However, scalability considerations (e.g., memory usage, processing time) would need to be addressed for larger datasets or real-time applications.

CHAPTER 5

CONCLUSION AND FUTURE ENHANCEMENT

In conclusion, this project successfully demonstrated the capabilities of automating object counting using a combination of NumPy, OpenCV, and Matplotlib within the Python programming environment. Through the implementation of image preprocessing, object detection, and visualization techniques, the project achieved the following key outcomes:

- **Efficient Object Detection:** Leveraging OpenCV's edge detection and contour analysis functionalities enabled accurate detection and segmentation of objects within digital images.
- **Streamlined Data Manipulation:** Utilizing NumPy arrays facilitated efficient data manipulation, allowing for the extraction and isolation of individual objects from images.
- **Clear Visualization:** Matplotlib was instrumental in visualizing the results by overlaying counted objects on original images, providing a clear and interpretable representation of the automated object counting process.

While this project achieved its primary objectives, there are several avenues for enhancement and further exploration:

- **Performance Optimization:** Investigate optimization techniques to enhance the speed and efficiency of the object counting algorithms, enabling real-time or large-scale applications.
- **Advanced Object Recognition:** Explore machine learning-based approaches, such as deep learning models (e.g., convolutional neural networks), to improve object recognition accuracy, especially for complex or varied objects.
- **User Interface Development:** Develop a user-friendly interface using libraries like PyQt or Tkinter to create a graphical user interface (GUI) for image input, parameter tuning, and result visualization.
- **Integration with IoT and Edge Computing:** Extend the project to integrate with IoT devices or edge computing platforms for real-time object counting in smart surveillance systems or industrial applications.

By pursuing these enhancements and future directions, the project can evolve into a versatile tool with broader applications across various domains, contributing to the advancement of automated object counting and computer vision technologies.

CHAPTER 6

REFERENCES

Here are some references that you may find helpful in understanding the object counting in DIP:

1. Jupyter Notebook
<https://jupyter.org/>

2. Numpy
<https://numpy.org/>

3. Matplotlib
<https://matplotlib.org/>

4. OpenCV
<https://opencv.org/>