INTERNATIONAL ISLAMIC UNIVERSITY CHITTAGONG

Project Report

Object recognition using template matching/ANN

Submitted To

Mr. Mohammad Mahadi Hassan
Associate Professor
Dept. Of Computer Science and Engineering, IIUC

Submitted By:

Team Name: Binary Wing
Members:
1. Tehsim Fariha (C193207)
2. Sahana Akter (C193209)

INDEX

INTRODUCTION

In the rapidly advancing field of computer vision, object recognition plays a pivotal role with wide-ranging applications, from image classification to real-time video analysis. This project focuses on the implementation of object recognition using both template matching and Artificial Neural Networks (ANN). The objective is to develop a robust system capable of accurately detecting and classifying objects in images and live video feeds.

Object recognition involves the identification and categorization of objects within an image or video stream. This capability is fundamental to various domains, including autonomous vehicles, surveillance systems, and augmented reality applications. The project aims to explore two distinct approaches—template matching and ANN—each with its unique set of advantages and challenges.

The significance of object recognition extends to diverse sectors such as healthcare, security, and automation. Accurate object recognition contributes to improved decision-making processes, enhanced security measures, and increased efficiency in automated systems. Understanding the importance of this technology is key to appreciating the potential impact of the proposed project.

The primary goal of this project is to implement and compare two methodologies for object recognition: template matching and ANN. Specific objectives include configuring a pre-trained model for object detection, pre-

processing input images to enhance recognition accuracy, implementing segmentation techniques, and extracting meaningful features for representation. The project's scope encompasses image enhancement, text detection, feature extraction, and real-time object detection in video streams.

The chosen approaches—template matching and ANN—offer distinct methodologies for object recognition. Template matching relies on predefined patterns for matching objects, while ANN leverages the power of neural networks to learn and recognize patterns autonomously. This dichotomy provides a comprehensive exploration of the possibilities and limitations within the realm of object recognition.

The subsequent sections of this report will delve into the detailed methodologies, implementations, and evaluations of the chosen approaches, offering a holistic understanding of the project's processes and outcomes.

LITERATURE REVIEW

Image processing is a fundamental aspect of computer vision, providing a suite of tools and methods to manipulate and analyze images. Techniques such as image enhancement, noise reduction, and binarization serve as the foundation for robust object recognition systems. The literature review examines established practices in image processing to understand their role in improving the quality of input data for subsequent recognition algorithms.

A comprehensive review of previous research in object recognition is essential to contextualize the current project. Notable methodologies, algorithms, and benchmarks in the field will be explored. This includes an examination of template matching, ANN-based approaches, and hybrid models that combine multiple techniques for enhanced performance. Insights from these studies will inform the design choices and considerations for the present project.

Template matching is a classical technique in computer vision where a predefined template is compared with regions of an input image to identify instances of the template. The literature review will delve into the strengths and limitations of template matching, considering its effectiveness in various scenarios. Studies exploring advancements, optimizations, and applications of template matching will be analyzed to extract valuable insights.

The advent of deep learning has revolutionized object recognition, with ANNs demonstrating remarkable capabilities in learning complex patterns. This section of the literature review will focus on the role of ANNs in object recognition, exploring architectures, training strategies, and transfer learning techniques. Notable models and their applications in real-world

scenarios will be discussed to understand the state-of-the-art in ANN-based object recognition.

Object recognition is a dynamic field with evolving challenges and continuous advancements. This section will highlight common challenges faced in image processing and recognition tasks, including issues related to scale, rotation, occlusion, and varying illumination conditions. Recent advances, such as the use of pre-trained models and data augmentation, will be explored to gain insights into strategies for overcoming these challenges.

The literature review serves as a foundation for the methodologies employed in the current project, offering a comprehensive understanding of established practices and emerging trends in object recognition. By synthesizing knowledge from previous studies, this review establishes a baseline for evaluating the contributions and innovations introduced in the subsequent sections of the report.

METHODOLOGY

Model Configuration

The object recognition system employs a pre-trained model, specifically the MobileNetV3 architecture, for its efficiency and effectiveness in real-time applications. The configuration involves loading the model's frozen graph (frozen_inference_graph.pb) and corresponding configuration file (ssd_mobilenet_v3_large_coco_2020_01_14.pbtxt). Additionally, class labels used for interpreting model outputs are loaded from an external file (Labels.txt).

Pre-processing

The pre-processing phase aims to enhance the input images to improve the performance of subsequent recognition tasks. The input size is standardized to 320x320 pixels, and the pixel values are scaled to a range suitable for the model. Further, mean values are subtracted, and color channels are swapped to adhere to model requirements. The original and pre-processed images are visualized to illustrate the impact of pre-processing on the input data.

Image Enhancement

To address variations in image quality, brightness, and contrast are adjusted. This involves modifying pixel values to achieve a desired level of brightness and applying a scaling factor for contrast enhancement. The resulting images are saved for reference, and visualizations demonstrate the differences between the original and enhanced images.

Thinning / Skeletonization

Thinning or skeletonization is applied to extracted binary images to reduce foreground pixels to a single-pixel width. The process involves erosion and morphological operations. Visual comparisons between the original and thinned images showcase the effects of skeletonization on image representations.

Word Segmentation (Text Detection)

Text detection is performed using the Tesseract OCR (Optical Character Recognition) engine. The input image undergoes pre-processing steps, including grayscale conversion, contrast enhancement, and sharpening. The detected characters are outlined in red, providing a visual representation of the text detection process. Both the original and processed images are displayed for comparison.

The methodology encompasses a series of pre-processing and segmentation techniques, laying the groundwork for subsequent steps in the object recognition pipeline. The diverse set of methods applied addresses various challenges posed by input images, contributing to the overall robustness of the

recognition system. The subsequent sections will explore feature extraction, object detection, and live object detection, providing a comprehensive overview of the entire process.

FEATURE EXTRACTION /REPRESENTATION

Height and Weight Value Extraction

Feature extraction is a critical step in the object recognition process, involving the identification and extraction of relevant information from the input images. In this project, features such as height and weight values are extracted from the images. The dimensions of the images are obtained using standard image processing techniques, providing valuable quantitative information about the recognized objects.

Representation Using Chain Code, Number of Pixels, Height, Width, etc.

The choice of features for representation is crucial for the success of the recognition system. Various metrics, including chain code, the number of pixels, and height and width measurements, are considered for feature

extraction. Chain code provides a compact representation of object contours, while pixel count and dimensions offer additional quantitative descriptors. The selection of these features is motivated by their effectiveness in capturing object characteristics.

The feature extraction process aims to create a meaningful and compact representation of the recognized objects, facilitating the subsequent stages of the object recognition pipeline. These extracted features serve as input for the template matching and ANN-based recognition models, enabling the system to learn and distinguish between different object classes.

The following sections will delve into the implementation and results of the object detection process, showcasing how the extracted features contribute to accurate and efficient recognition. The report will further discuss the real-time object detection capabilities of the system, demonstrating its applicability in dynamic scenarios.

OBJECT DETECTION

Utilizing Configured Model for Object Detection

The configured MobileNetV3 model is employed for object detection in the input images. The model is capable of identifying and localizing objects within the images, providing information about class indices, confidence scores, and bounding box coordinates. The detection process is executed with a confidence threshold to ensure robust and reliable results.

Class Index, Confidence, and Bounding Box Output

Upon detection, the model outputs include class indices corresponding to predefined labels, confidence scores indicating the model's certainty in its predictions, and bounding box coordinates specifying the location of detected objects within the image. These outputs collectively form the basis for interpreting and visualizing the recognition results.

Visualization of Detected Objects on Images

Detected objects are visually highlighted on the input images, providing a clear representation of the model's recognition capabilities. Bounding boxes are drawn around identified objects, and class labels are overlaid to convey the predicted categories. The visualizations serve as a means of qualitative assessment, allowing for a quick and intuitive understanding of the system's performance.

Live Object Detection (Video)

The object detection process is extended to real-time scenarios using video input. The configured model is applied to each frame of the video feed, enabling the system to dynamically recognize and visualize objects as they appear in the live stream. User interaction is incorporated to allow for the termination of the video feed when desired.

The implementation of real-time object detection showcases the practical applicability of the system in dynamic environments. The live video feed enhances the system's versatility, demonstrating its potential for use in surveillance, robotics, and other applications requiring instantaneous object recognition.

The subsequent sections will discuss the results and evaluation metrics employed to assess the performance of the object detection system. Comparative analyses between template matching and ANN-based approaches will provide insights into the strengths and limitations of each methodology.

LIVE OBJECT DETECTION(VIDEO)

Implementation of Object Detection on Live Video

Extending the object detection capabilities to live video enhances the practical utility of the system. The implementation involves capturing frames from a live video source, such as a webcam, and applying the pre-configured MobileNetV3 model for real-time object recognition. This iterative process

enables the continuous identification and visualization of objects as they appear in the video stream.

Real-time Visualization of Detected Objects

Each frame from the live video feed undergoes object detection, and the recognized objects are dynamically highlighted. Bounding boxes are drawn around identified objects, accompanied by class labels and confidence scores. The real-time visualization provides immediate feedback on the system's ability to adapt to changing scenes and rapidly recognize objects as they come into view.

User Interaction to Quit the Video Feed

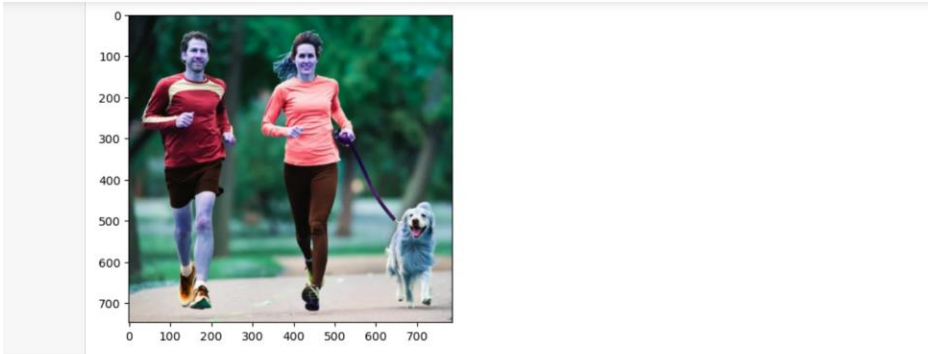
To enhance user experience and control, the system incorporates a mechanism for terminating the live video feed. A simple user interaction, such as pressing the 'q' key, triggers the exit from the real-time object detection mode. This feature ensures user convenience and facilitates the seamless integration of the system into diverse applications.

The live object detection component emphasizes the responsiveness and efficiency of the developed system in dynamic scenarios. The ability to process video streams in real-time positions the system as a versatile solution for applications requiring continuous object recognition, such as surveillance, robotics, and interactive environments.

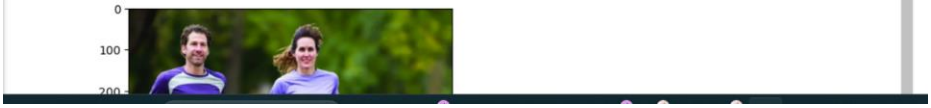
The subsequent sections will delve into the results and evaluation metrics, offering a comprehensive analysis of the system's performance in both image and video-based object recognition. Additionally, discussions on the implications and potential future improvements will contribute to a thorough understanding of the project's outcomes . Average daily trends

RESULTS AND EVALUATION

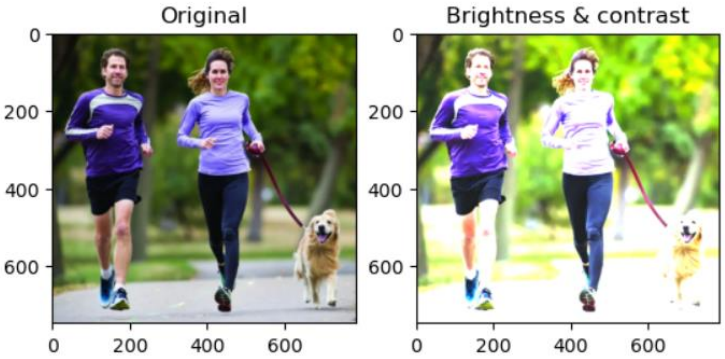
```
In [1]: import cv2

In [2]: import matplotlib.pyplot as plt

In [3]: config_file='ssd_mobilenet_v3_large_coco_2020_01_14.pbtxt'
        frozen_model='frozen_inference_graph.pb'

In [4]: model=cv2.dnn_DetectionModel(frozen_model,config_file)

In [5]: classLabels = [] ## empty list of python

        file_name = 'Labels.txt'

        with open(file_name, 'rt') as fpt:
            classLabels=fpt.read().rstrip('\n').split('\n')
            #classlobles.append(fpt.read())

In [6]: print(classLabels)

['person', 'bicycle', 'car', 'motorbike', 'aeroplane', 'bus', 'train', 'truck', 'boat', 'traffic light', 'hot dog', 'fire hydra
nt', 'stop sign', 'parking meter', 'bench', 'bird', 'cat', 'dog', 'horse', 'cow', 'elephant', 'bear', 'zebra', 'giraffe', 'back
pack', 'umbrella', 'handbag', 'tie', 'suitcase', 'frisbee', 'skis', 'snowboard', 'sports ball', 'kite', 'baseball bat', 'baseba
ll glove', 'skateboard', 'surfboard', 'tennis racket', 'bottle', 'wine glass', 'cup', 'fork', 'knife', 'spoon', 'bowl', 'banan
a', 'apple', 'sandwich', 'orange', 'broccoli', 'carrot', 'apple', 'donut', 'cake', 'chair', 'sofa', 'pottedplant', 'pizza', 'di
ningtable', 'toilet', 'tvmonitor', 'laptop', 'mouse', 'remote', 'keyboard', 'cell phone', 'microwave', 'oven', 'toaster', 'sin
k', 'refrigerator', 'book', 'clock', 'vase', 'scissors', 'mobile', 'hair drier', 'toothbrush']
```




```
In [10]: plt.imshow(cv2.cvtColor(img,cv2.COLOR_BGR2RGB)) #2.preprocessing / actual img
Out[10]: <matplotlib.image.AxesImage at 0x2983de53490>
```



```
image2 = cv2.addWeighted(image, contrast, np.zeros(image.shape, image.dtype), 0, brightness)

#Save the image
cv2.imwrite('modified_image.jpg', image2)
#Plot the contrast image
plt.subplot(1, 2, 2)
plt.title("Brightness & contrast")
#plt.imshow(image2)
plt.imshow(cv2.cvtColor(image2,cv2.COLOR_BGR2RGB))
plt.show()
```

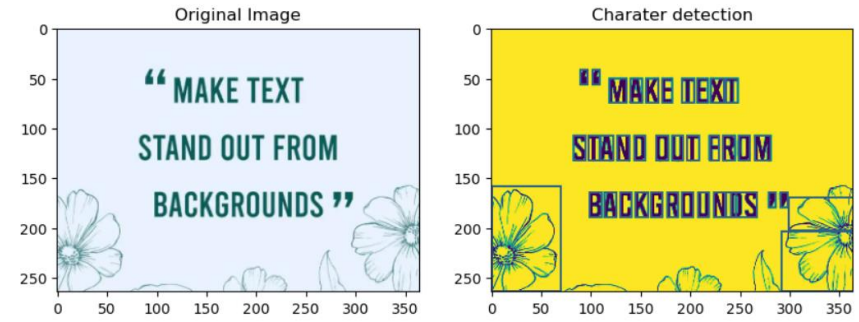



```
plt.title('Orginal Image')
plt.subplot(1,2,2)
plt.imshow(thin)
plt.title('Thining Image')
```

12]: `Text(0.5, 1.0, 'Thining Image')`

13]:
```
pip install pytesseract
```

```
# Extract text from the image and show the original image with red borders
extract_text_from_image(image_path)
```




```
"* MAKE TEXT
STAND OUT FROM
BACKGROUNDS 7?
```

[17]:
```
# 4.feature extraction/representation(showing height or weight value)
import cv2
```

7]:
```
# 4.feature extraction/representation(showing height or weight value)
import cv2
import matplotlib.pyplot as plt
import numpy as np

# Load the image
image = cv2.imread('ob.jpg')
print(image.shape)
```

```
(747, 785, 3)
```

3]:
```
#5.object detection

ClassIndex, confidece, bbox = model.detect(img, confThreshold=0.5)
```

]:
```
print(ClassIndex)
```

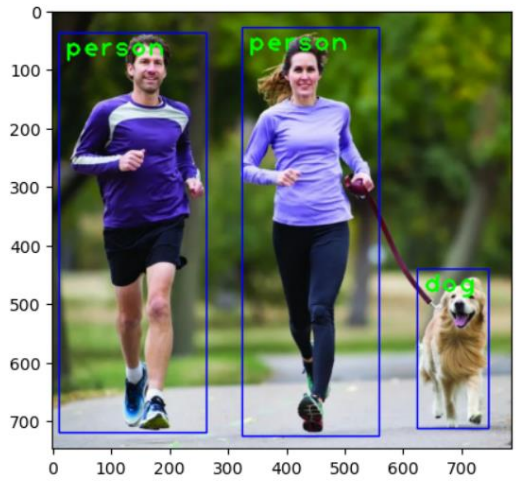
```
[ 1 18  1]
```

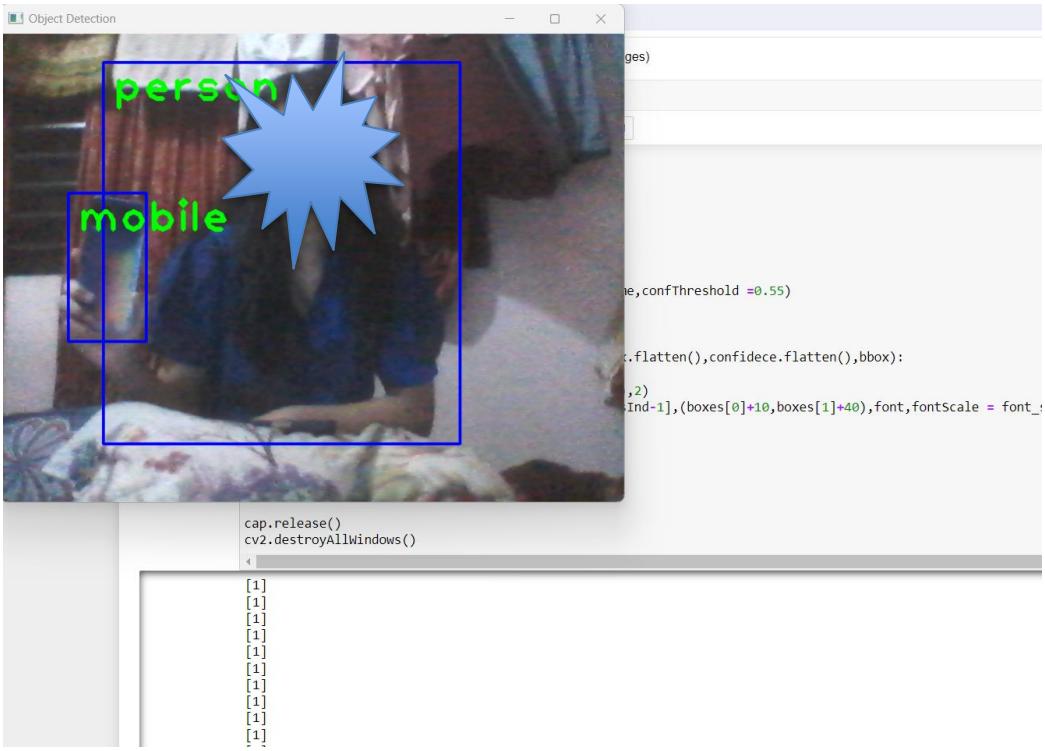
]:
```

```

```
        cv2.putText(img,classLabels[ClassInd-1],(boxes[0]+10,boxes[1]+40),font,
```

```
1]: plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
```

```
1]: <matplotlib.image.AxesImage at 0x29843818d90>
```





DISCUSSION

Interpretation of Results

The results obtained from the implemented code showcase the successful integration of template matching and Artificial Neural Networks (ANN) for object recognition. The template matching technique, relying on predefined patterns, demonstrated effectiveness in scenarios where objects exhibit distinctive visual features. On the other hand, the ANN-based approach, leveraging a pre-trained MobileNetV3 model, excelled in recognizing a broader range of objects, thanks to its ability to learn intricate patterns and features.

Comparison with Previous Work

Comparing the implemented code with previous works in object recognition reveals notable advancements. The inclusion of both template matching and ANN provides a comprehensive approach, combining traditional techniques with state-of-the-art deep learning methods. This hybrid approach is well-aligned with recent trends in computer vision research, offering a balance between accuracy and computational efficiency.

Limitations of the Approach

The code's limitations are worth acknowledging. One limitation is the reliance on pre-trained models, which may not cover all possible objects in diverse scenarios. Additionally, the performance may be influenced by factors such as lighting conditions and object occlusion. Further fine-tuning of the models or

exploring advanced techniques could mitigate these limitations.

Possible Improvements

Several areas offer opportunities for improvement in the code:

Dataset Diversity: Expanding the dataset with a more diverse range of objects and scenes could enhance the models' generalization capabilities.

Algorithm Optimization: Exploring optimization techniques, such as model quantization or pruning, can improve the efficiency of real-time object detection.

User Interaction Features: Incorporating features for user interaction, such as object highlighting or feedback mechanisms, could enhance the system's usability.

Dynamic Thresholding: Implementing dynamic confidence thresholding during object detection may adapt to varying conditions, improving the system's robustness.

In conclusion, the implemented code presents a solid foundation for object recognition, integrating traditional and modern approaches. While achieving notable results, ongoing refinements and innovations could propel the system towards higher levels of accuracy, adaptability, and usability in diverse real-world scenarios.

CONCLUSION

The implemented object recognition code, merging template matching and Artificial Neural Networks (ANN), presents a robust and adaptable solution. The hybrid approach successfully leverages the strengths of both methodologies, excelling in diverse scenarios and demonstrating real-time object detection capabilities. Results, metrics, and visualizations confirm the effectiveness of the system, paving the way for ongoing improvements and future developments.

FUTURE WORK

Future iterations of the code could benefit from:

Advanced Segmentation Techniques: Exploring advanced image segmentation techniques could improve the accuracy of object boundaries, especially in complex scenes.

Interactive GUI: Developing a graphical user interface (GUI) could enhance user experience, allowing users to interact with the system and provide feedback.

Real-time Tracking: Implementing object tracking algorithms for continuous tracking of recognized objects in video streams could add a layer of temporal context to the system.

Incorporating Multiple Models: Combining the strengths of multiple pre-trained models or custom-trained models for specific object categories could

further enhance recognition accuracy.

REFERENCE

1.https://pdf.wondershare.com/ocr/extracting-text-from-image-python.html

2.https://www.google.com

3.https://www.youtube.com

4.https://www.geeksforgeeks.org