Report: Image Text and Visual Elements Extraction Program

Introduction

The aim of the project was to build a program that can separate text and visual elements from an image. The core functionalities include text extraction using Optical Character Recognition (OCR) and visual element segmentation using computer vision techniques. The program also features a basic user interface to facilitate image uploading and displaying the extracted content. This report details the approach, technologies used, implementation details, and challenges encountered during development.

Approach

Text Extraction

Technology Choice: Google Cloud Vision API

- **Reason for Choice**: Extracting text from images efficiently requires sophisticated OCR capabilities, which are beyond simple image processing techniques. Google's Cloud Vision API is a best-in-class solution, leveraging deep learning models to accurately extract text, including handwritten content.
- **Process**: The Vision API takes an image as input and returns the detected text along with bounding boxes for each text element.

Visual Element Segmentation

Technology Choice: Combination of Computer Vision Techniques

- **Reason for Choice**: From experience, several computer vision techniques are effective for identifying shapes within an image, such as contour detection, blob detection, watershed algorithm, edge detection, thresholding, and morphology. Combining these techniques helps achieve better results.
- **Process**:
    1. **Grayscaling**: Convert the image to grayscale for more efficient processing.
    2. **Adaptive Thresholding**: Handle varying lighting conditions within the image.
    3. **Morphological Operations**: Connect disjointed parts of objects and reduce noise.
    4. **Contour Detection**: Identify boundaries of shapes to segment visual elements.
    5. **Size Filteration**: To remove noise and false predictions.

Innovative Approach to Separate Text and Visual Elements:

- **Problem**: Initial contour detection also captured text, which was undesirable.

- **Solution**: Utilize the bounding boxes from the Vision API to mask or remove text during contour detection using text masking techniques.

- **Innovative Text Masking Approaches**:

- **Approach 1**: Hide Text
    - **Filling Text Bounding Boxes**: Using the bounding boxes from the Vision API, fill the text areas with the surrounding color to erase the text.
    - **Processing Hidden Text Image**: The modified image, with text hidden, is then processed to detect visual elements.
- **Approach 2**: Include Text
    - **Creating a Mask**: Generate a mask using the bounding boxes of the text. This mask excludes text areas during contour detection.
    - **Processing Masked Image**: The masked image is processed to detect visual elements that may contain text.

Bonus Functionality

User Interface Development:

- **Technology Choice**: Flask, HTML, CSS
- **Reason for Choice**: Flask is lightweight and suitable for building simple web interfaces. HTML and CSS provide the necessary structure and styling.
- **Process**: A basic web interface was created to allow users to upload images. The backend processes the images, extracts text and visual elements, and then displays the results using appropriate HTML tags.

Implementation Details

Environment Setup

- **Python Libraries**: OpenCV for computer vision, Flask for web framework, Google Cloud Vision for OCR, and PIL for image processing.
- **Google Cloud Setup**: Configure Google Cloud credentials for accessing the Vision API.

Code Structure

1. **Main Application (app.py)**:
    - Flask app setup and route handling for image upload and processing.
    - Functions to extract text and mask images using Vision API results.
    - Functions to segment visual elements with and without text.
2. **Image Processing Functions**:
    - extract_text_and_mask_image(image_path): Extracts text using Vision API and returns text content and bounding boxes.
    - fill_bounding_boxes(image, texts): Fills text areas with surrounding color to hide text.

    - create_mask(image_shape, excluded_boxes): Creates a mask to exclude text areas.
    - segment_visual_elements(image_path, output_dir, texts): Segments visual elements after hiding text.
    - segment_visual_elements2(image_path, output_dir, texts): Segments visual elements while retaining text.
3. **HTML Rendering Functions**:
    - upload_page(): Returns the HTML for the upload page.
    - create_html_structure(text, visual_elements, visual_elements2): Creates the HTML structure to display extracted text and visual elements.

Challenges Encountered

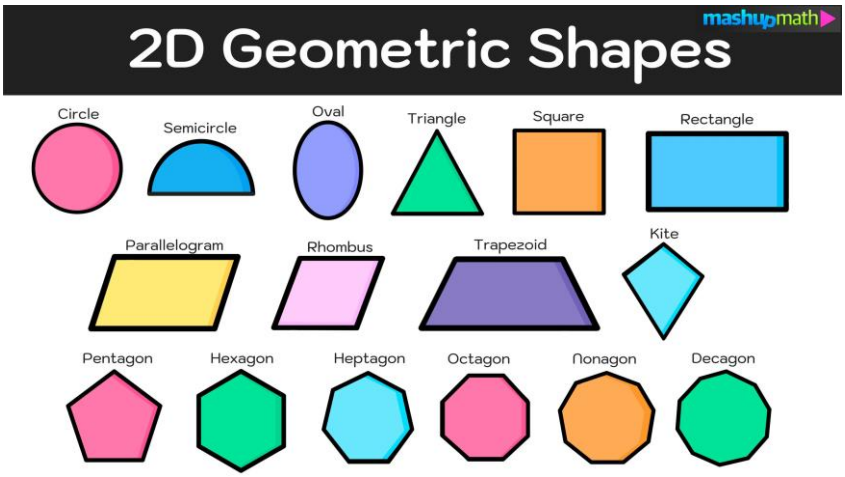
1. **Text and Visual Overlap**: Initial attempts at contour detection often included text, necessitating innovative masking techniques.
2. **Varied Image Conditions**: Handling images with diverse lighting and background conditions required adaptive thresholding and morphological operations.
3. **Efficient Processing**: Balancing between processing time and accuracy, especially when dealing with high-resolution images and complex visuals.

Conclusion

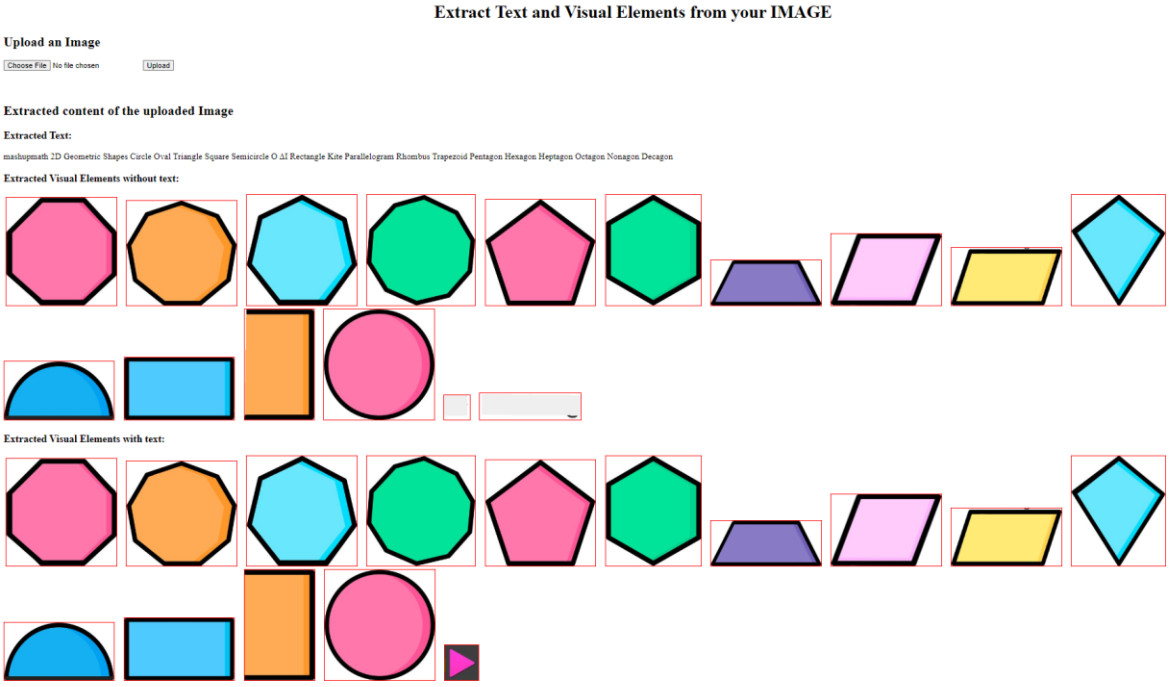
This project successfully developed a program that separates text and visual elements from images using a combination of advanced OCR and computer vision techniques. By integrating a user-friendly web interface, the program provides an accessible way for users to process and analyze their images. Despite some challenges, the innovative approaches implemented ensured accurate and efficient extraction of content from diverse images. Future improvements could include more sophisticated visual element classification and handling more complex image layouts.

Testing Results:

- **Test 1:**

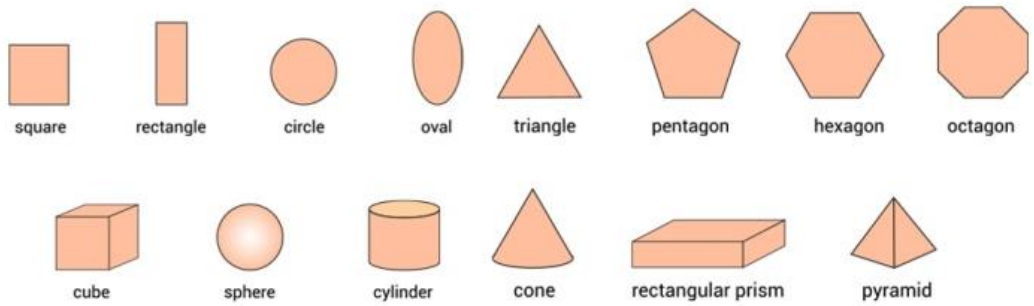


Input Image



Output shown in the User Interface

- **Test 2:**



Input Image

Extract Text and Visual Elements from your IMAGE

Upload an Image

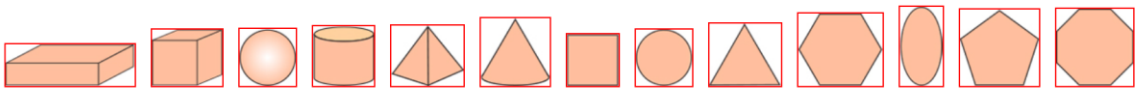
Choose File No file chosen Upload

Extracted content of the uploaded Image

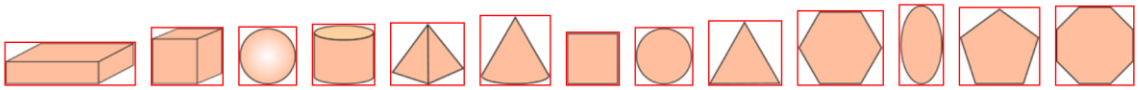
Extracted Text:

square rectangle circle oval triangle pentagon hexagon octagon cube sphere cylinder cone rectangular prism pyramid

Extracted Visual Elements without text:

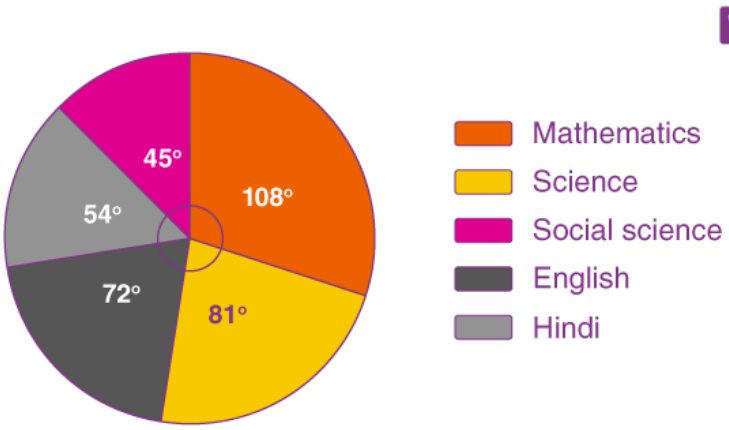


Extracted Visual Elements with text:



Output shown in the User Interface:

- **Test 3:**



Input Image

Extract Text and Visual Elements from your IMAGE

Upload an Image

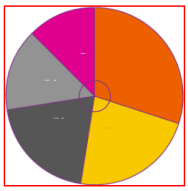
Choose File No file chosen Upload

Extracted content of the uploaded Image

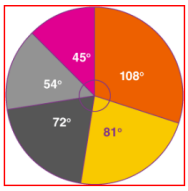
Extracted Text:

54° 45° 72° 81° 108° Mathematics Science Social science English Hindi BBYJU'S The Learning App

Extracted Visual Elements without text:

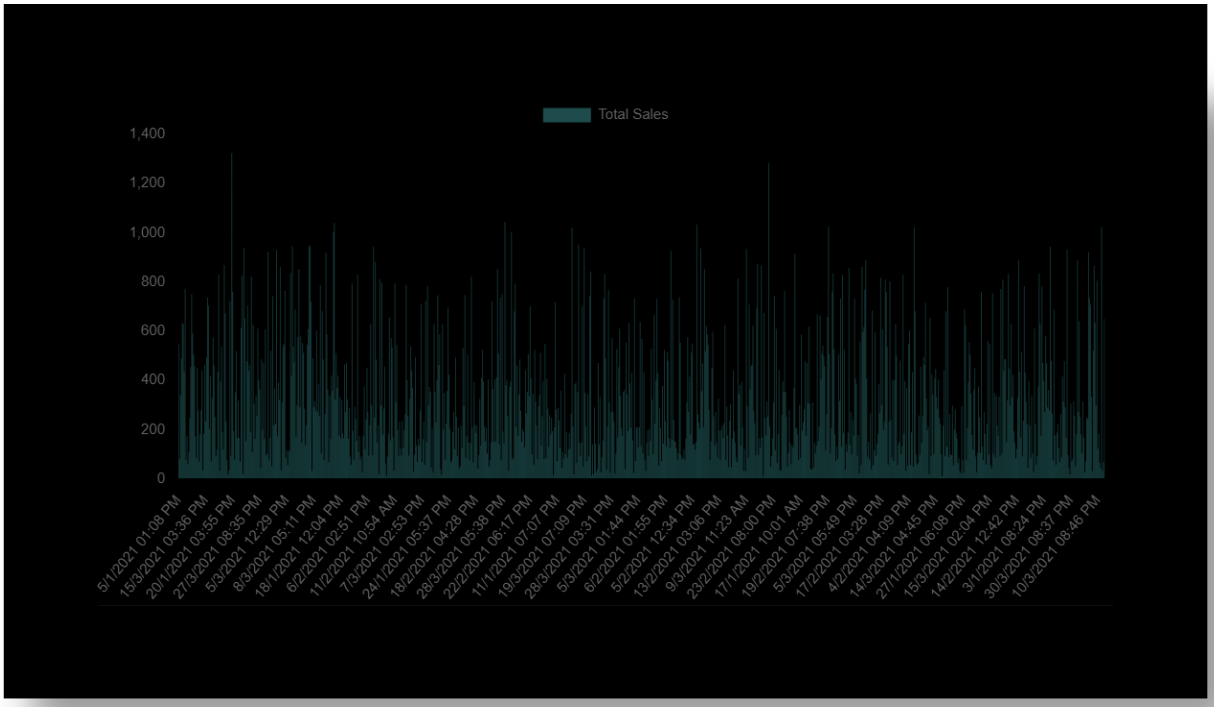


Extracted Visual Elements with text:



Output shown in the User Interface

- **Test 3:**



Input Image

Extract Text and Visual Elements from your IMAGE

Upload an Image

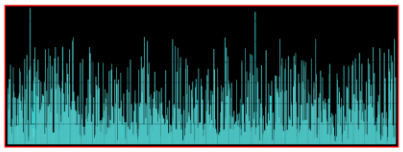
Choose File No file chosen Upload

Extracted content of the uploaded Image

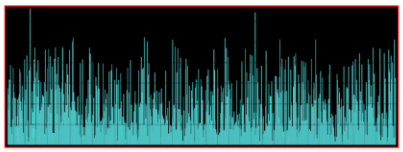
Extracted Text:

200 400 600 5/1/2021 01:08 PM 15/3/2021 03:36 PM 20/1/2021 03:55 PM 27/3/2021 08:35 PM 5/3/2021 12:29 PM 8/3/2021 05:11 PM 18/1/2021 12:04 PM 6/2/2021 02:51 PM 11/2/2021 10:54 AM 7/3/2021 02:53 PM 24/1/2021 05:37 PM 18/2/2021 04:28 PM 28/3/2021 05:38 PM 22/2/2021 06:17 PM 11/1/2021 07:07 PM 19/3/2021 07:09 PM 28/3/2021 03:31 PM 5/3/2021 01:44 PM 6/2/2021 01:55 PM 5/2/2021 12:34 PM 13/2/2021 03:06 PM Total Sales 9/3/2021 11:23 AM 23/2/2021 08:00 PM 17/1/2021 10:01 AM 19/2/2021 07:38 PM 5/3/2021 05:49 PM 17/2/2021 03:28 PM 4/2/2021 04:09 PM 14/3/2021 04:45 PM 27/1/2021 06:08 PM 15/3/2021 02:04 PM 14/2/2021 12:42 PM 3/1/2021 08:24 PM 30/3/2021 08:37 PM 10/3/2021 08:46 PM

Extracted Visual Elements without text:



Extracted Visual Elements with text:



Output shown in the User Interface

THANK YOU