**Image Segmentation and Object Identification Pipeline Documentation**

**Table of Contents**

1. [Introduction](https://stackedit.io/app#introduction)
2. [Project Structure](https://stackedit.io/app#project-structure)
3. [Modules Overview](https://stackedit.io/app#modules-overview)

* [Image Segmentation](https://stackedit.io/app#image-segmentation)
* [Object Extraction](https://stackedit.io/app#object-extraction)
* [Object Identification](https://stackedit.io/app#object-identification)
* [Text Extraction](https://stackedit.io/app#text-extraction)
* [Attribute Summarization](https://stackedit.io/app#attribute-summarization)
* [Data Mapping](https://stackedit.io/app#data-mapping)
* [Output Generation](https://stackedit.io/app#output-generation)

1. [Streamlit Application](https://stackedit.io/app#streamlit-application)
2. [Conclusion](https://stackedit.io/app#conclusion)

**1. Introduction**

This project aims to create a comprehensive pipeline for image segmentation, object extraction, identification, text extraction, and summarization of object attributes. The pipeline is designed to process images and provide meaningful insights into the objects contained within them.

**2. Project Structure**

The project consists of several modules, each responsible for specific functionalities. The overall structure is as follows:

/project\_root

├── app.py

├── segment.py

├── object\_extractor.py

├── object\_identifier.py

├── text\_extractor.py

├── attribute\_summary.py

├── data\_mapping.py

├── output\_generation.py

├── extracted\_objects/

├── metadata/

├── identified\_objects.json

├── extracted\_text.json

└── summarized\_attributes.json

**3. Modules Overview**

**Image Segmentation**

* **File Name**: segment.py
* **Description**: This module is responsible for segmenting images into distinct regions or objects.
* **Key Functions**:
* process\_image(image): Segments the input image and returns the original image, segmented image, masks, bounding boxes, and labels for each object.

**Object Extraction**

* **File Name**: object\_extractor.py
* **Description**: This module extracts segmented objects from the image and saves them as individual files.
* **Key Functions**:
* extract\_objects(original\_image, masks, boxes, labels): Takes the original image, masks, bounding boxes, and labels to extract and save each object.

**Object Identification**

* **File Name**: object\_identifier.py
* **Description**: This module identifies the extracted objects using a pre-trained YOLOv5 model. It assigns labels and confidence scores to each identified object, filtering out low-confidence predictions to ensure the quality of results.
* **Key Functions**:
* identify\_objects(extracted\_objects): Accepts a list of extracted objects and returns a list of identified objects with labels and confidence scores.

**Text Extraction**

* **File Name**: text\_extractor.py
* **Description**: This module extracts text from the identified objects using Optical Character Recognition (OCR) techniques.
* **Key Functions**:
* extract\_from\_objects(identified\_objects): Extracts text data from the identified objects and saves it.
* save\_extracted\_text(extracted\_text\_data, output\_file): Saves the extracted text to a JSON file.

**Attribute Summarization**

* **File Name**: attribute\_summary.py
* **Description**: This module summarizes the attributes of identified objects based on extracted text and object labels.
* **Key Functions**:
* summarize\_attributes(): Summarizes attributes for each identified object.
* save\_summary(summary, output\_file): Saves the summarized attributes to a JSON file.

**Data Mapping**

* **File Name**: data\_mapping.py
* **Description**: This module maps the extracted data to each object and the master input image, creating a comprehensive data structure.
* **Key Functions**:
* load\_data(): Loads necessary data for mapping.
* save\_mapped\_data(output\_file): Saves the mapped data to a JSON file.

**Output Generation**

* **File Name**: output\_generation.py
* **Description**: This module generates the final output image with annotations and a data table summarizing the results.
* **Key Functions**:
* generate\_final\_output(output\_file): Generates and saves the final output image with annotations.

**4. Streamlit Application**

**8. Streamlit App**

* **File Name**: app.py
* **Description**: This is the main application file that serves as the user interface for the image segmentation, extraction, identification, and attribute summarization pipeline. It uses Streamlit to create an interactive web application where users can upload images and view results at each step of the processing pipeline.

**Key Features:**

* **Image Upload**: Users can upload images directly through the web interface.
* **Step-by-Step Processing**: The app guides users through each step of the pipeline, displaying results and outputs after each major action.
* **Dynamic Display**: The app dynamically displays segmented images, extracted objects, identified labels, confidence scores, and summarized attributes.
* **Final Output Visualization**: The application shows the final output image with annotations and a data table summarizing the extracted data.

**Example Usage:**

1. **Run the Streamlit Application**:

* Open a terminal and navigate to the directory containing the app.py file.
* Execute the command:

streamlit run app.py

1. **Access the Web Interface**:

* After running the command, a new tab will open in your default web browser, displaying the Streamlit app.
* Alternatively, you can access it via http://localhost:8501.

1. **Upload an Image**:

* Click on the “Upload Image” section to upload an image in JPG or PNG format.
* The app will automatically process the image through the segmentation, extraction, identification, and summarization steps.

1. **View Results**:

* After each step, the app will display relevant results, including segmented images, extracted objects, identified labels with confidence scores, and summarized attributes.
* The final output will include an annotated image alongside a data table.

**Requirements:**

* Ensure that all dependencies required by the pipeline (e.g., Streamlit, YOLOv5, PyTorch, OpenCV, and any other libraries) are installed. This can typically be done via:

pip install -r requirements.txt

Make sure to create a requirements.txt file that lists all the necessary packages for your project.

**5. Conclusion**

This documentation outlines the complete workflow of the image segmentation and object identification pipeline, including all modules, their functions, and the user interface provided by the Streamlit application. Users can leverage this pipeline for various applications, such as automated image analysis and reporting.