### ****Smart Furnish Application Documentation and Report****

#### ****1. Introduction****

The **Smart Furnish** application is an interactive tool designed to allow users to seamlessly remove objects from images and virtually place furniture items in their desired location. This application leverages advanced deep learning models such as **FastSAM** for object segmentation and **SimpleLama** for inpainting (image restoration) to achieve realistic image manipulation. The implementation is powered by **Streamlit**, which offers an intuitive interface for image processing tasks.

### ****2. Methodology****

#### ****2.1 Data Preparation****

**Input Data:**

* Users provide an input image by uploading it via the Streamlit interface.
* Accepted file types: JPEG, PNG, JPG.
* The image is resized (maintaining aspect ratio) if its width exceeds 700 pixels to ensure it fits well within the application interface.

**Bounding Box Selection:**

* A drawable canvas is used to capture the user's selection for object removal.
* Users draw a rectangular bounding box around the object they want to remove, providing an interactive approach to selecting areas of interest.

**Furniture Data:**

* Furniture images (stored locally) are categorized into different types (Chair, Table, Other Furniture).
* Users can select specific furniture images from the options provided in the sidebar.

#### ****2.2 Model Architecture****

**FastSAM:**

* **FastSAM** (Fast Segment Anything Model) is utilized for segmenting objects within a selected bounding box.
* It is a lightweight and efficient variant of the Segment Anything Model (SAM), designed for fast and accurate object segmentation.
* **FastSAM** can detect multiple object instances within a specified region using bounding boxes and masks, with settings for confidence (conf=0.2) and Intersection over Union (iou=0.5).

**SimpleLama:**

* **SimpleLama** is an inpainting model based on a simplified version of the LaMa (Large Mask) architecture, specifically designed for filling in missing or removed regions in an image.
* It uses a convolutional neural network (CNN) to understand the context around the masked area and generates realistic inpainting results.

#### ****2.3 Implementation Details****

**Step 1: Image Upload and Resizing**

* The application resizes the image if its width exceeds 700 pixels for better visualization.
* The resized image is displayed, and users can proceed to draw a bounding box around the object to be removed.

**Step 2: Object Detection and Mask Creation**

* When a user selects an area for object removal, the **FastSAM** model detects objects within the bounding box.
* The model creates a mask for the detected objects, and the mask is stored in the session state.

**Step 3: Inpainting with SimpleLama**

* If an object is detected, the inpainting process begins. The masked area is passed to the **SimpleLama** model.
* The model fills in the selected area based on surrounding pixels, effectively removing the object from the image.

**Step 4: Furniture Selection and Placement**

* Users choose a furniture type and select an image from the given options.
* The chosen furniture image is processed through **FastSAM** to extract its segmentation mask.
* The segmented furniture image is resized to fit the bounding box area left by the removed object.

**Step 5: Interactive Drag-and-Resize Feature**

* The application allows users to drag the furniture across the image and resize it using mouse interactions.Leveraged Opencv for this feature
* The new position and size of the furniture are continuously updated and displayed in real time.

**Step 6: Saving the Final Image**

* Users can save the final image by pressing 's' after positioning the furniture.

The user interface is created by one of the SOTA yet simplistic python UI framework called Streamlit.

### ****3. Algorithm Performance, Limitations, and Improvements****

#### ****3.1 Performance Analysis****

**FastSAM:**

* **Strengths:**
  + Fast object detection with efficient use of bounding boxes and masks.
  + Handles multiple object instances well and provides a segmented mask.
* **Weaknesses:**
  + Struggles with complex scenes or occluded objects where precise segmentation is difficult.
  + Performance can drop when the confidence threshold is set too low, leading to false positives.

**SimpleLama:**

* **Strengths:**
  + Provides realistic and smooth inpainting results by filling in the masked region using context from surrounding pixels.
  + Effective in simple and moderately complex backgrounds where the inpainting region has a clear context.
* **Weaknesses:**
  + Fails in highly complex or textured areas where it cannot infer missing details correctly, leading to noticeable artifacts.
  + May produce blurred or smeared results when the surrounding area lacks distinguishable features.

#### ****3.2 Limitations****

1. **Limited Object Categories for Detection:**
   * The current implementation focuses on segmenting objects within a manually defined bounding box. However, it may miss detecting objects outside this region, requiring users to precisely define areas of interest.
2. **Dependency on Mask Quality:**
   * The inpainting quality is highly dependent on the accuracy of the mask produced by **FastSAM**. Incorrect masks or masks that do not tightly fit the object can lead to imperfect removal and unrealistic inpainting.
3. **User Interaction for Furniture Placement:**
   * While interactive placement is a key feature, it is dependent on precise user control, which can be challenging on smaller screens or with high-resolution images.
4. **Performance on High-Resolution Images:**
   * The processing time for high-resolution images can be significant, affecting user experience. There might be delays in detecting objects and generating masks.

#### ****3.3 Potential Improvements****

1. **Enhanced Object Detection:**
   * Implementing an improved version of **FastSAM** or integrating other models like **YOLOv8** or **Mask R-CNN** could enhance detection accuracy, especially in crowded or complex scenes.
2. **Adaptive Mask Refinement:**
   * Adding a post-processing step using morphological operations (e.g., erosion and dilation) can refine the generated mask, leading to better inpainting results.
3. **Automated Region Suggestion:**
   * Incorporating a feature that automatically suggests regions for object removal based on object saliency or user-defined criteria can reduce manual intervention.
4. **Furniture Placement Suggestions:**
   * Implement an AI-based suggestion system to recommend optimal furniture placement based on the layout and detected features of the room in the image.
5. **User Feedback Integration:**
   * Adding interactive sliders for adjusting confidence levels and thresholds in segmentation models can give users more control over detection accuracy.