Approach of the Algorithm

- The algorithm starts by loading a pre-trained object detection model (facebook/detr-resnet-50)

- The script uses the `argparse` library to handle command-line arguments. These arguments include the image file path, the object class to detect, azimuth and polar angles for rotation, and the output file path.

- The input image is loaded using the Python Imaging Library (PIL), and converted to a NumPy array for processing.

- The loaded image is preprocessed using the processor associated with the model. This step prepares the image for inference by normalizing it and converting it to a format suitable for the model.

- The model performs inference on the preprocessed image, generating output logits and predicted bounding boxes. The predicted classes for each bounding box are determined using softmax activation.

- The algorithm selects the bounding boxes corresponding to the specified object class (in this case, "chair"). The boxes are normalized to the image dimensions.

- If no azimuth or polar angles are provided, the algorithm applies a red mask to the detected object. This is done by iterating over the bounding boxes and setting the pixel values within those boxes to red.

- If azimuth or polar angles are provided, the algorithm rotates the detected object(s) based on the specified azimuth angle. The rotation is performed using OpenCV's `warpAffine` function, which allows for transformation of the image around the center of the object.

- Before performing any operations on the bounding boxes, the algorithm includes checks to ensure that the box coordinates are within the image dimensions. This prevents errors that could arise from trying to access pixels outside the image bounds.

- The script prints the output path to indicate where the generated image has been saved.

But warpAffine function is failing so the task 2 is failling. I will fix it as early as possible.