Machine Learning Assignment

Finding pick point in 2D image

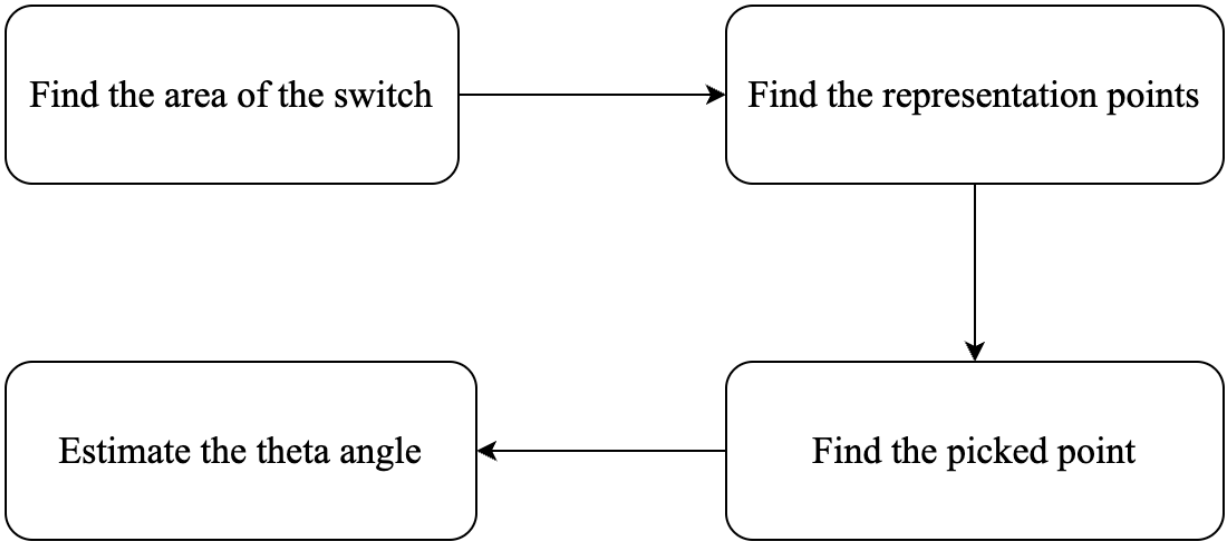
Author

Nguyễn Phú Vượng

Contact

Email: vuong.np040119991@gmail.com
Phone: 0982385326

1. Main steps



Picture 1. Main steps

1.1. Find the area of the switch

- Use a segmentation model to find switch areas in the image (top or overlap).

Procedure:

1. Prepare data (see prepare_data.py file)
   1.1. Split the data in train_ids.txt into train and val sets with a 6:1 ratio.
   1.2. Process the data in YOLO format, which includes two labels – 0 for "Top" and 1 for "Overlap".

2. Train the YOLOv8 segmentation model (see train.py file)
    2.1. See the results of the segmentation model in Section 2.1
3. Use the trained model to predict the switch area in the image
    3.1. Use a shallow IOU threshold (0.1) to eliminate switches with overlapping bounding boxes.

Another option: I think we can use Polygon Detection to find the switch's area instead of Segmentation.

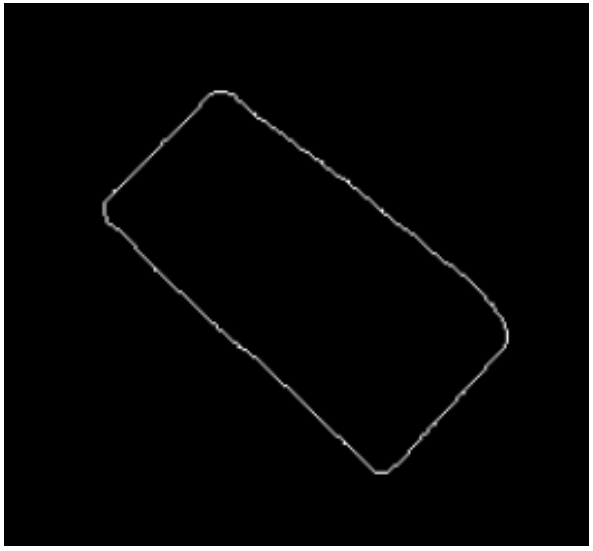
1.2. Find the representation points

- The results of the segmentation model are masks (consisting only of the numbers 0 and 1 representing the appearance of objects in the image).

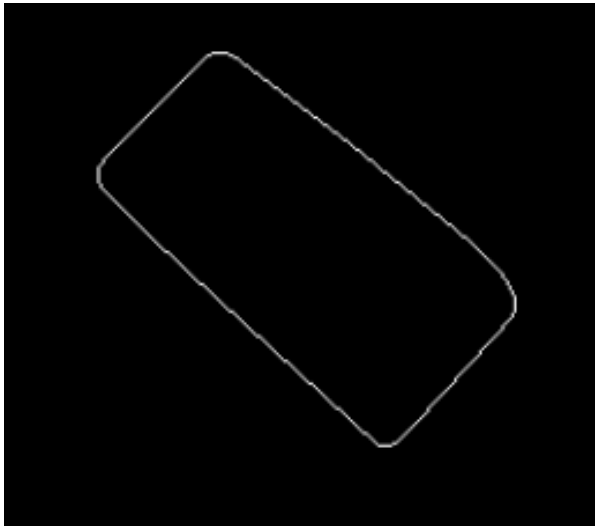
Procedure:

1. Only process objects labeled 0 (Top switches).
2. Find contours in the mask using OpenCV library. Applying the opening technique to remove noises before finding contours. (See *Picture 2*)
3. Find the convex hull of the longest contour. (See *Picture 3*)
4. Remove the convex hulls that do not satisfy the conditions (too short, too small). Because the camera projects straight and perpendicular to the switch tray → the convex hulls found should have approximately the same area.
5. Arrange satisfactory hulls in decreasing order of area (priority is given to handling hulls with larger areas). (See *Picture 8a & 8b*)

See implementation for more details.



Picture 2. Contour

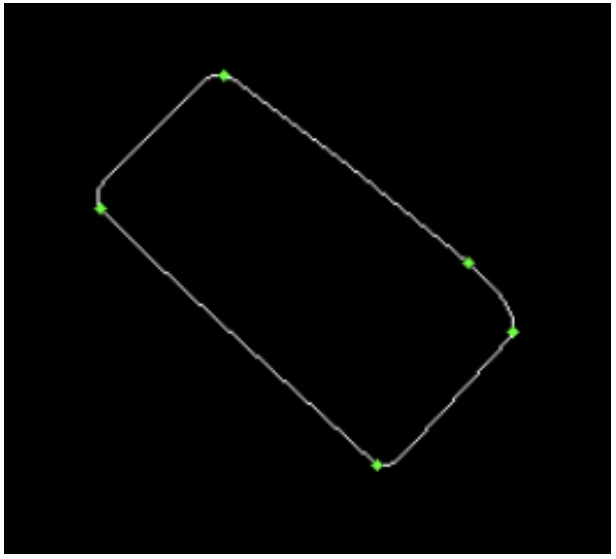


Picture 3. Convex hull

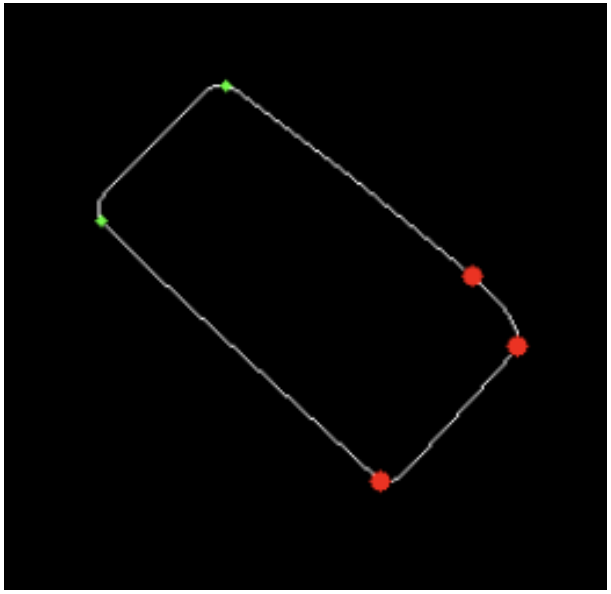
1.3. Find the picked point

Procedure:

1. Loop through the hulls found in step 1.2
2. Skip the current hull if it overlaps with any previously processed hull.
3. Find 5 points representing the convex hull (5 points form the polygon with the largest area). (See *Picture 4*)
4. Find the center point of these 5 points.
5. Find 3 points opposite the remaining 2 points (called the 3 bottom points and the 2 top points). These are the 3 points with the smallest total distance from each other among the 5 points. (See *Picture 5*)
6. Find the 2 points farthest from the center point from the 3 bottom points. Combine these 2 points with the 2 top points to get 4 points. The picked point is the center of these 4 points. (See *Picture 6*)



Picture 4. Five representation points (green points)



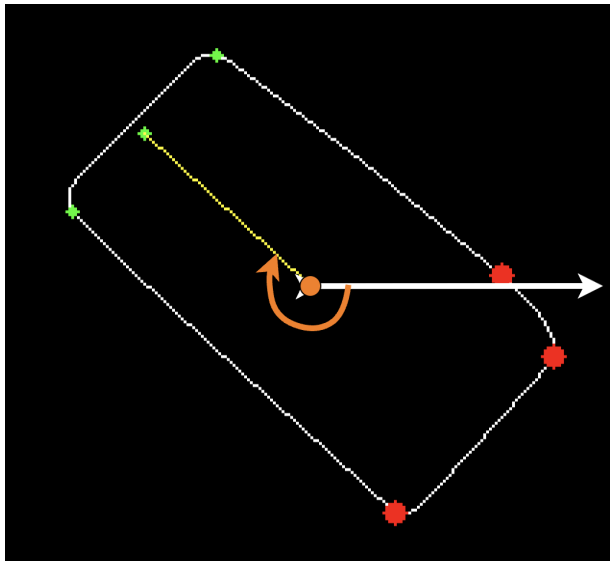
Picture 5. Three bottom points (red points)

See implementation for more details.

1.4. Estimate the theta angle

Procedure:

1. Get the center point of the 2 top points (called End point).
2. Calculate the angle between the Picked point (or Start point) and the End point with the x-axis. (See *Picture 6*)



Picture 5. Picked point and theta angle (orange)

See implementation for more details.

2. Result

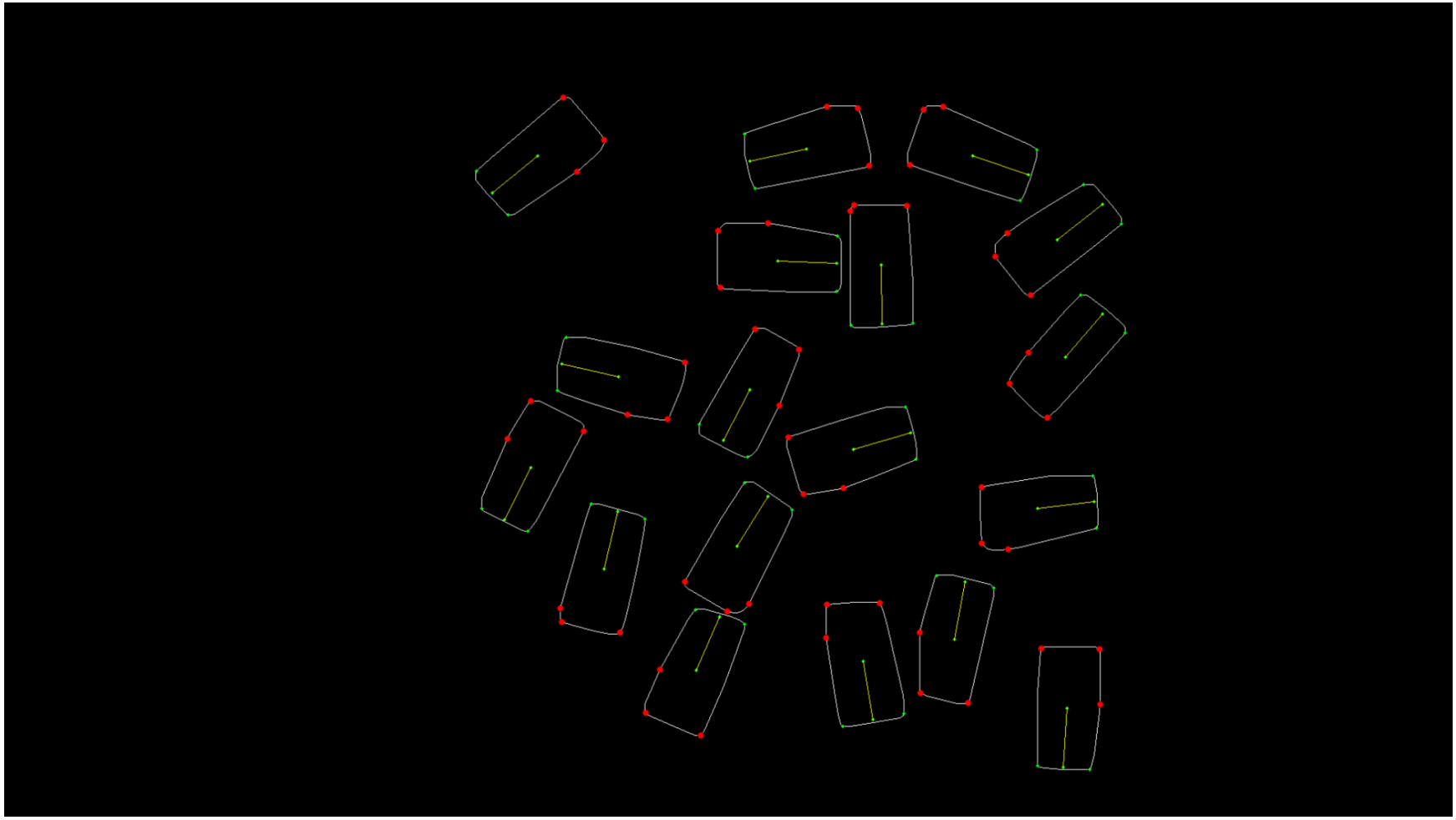
2.1. Segmentation model

Model	Box				Mask			
	AP@0.5		AP@0.5:0.95		AP@0.5		AP@0.5:0.95	
	Top	Overlap	Top	Overlap	Top	Overlap	Top	Overlap
YOLOv8s_640x640	93.5	80.0	84.5	61.2	93.5	79.9	78.4	50.9
YOLOv8s_960x960	94.0	83.3	86.7	65.6	94.0	83.3	81.3	59.0
YOLOv8s_1920x1080	**96.1**	**84.0**	**90.3**	**71.4**	**96.1**	**84.9**	**86.0**	**64.1**

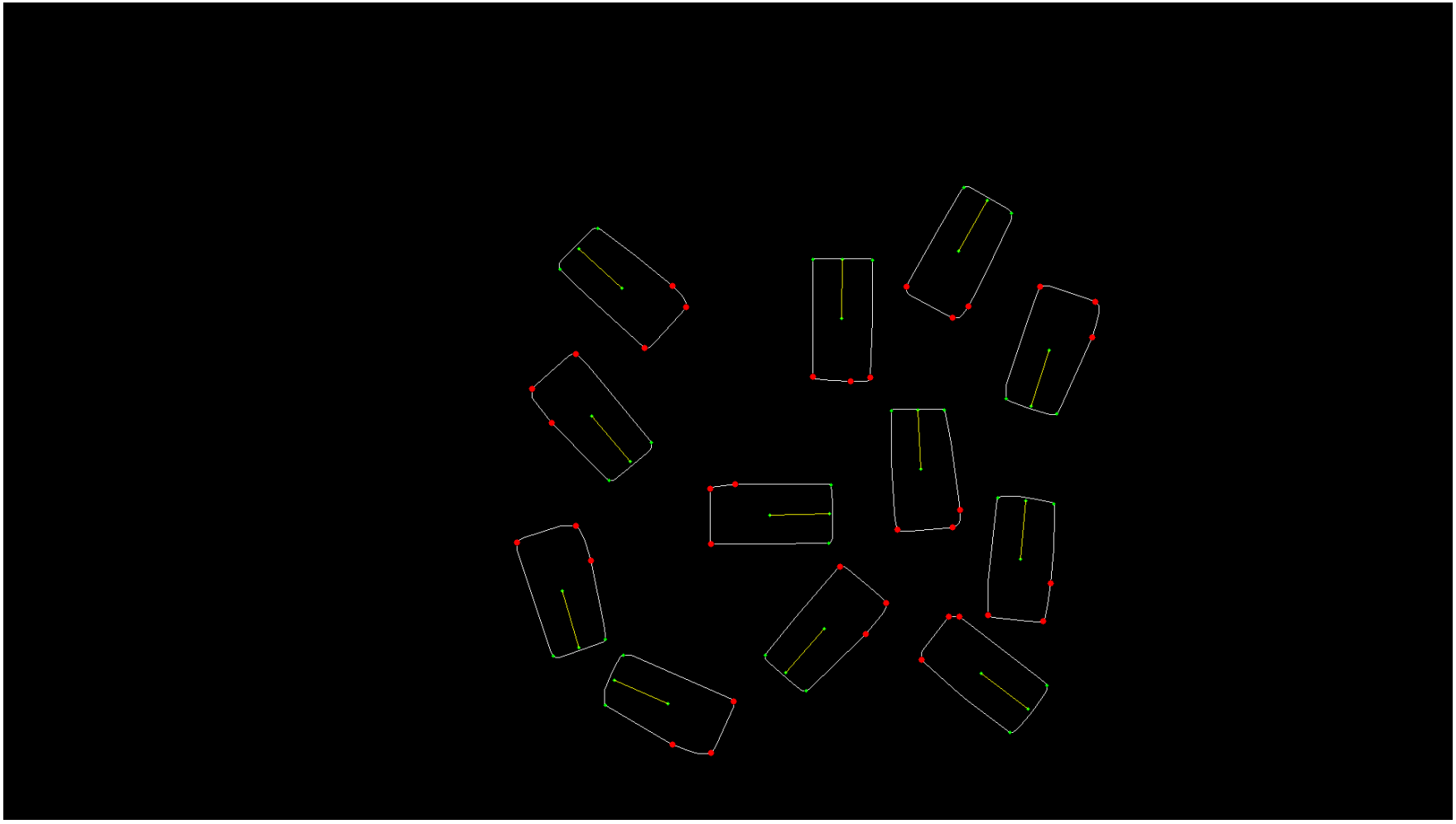
- The YOLOv8s_1920x1080 model has the highest accuracy. Since there are no constraints on processing latency, I choose to use this model.
- Using a model with larger sizes (M, L, X) could help improve accuracy, but I cannot do it yet due to the lack of resources.

2.2. Picked points in 2D images

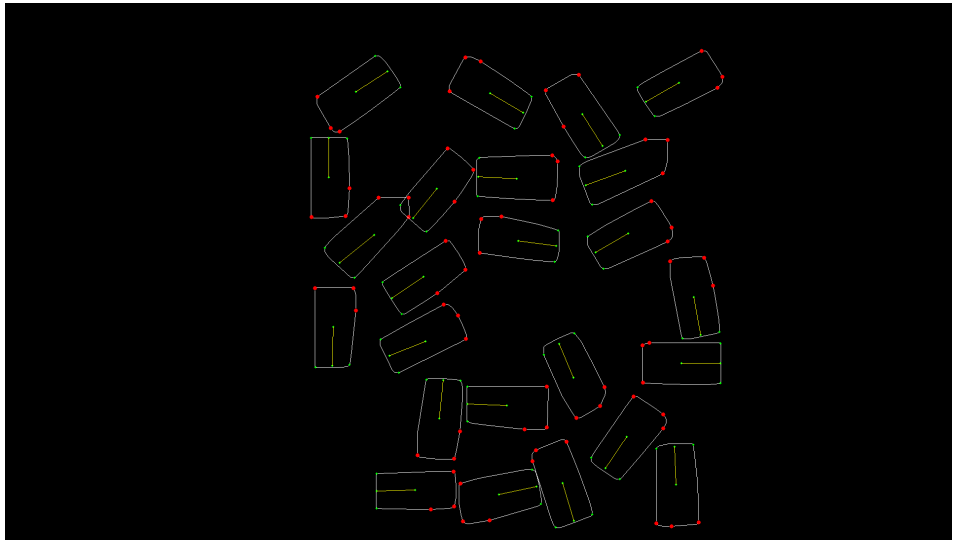
- See all results on test images: Link



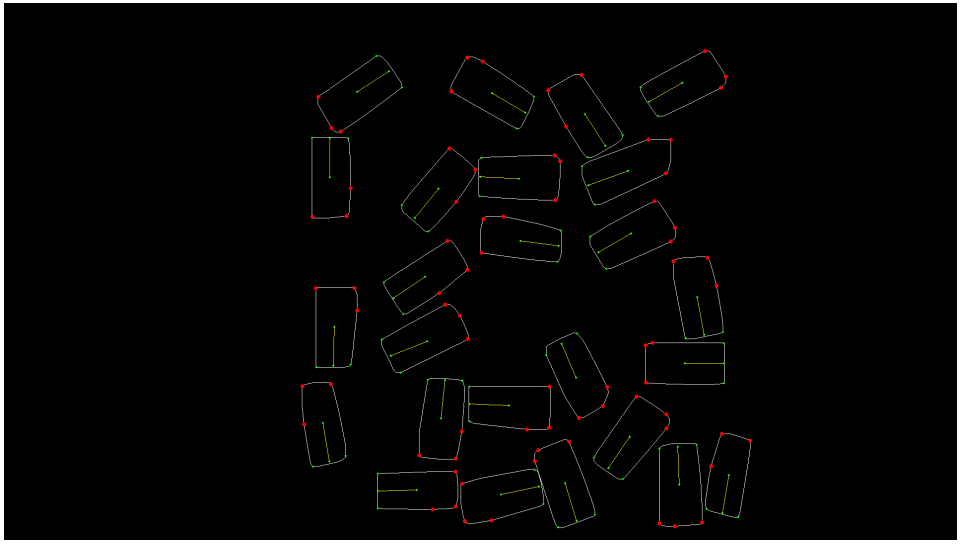
Picture 7a. Sample result on a test image.



Picture 7b. Sample result on a test image.



Picture 8a. The result on the test image before prioritizing the larger hulls



Picture 8b. The result on the test image after prioritizing the larger hulls

3. References

1. Source code: Link
2. External libraries and frameworks:
    a. Segmentation model: Ultralytics
    b. Image processing: OpenCV
    c. Polygon processing: Shapely