Salient object detection and automated cropping

Presented by Alexander Utkov

Outline

- Business Context
- Baseline
- Solution
- Results
- Future Prospects

Goal

What?

The goal is to detect the most salient part of an image, crop it and inpaint into geometric figure, most suitable for it.

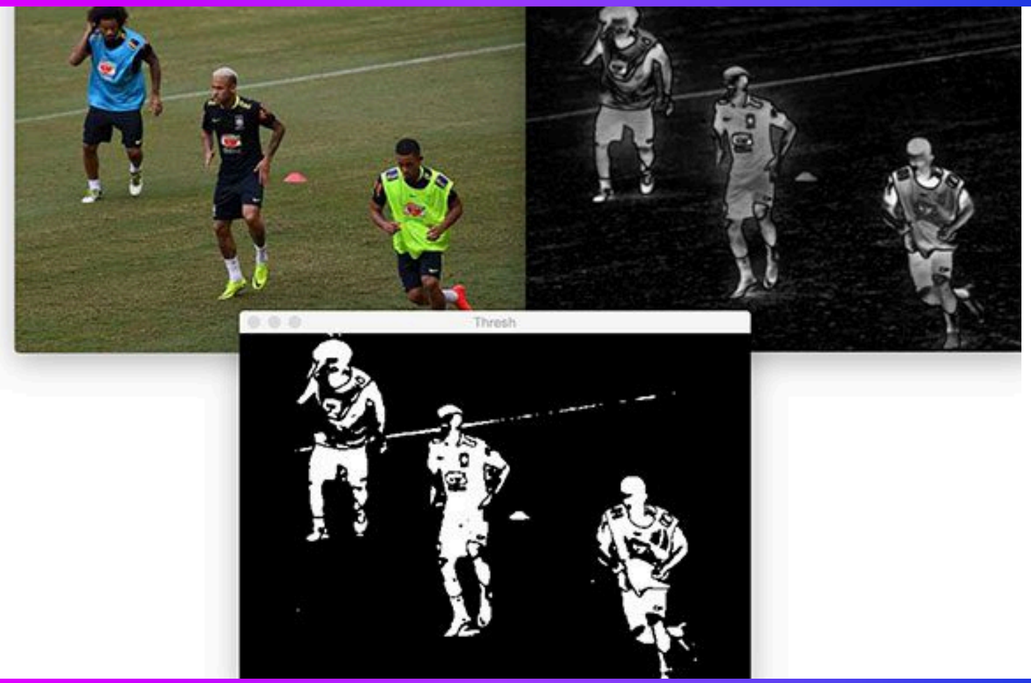
Why?

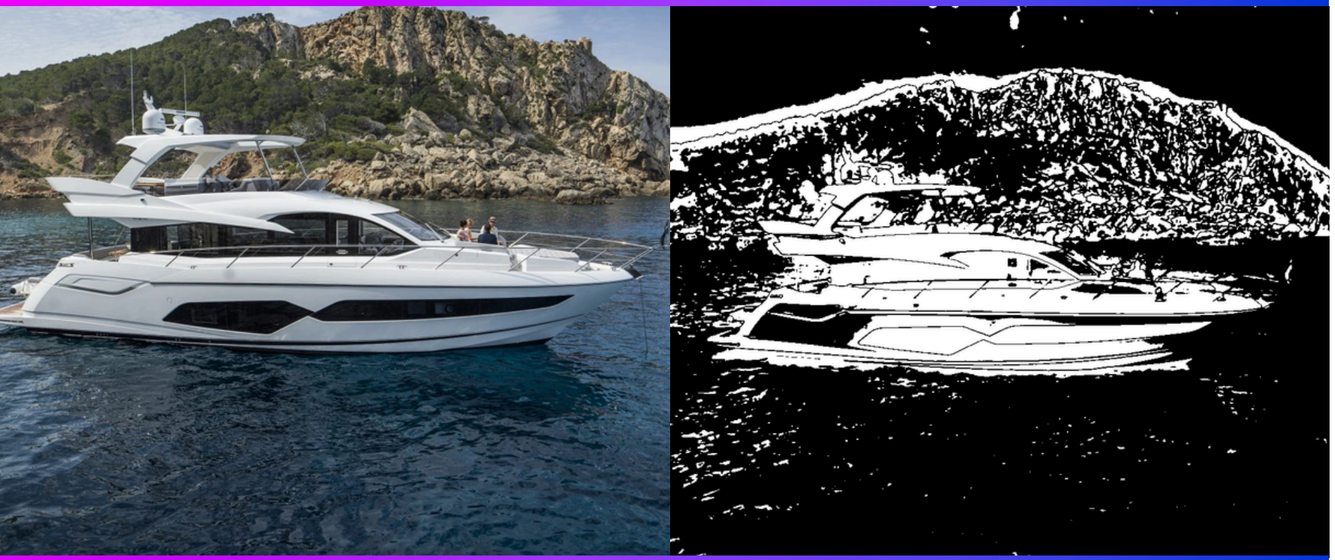
It’s a component for a system, which automatically improves design of your presentations using AI.
And images inside geometric figures used very often.

Baseline

<u>GroundingDINO</u> + prompt

- Unclear prompt, inaccurate result
- This pipeline only supports rectangular shapes, and we want custom blobs
- Resource-intensive, too big neural network for such a task



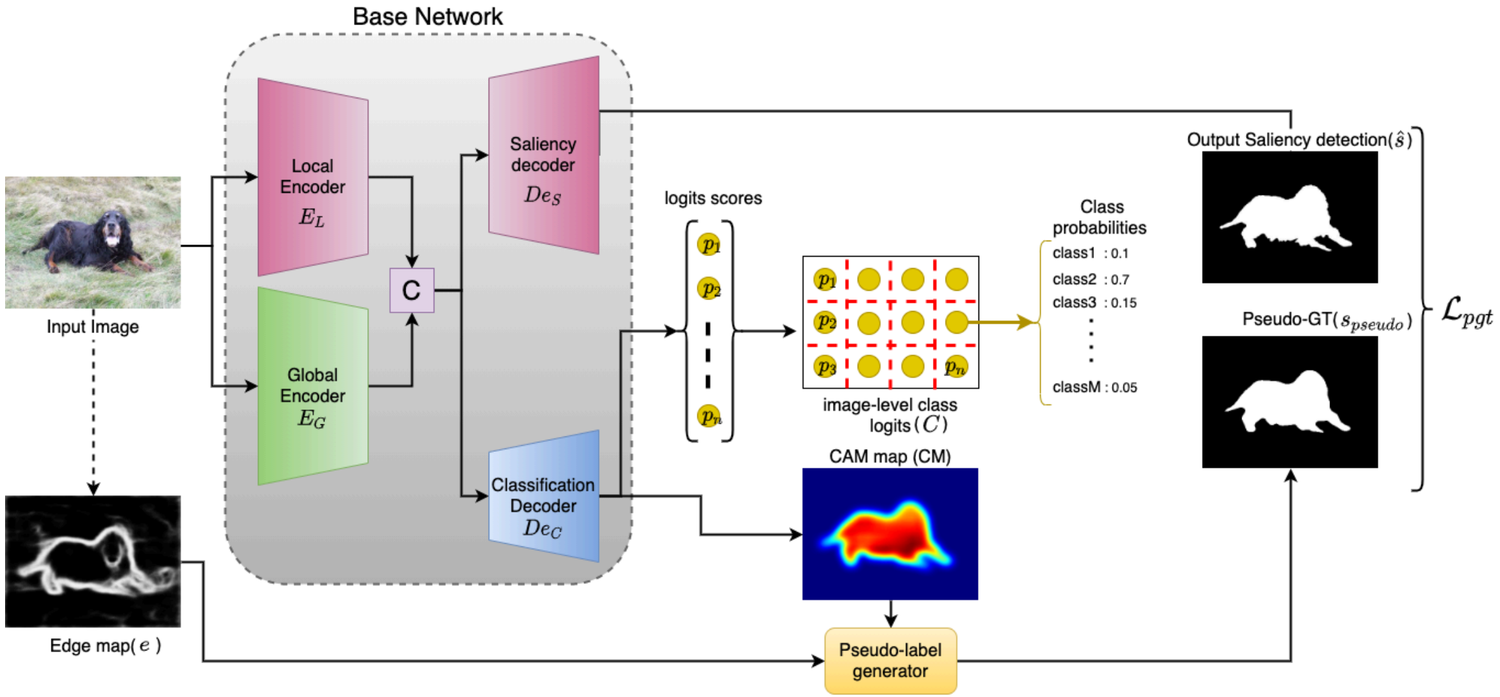


Baseline

OpenCV

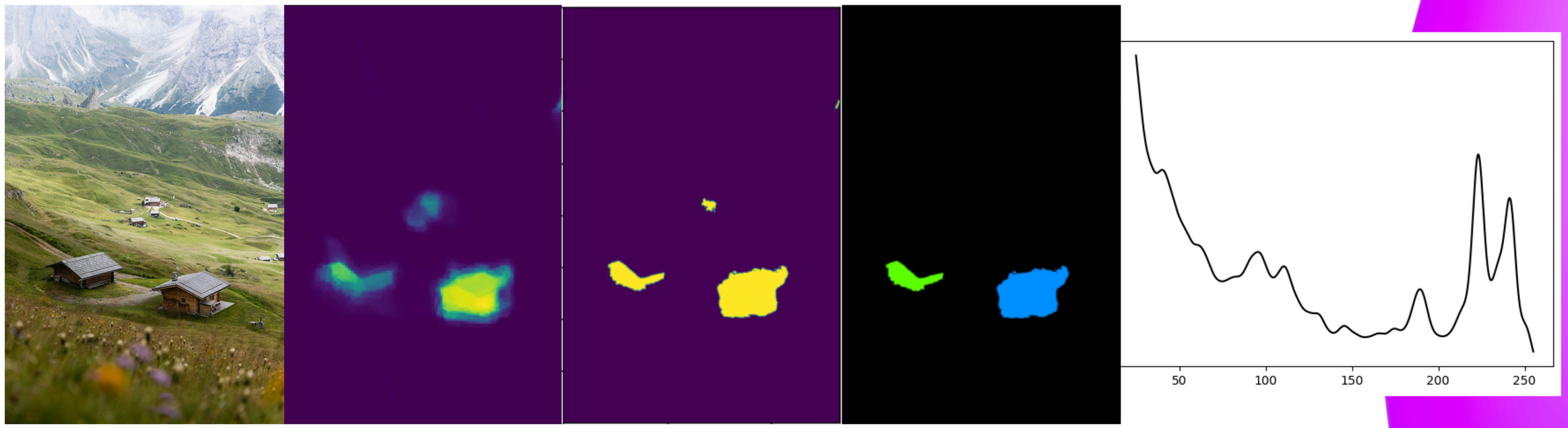
- Works fast on simple convolutions
- Allows us to get custom shaped blobs, so we can match more figures
- Output is extremely noisy
- Classical CV algorithms do not “understand” what’s happening on the photo

Solution



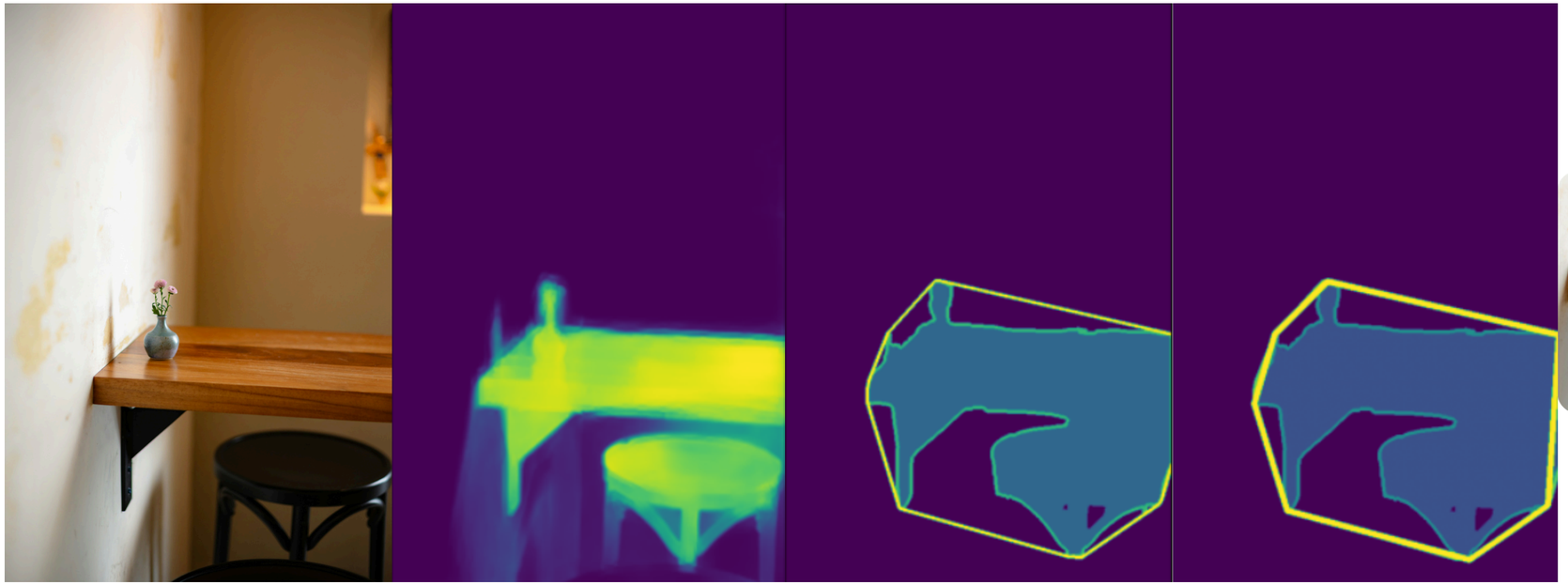
- 3SD: Self-Supervised Saliency Detection
- We train model both for saliency detection and object detection and use class activation maps to get pseudo labels for saliency detector
- For classification, training is similar to what was in DINO's article, due to which the network focuses much better on the object itself, and not on background features

Solution



- We use dynamic threshold (looking for a local minimum) to binarize the mask. Typically this is ~100 on a scale of 0 to 255.
- We use DBSCAN for clustering and filtering out unnecessary things. K-means is not very good for non-convex objects, and OPTICS is longer for small objects

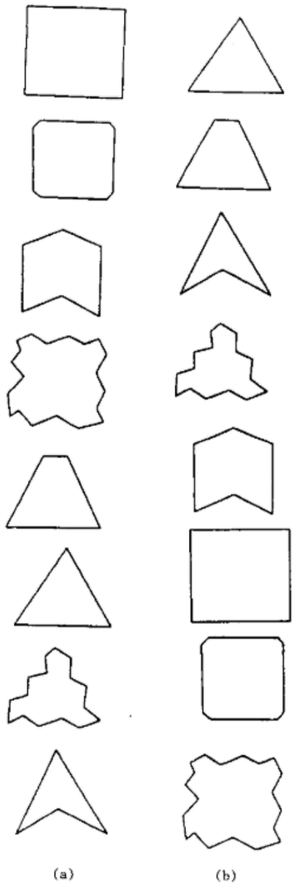
Magic of OpenCV

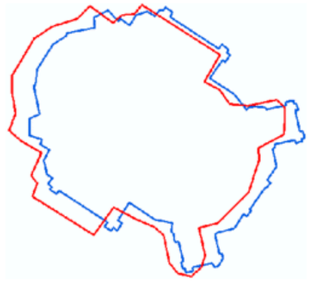


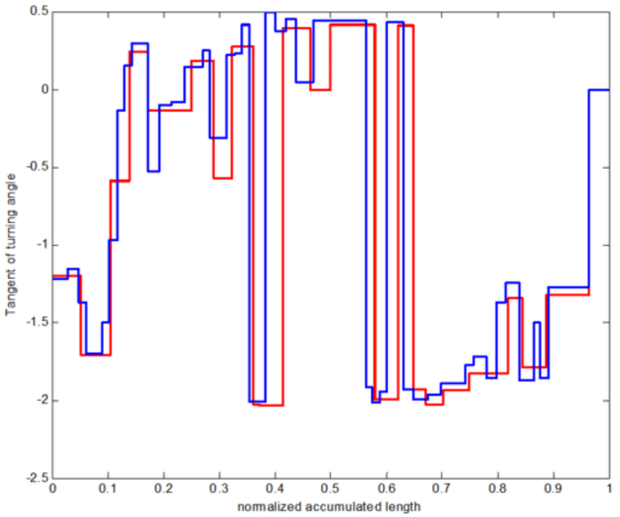
- cv2.findContours allows us to turn a mask into a precise contour
- cv2.moments allows us to find the first central moment to compare the center of the figure and the center of an important object
- cv2.convexHull allows us to build a surrounding contour for an important object, cv2.approxPolyDP will brute force simplify this contour until the lost usable area becomes minimal
- Turning function allows us to select the closest polygon

Turning function

- Function for calculating proximity between polygons
- Sensitive to rotation and selection of the starting point
- Measured from 0 to infinity
- A value less than 0.5 is generally considered similar by people

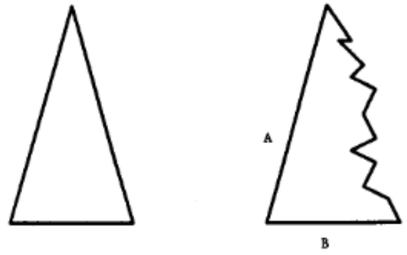






a. a pair of footprints

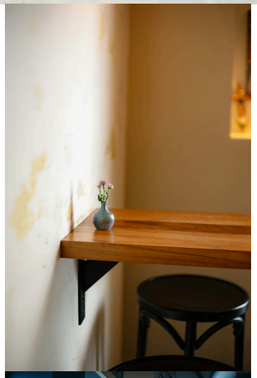
b. overlap of the turning functions

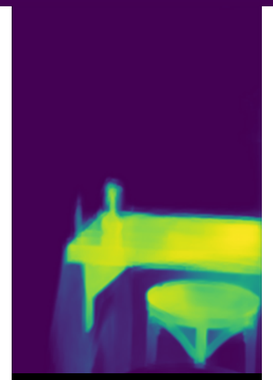


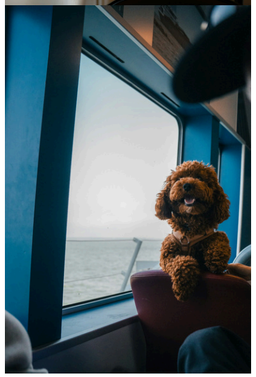
$$d^2_{A,B} = \int (\theta_A(s) - \theta_B(s))^2 ds$$

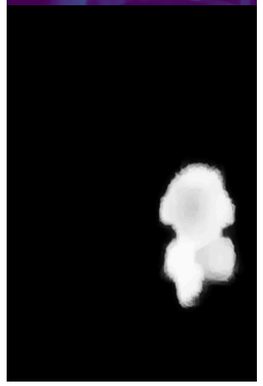
https://ieeexplore.ieee.org/document/75509

Examples

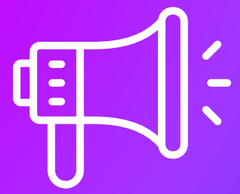








Future Prospects



01 Increase model accuracy

02 Improve algorithm speed

03 Add support of more complex polygon shapes

04 Enhance inpainting algorithm

Thanks for your attention