Efficient Automatic Annotation of Binary Masks for Enhanced Training of CV Models

Dylan Seychell
Matthew Kenely
Matthias Bartolo
Carl James Debono
Mark Bugeja
Matthew Sacco

L-Università ta' Malta
Faculty of Information &
Communication Technology

Department
of Artificial
Intelligence

Not All Datasets Are Created Equal

A tool to address this.

Mask to Annotation tool converts a binary mask to an annotation in the COCO, VGG and YOLO format.

Empowers small datasets to be also included in the training data alongside other bigger datasets such as the COCO dataset.

Problem Definition

Training data in computer vision necessitates **augmentation** due to quantity and quality requirements.

Annotating supplementary datasets which contain binary masks but lack standard annotation formats like COCO, VGG or YOLO can be tedious and time-consuming.

Current state-of-the-art segmentation techniques, such as Segment Anything Model (SAM), though effective, have excessively complex architectures and are computationally expensive for such a task.

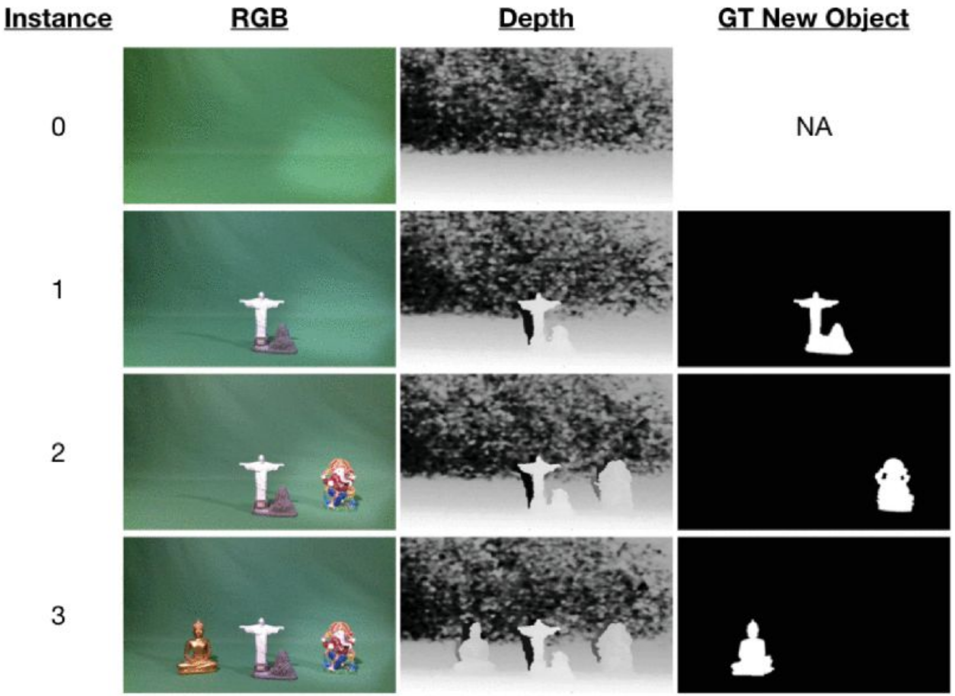
The COTS Dataset

Common Objects of a Traveling Scientist Dataset

A multipurpose RGB-D dataset that enables the evaluation of a pipelined solution

A travel-themed dataset containing 120 different instances organized in a selection of scenes

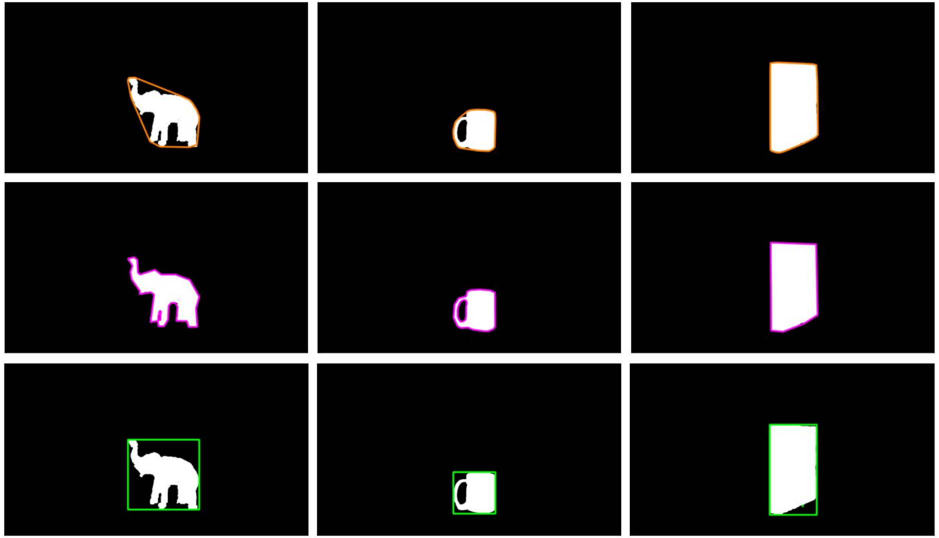
Provides RGB images, depth maps and binary masks



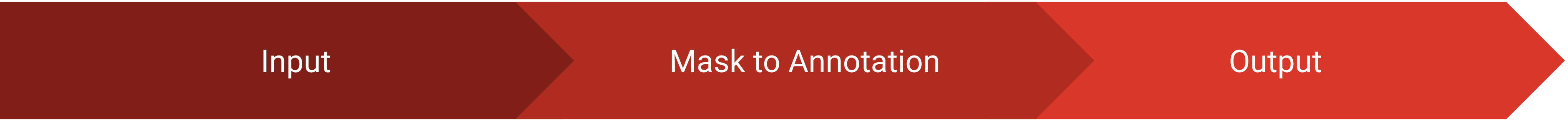
Proposed Solution

mask-to-annotation is a simple yet powerful tool for automatically generating annotations in popular computer vision formats such as COCO, YOLO, and VGG from binary masks.

By using **contour detection** algorithms and **image processing** techniques, we have created software which automates the annotation process, saving valuable time and effort.



Mask to Annotation Process

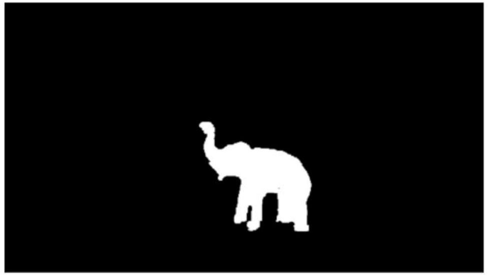


Input

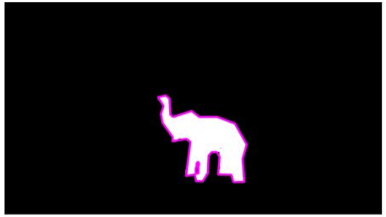
Mask to Annotation

Output

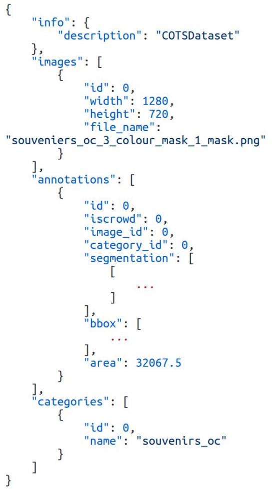
2D Mask



Contour Detection



Annotation in a format of choice

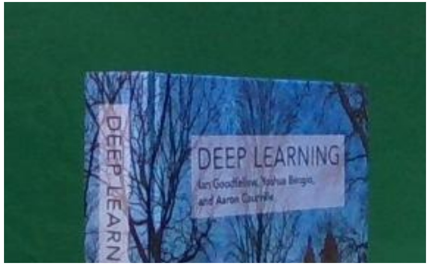


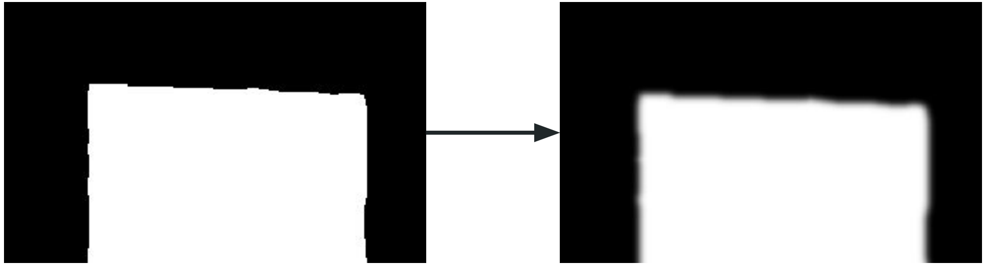
```
{
    "info": {
        "description": "COTSDataset"
    },
    "images": [
        {
            "id": 0,
            "width": 1280,
            "height": 720,
            "file_name":
"souveniers_oc_3_colour_mask_1_mask.png"
        }
    ],
    "annotations": [
        {
            "id": 0,
            "iscrowd": 0,
            "image_id": 0,
            "category_id": 0,
            "segmentation": [
                [
                    ...
                ]
            ],
            "bbox": [
                ...
            ],
            "area": 32067.5
        }
    ],
    "categories": [
        {
            "id": 0,
            "name": "souvenirs_oc"
        }
    ]
}
```

Preprocessing of Masks

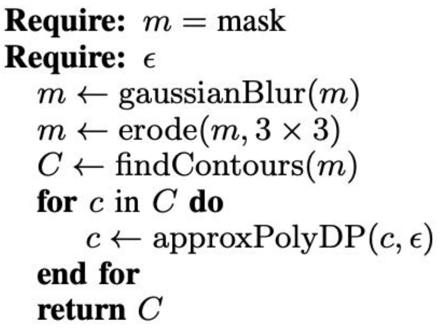
We scale the binary mask to the resolution of the original image and apply Gaussian blurring and erosion to feather the edges.

If the resolution of the binary masks is lower than that of the original image, there will be aliasing around the edges





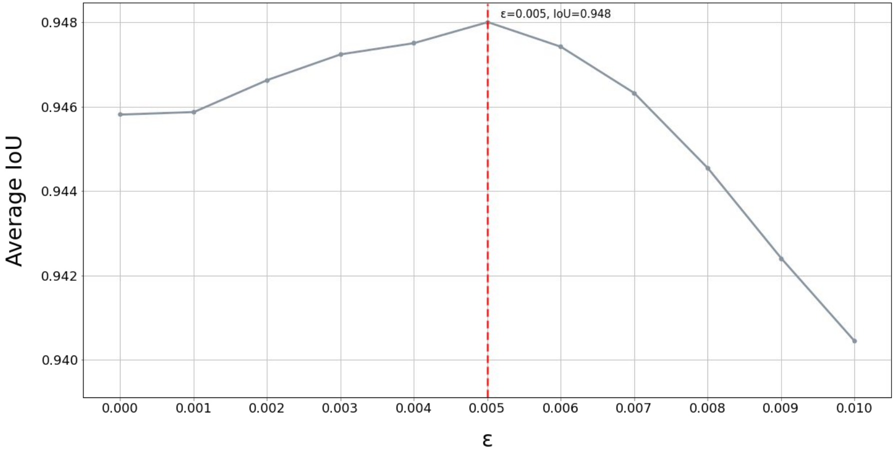
Ramer-Douglas-Peucker Algorithm



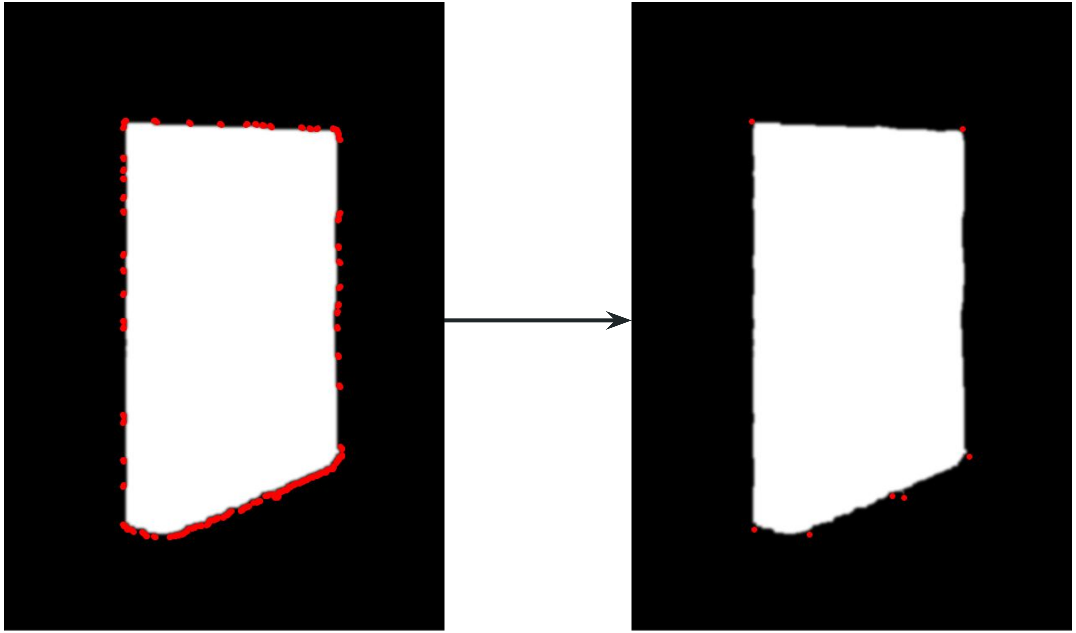
```
**Require:** $m$ = mask
**Require:** $\epsilon$
  $m \leftarrow \text{gaussianBlur}(m)$
  $m \leftarrow \text{erode}(m, 3 \times 3)$
  $C \leftarrow \text{findContours}(m)$
  **for** $c$ in $C$ **do**
      $c \leftarrow \text{approxPolyDP}(c, \epsilon)$
  **end for**
  **return** $C$
```

This polygon approximation algorithm is applied to each contour found in the mask to approximate the polygon surrounding the mask.

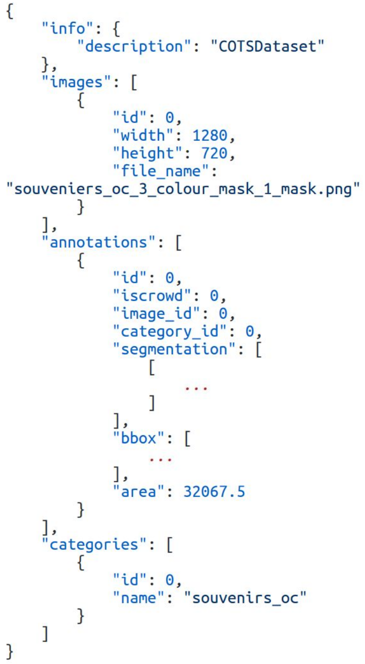
IoU between the generated binary mask and the original for different values of epsilon (higher = more approximation, less verbosity), performed on 80 images in the COTS Dataset.



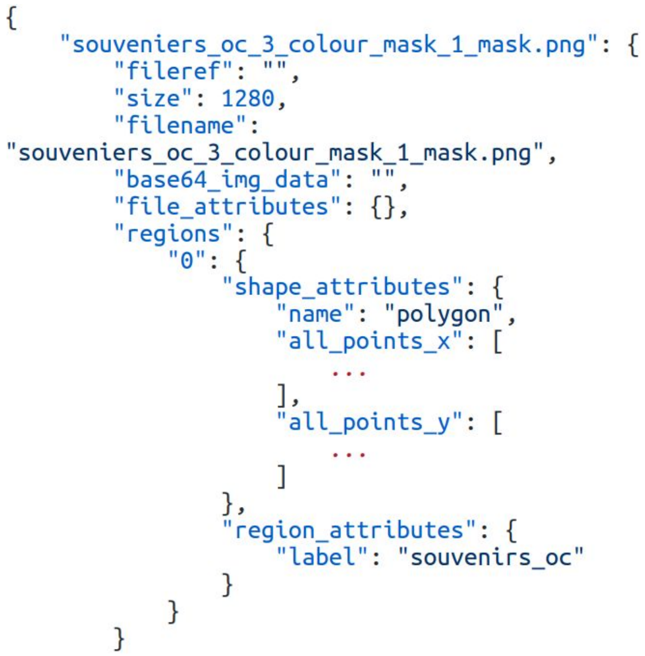
Ramer-Douglas-Peucker Algorithm (Polygon Approximation)



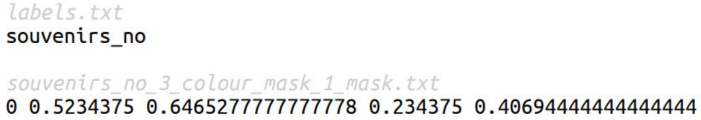
Annotation Formats (COCO, YOLO, VGG)



```
{
    "info": {
        "description": "COTSDataset"
    },
    "images": [
        {
            "id": 0,
            "width": 1280,
            "height": 720,
            "file_name":
"souveniers_oc_3_colour_mask_1_mask.png"
        }
    ],
    "annotations": [
        {
            "id": 0,
            "iscrowd": 0,
            "image_id": 0,
            "category_id": 0,
            "segmentation": [
                [
                    ...
                ]
            ],
            "bbox": [
                ...
            ],
            "area": 32067.5
        }
    ],
    "categories": [
        {
            "id": 0,
            "name": "souvenirs_oc"
        }
    ]
}
```



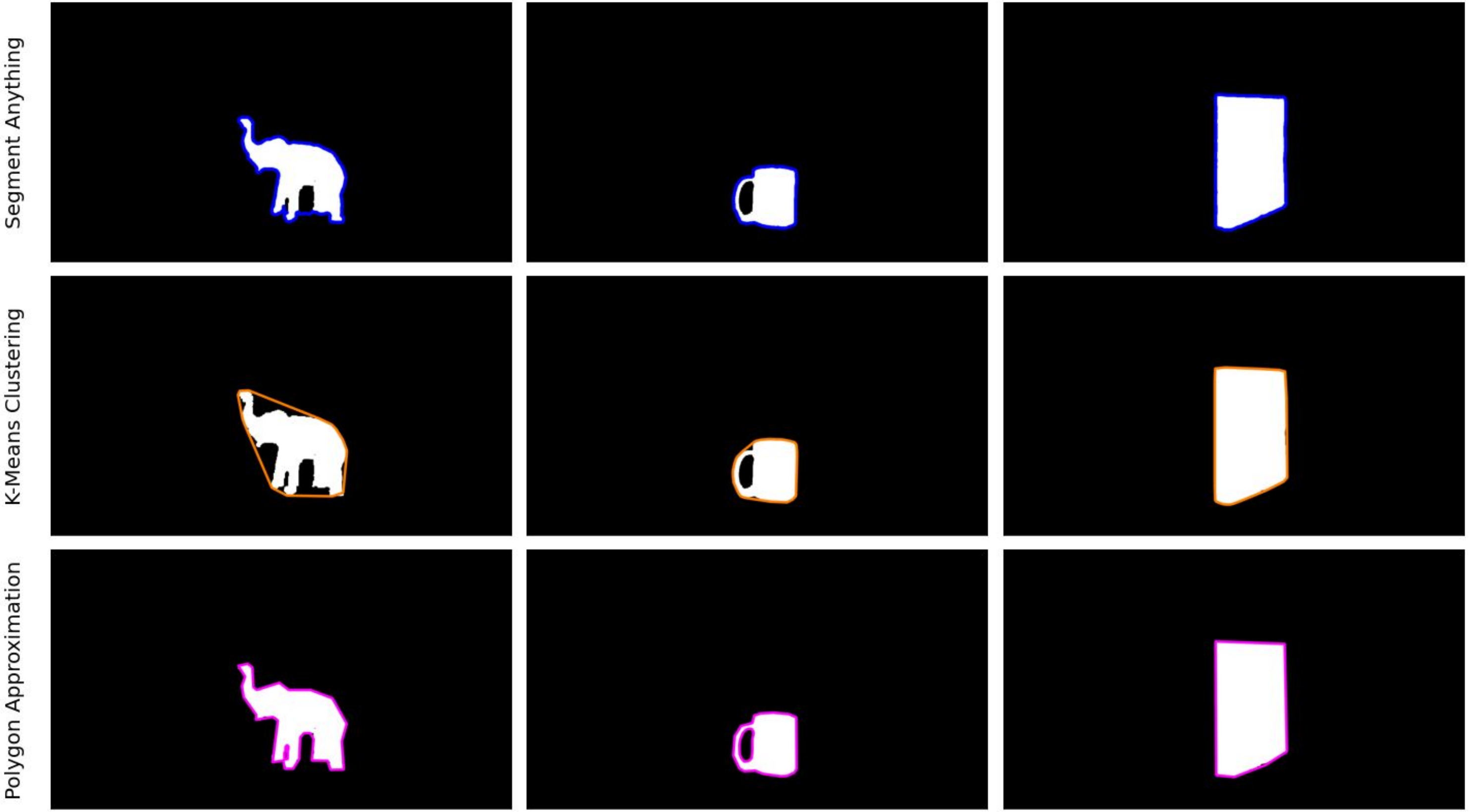
```
{
    "souveniers_oc_3_colour_mask_1_mask.png": {
        "fileref": "",
        "size": 1280,
        "filename":
"souveniers_oc_3_colour_mask_1_mask.png",
        "base64_img_data": "",
        "file_attributes": {},
        "regions": {
            "0": {
                "shape_attributes": {
                    "name": "polygon",
                    "all_points_x": [
                        ...
                    ],
                    "all_points_y": [
                        ...
                    ]
                },
                "region_attributes": {
                    "label": "souvenirs_oc"
                }
            }
        }
    }
```



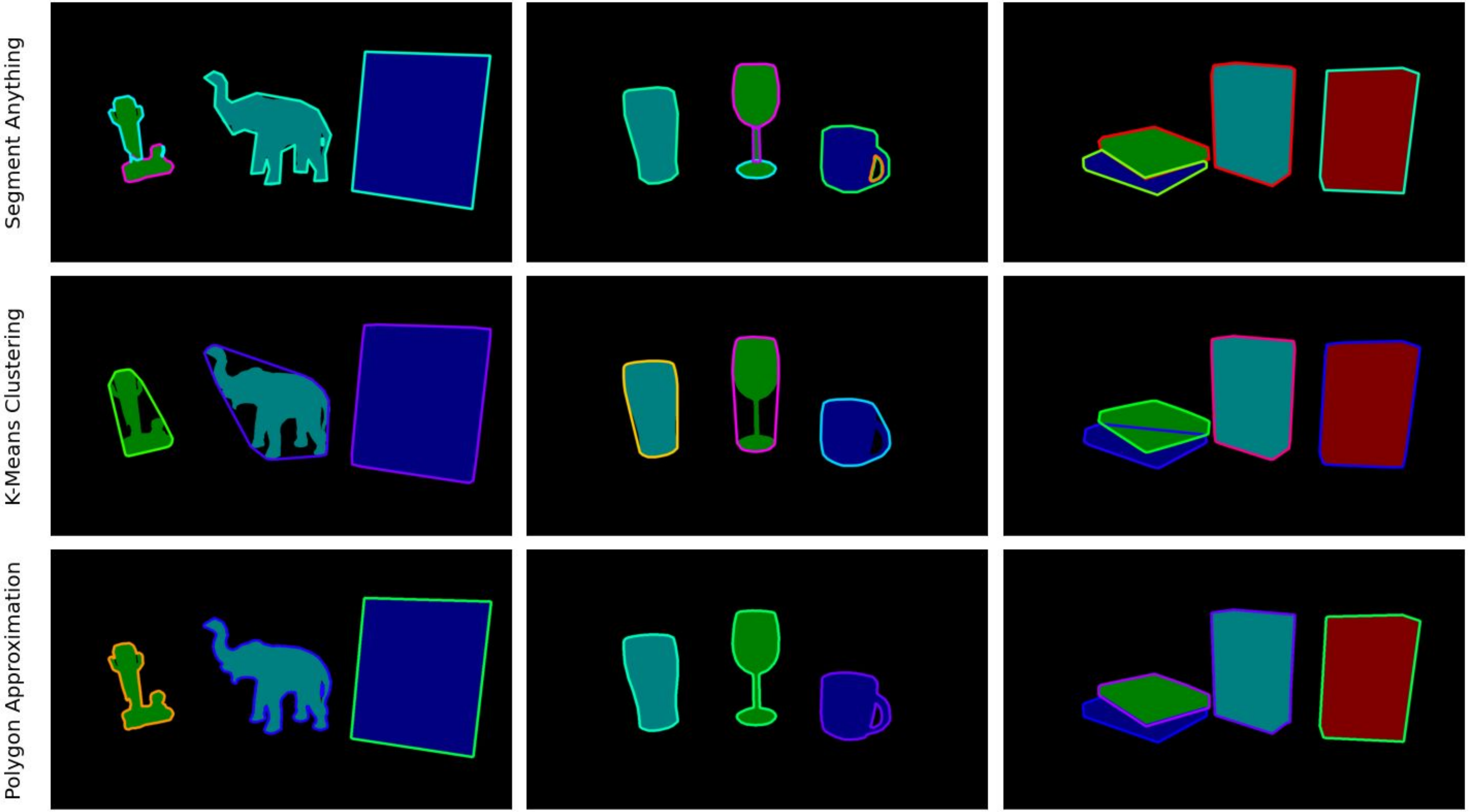
```
labels.txt
souvenirs_no

souvenirs_no_3_colour_mask_1_mask.txt
0 0.5234375 0.6465277777777778 0.234375 0.40694444444444444
```

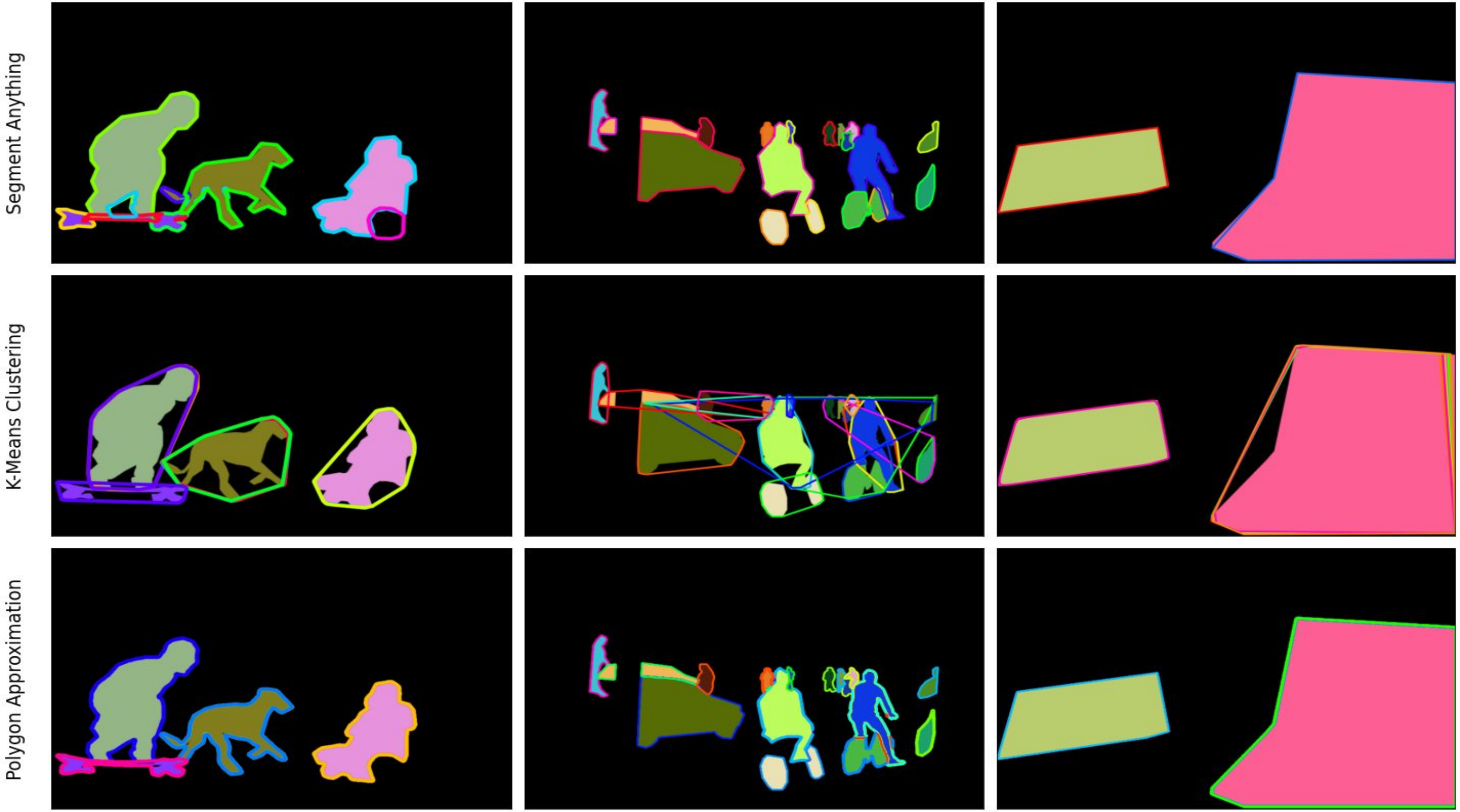
Single Object Annotation with different methods



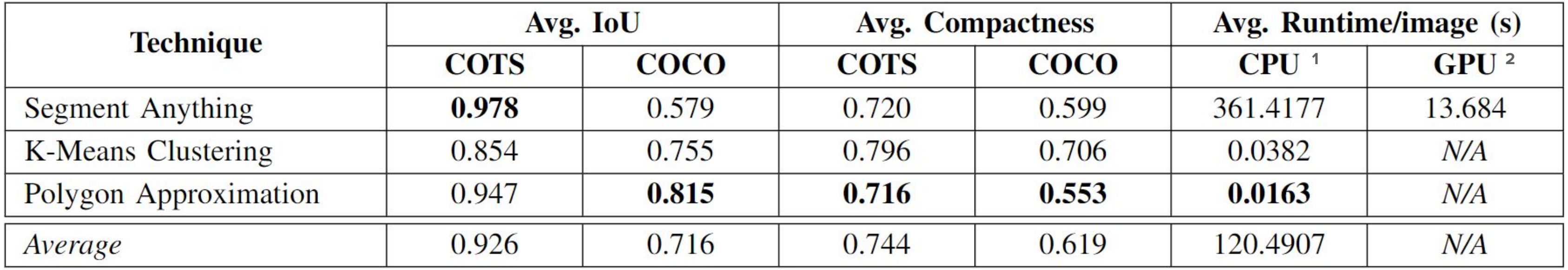
Extending it to Multiple Object Annotation



Evaluation was carried out on the COCO Dataset



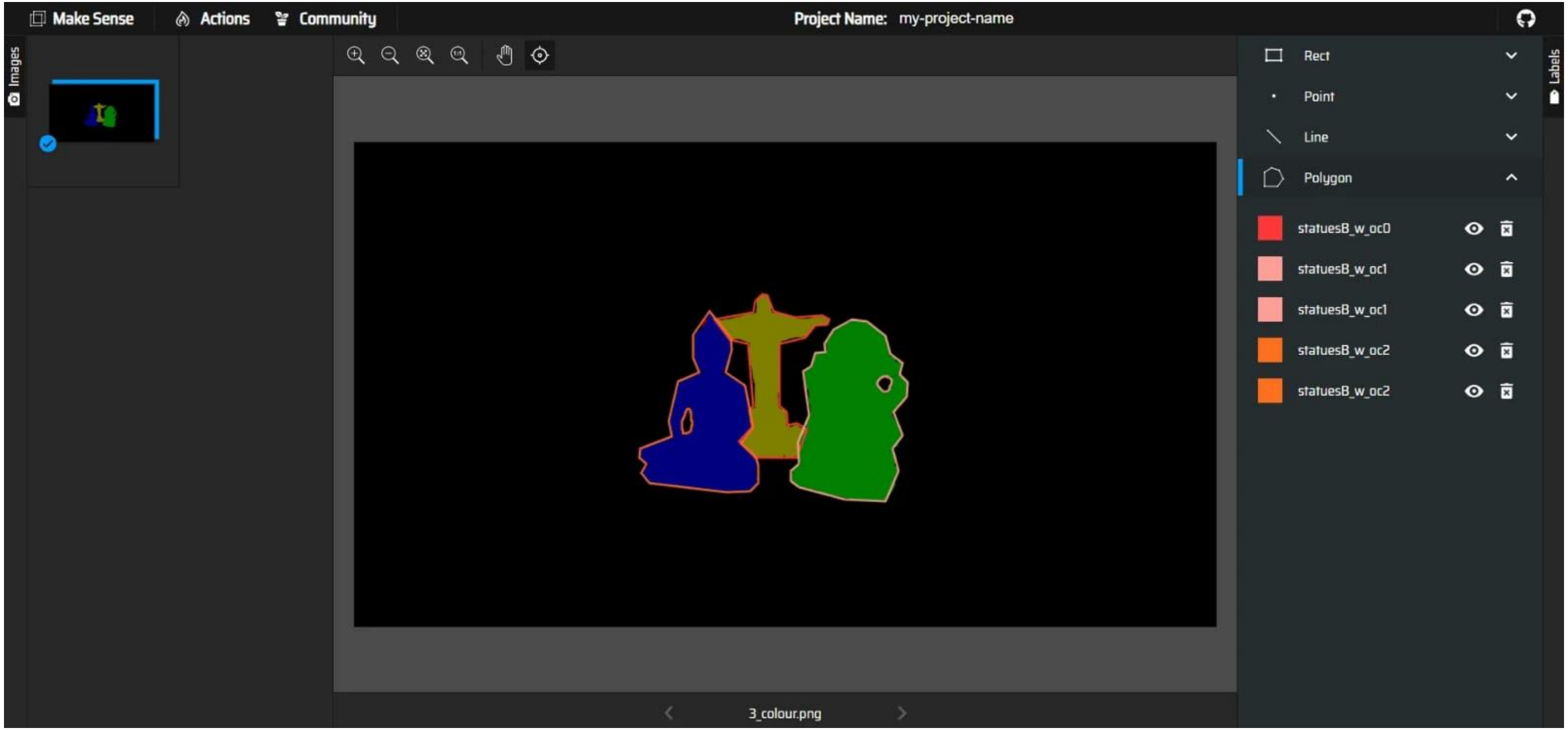
Results



Technique	Avg. IoU		Avg. Compactness		Avg. Runtime/image (s)	
	COTS	COCO	COTS	COCO	CPU [1]	GPU [2]
Segment Anything	**0.978**	0.579	0.720	0.599	361.4177	13.684
K-Means Clustering	0.854	0.755	0.796	0.706	0.0382	*N/A*
Polygon Approximation	0.947	**0.815**	**0.716**	**0.553**	**0.0163**	*N/A*
Average	0.926	0.716	0.744	0.619	120.4907	*N/A*

[1] AMD Ryzen™ 5 5625U
[2] NVIDIA GeForce RTX 2060

Testing annotation on makesense.ai



All you need in just 4 files

- annotation_helper.py
- coco.py
- yolo.py
- vgg.py

Code is **open-source** and available on **GitHub**:

github.com/dylanseychell/mask-to-annotation

✅ Demo notebooks, requirements and environment files are all provided

Thank you!

dylan.seychell@um.edu.mt
@DylanSeychell