Report 9

Contours on Detectron2 mask generated from model

Vibhav Joshi

22 June, 2020

Link to google colab:

https://colab.research.google.com/drive/1tC81waqE5uJ8oVpXjlOJ2Hgj6MvPYJFM?usp=sharing

Mask generated from Detectron2 model

```
{'instances': Instances(num_instances=2, image_height=225,
image_width=225, fields=[pred_boxes: Boxes(tensor([[ 23.9046, 137.1301,
203.4006, 189.5340],

        [  1.3053,   5.2973, 223.8218, 225.0000]], device='cuda:0')),
scores: tensor([0.9941, 0.9933], device='cuda:0'), pred_classes:
tensor([0, 1], device='cuda:0'), pred_masks: tensor([[[False, False,
False,  ..., False, False, False],

         [False, False, False,  ..., False, False, False],

         [False, False, False,  ..., False, False, False],

         ...,

         [False, False, False,  ..., False, False, False],

         [False, False, False,  ..., False, False, False],

         [False, False, False,  ..., False, False, False]],


        [[False, False, False,  ..., False, False, False],

         [False, False, False,  ..., False, False, False],

         [False, False, False,  ..., False, False, False],

         ...,

         [False, False, False,  ..., False, False, False],

         [False, False, False,  ..., False, False, False],

         [False, False, False,  ..., False, False, False]]],
device='cuda:0')])}
```

The function to process from mask to contours

- The binary mask is given to create_sub_mask function which creates its mask for processing in numpy data
- `create_sub_mask_annotation(sub_mask,label) this functions converts the mask to contours`

```
def create_sub_masks(mask_image,label):

    width, height = mask_image.size

    # Initialize a dictionary of sub-masks indexed by RGB colors
    sub_masks = {}
    for x in range(width):
        for y in range(height):

            pixel = mask_image.getpixel((x,y))[:3]


            # If the pixel is not black...
            if pixel != (0, 0, 0):
                # Check to see if we've created a sub-mask...
                pixel_label = label
                sub_mask = sub_masks.get(pixel_label)

                if sub_mask is None:

                   # Create a sub-mask (one bit per pixel) and add to the
dictionary

                    # Note: we add 1 pixel of padding in each direction

                    # because the contours module doesn't handle cases

                    # where pixels bleed to the edge of the image

                    sub_masks[pixel_label] = Image.new('1', (width+2,
height+2))


                # Set the pixel value to 1 (default is 0), accounting for
padding

                sub_masks[pixel_label].putpixel((x+1, y+1), 1)


    return sub_masks

def create_sub_mask_annotation(sub_mask,label):


    contours = measure.find_contours(sub_mask, 0.5,
positive_orientation='low')


    segmentations = []

    polygons = []

    for contour in contours:


        for i in range(len(contour)):
            row, col = contour[i]

            contour[i] = (col - 1, row - 1)



        poly = Polygon(contour)

        poly = poly.simplify(1.0, preserve_topology=False)

        polygons.append(poly)

    classes = ['keyboard', 'laptop']



    multi_poly = MultiPolygon(polygons)

    x, y, max_x, max_y = multi_poly.bounds

    width = max_x - x

    height = max_y - y

    bbox = [x, y, width, height]

    #print(bbox)

    area = multi_poly.area

    print("countour:",contours)

    print("area",area)

    print("label",classes[label])
```