Report 8

Preparing detectron2 data for Labelbox Dental data

Vibhav Joshi

10th June, 2020

Label box Data

Link of Google colab:

https://colab.research.google.com/drive/16bnoHyT0CvV7oPFswWFDWNNPor9keGaH?usp=sharing

https://colab.research.google.com/drive/135qovEQU-67ziRLg6cf_ajW_Pd59uwne?usp=sharing

https://colab.research.google.com/drive/1t3U20_8bOAZQIPRPL-jbgQNI49hWcTya?usp=sharing

{"ID":"ckami99bnfxm90751hmb82fm1","DataRow ID":"ckamg4ri4xayf0bltdwef6zvb","Labeled Data":"https://storage.labelbox.com/ck7ede8c82pw9088993xydstx%2F08c22d1c-0b08-2ff3-442d-0d934df21c04-12.jpg?Expires=1594019666527&KeyName=labelbox-assets-key-1&Signature=FLkZF2oomZAZturBkWCVkcqPZDI" ,
"Label":{"objects":[{"featureId":"ckamhum0y0kvr0zc2bimrymoi","schemaId":"ckamhk8840kgj0zc2pl3foj9g","title":"Upper_Right_Central_Incisor","value":"upper_right_central_incisor","color":"#FF0000","instanceURI":"https://api.labelbox.com/masks/feature/ckamhum0y0kvr0zc2bimrymoi?token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOiJjazdlZGU4Y3AwajIyMDk0OGFscWR6OXZjIiwib3JnYW5pemF0aW9uSWQiOiJjazdlZGU4YzgycHc5MDg4OTkzeHlkc3R4IiwiaWF0IjoxNTkyODEwMDY2LCJleHAiOjE1OTU0MDIwNjZ9.rt1tYSLpPzPeHqavP4ioFXJBoOIjhTUkmhqr9vUyH_Y"},

- Studied the data and the mask needed to be transformed
- The mask for each class

- Total classes classes = ['upper_right_central_incisor', 'upper_left_central_incisor','upper_right_lateral_incisor','upper_left_lateral_incisor','upper_right_canine','upper_left_canine','lower_right_central_incisor','lower_left_central_incisor_', 'lower_right_lateral_incisor','lower_left_lateral_incisor','lower_right_canine','lower_right_first_premolar_surface','lower_right_second_premolar_surface','upper_right_central_incisor',

  'upper_left_central_incisor','upper_right_lateral_incisor','upper_left_lateral_incisor','upper_right_canine','upper_left_canine','lower_right_central_incisor','lower_left_central_incisor_',

  'Lower_right_lateral_incisor','lower_right_canine','lower_right_first_premolar_surface','lower_right_second_premolar_surface','background','background','lower_left_lateral_incisor']

Functions to transform data to standard detectron2 dataset

```
def create_sub_masks(mask_image):

    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):
            # Get the RGB values of the pixel
            pixel = mask_image.getpixel((x,y))[:3]
```

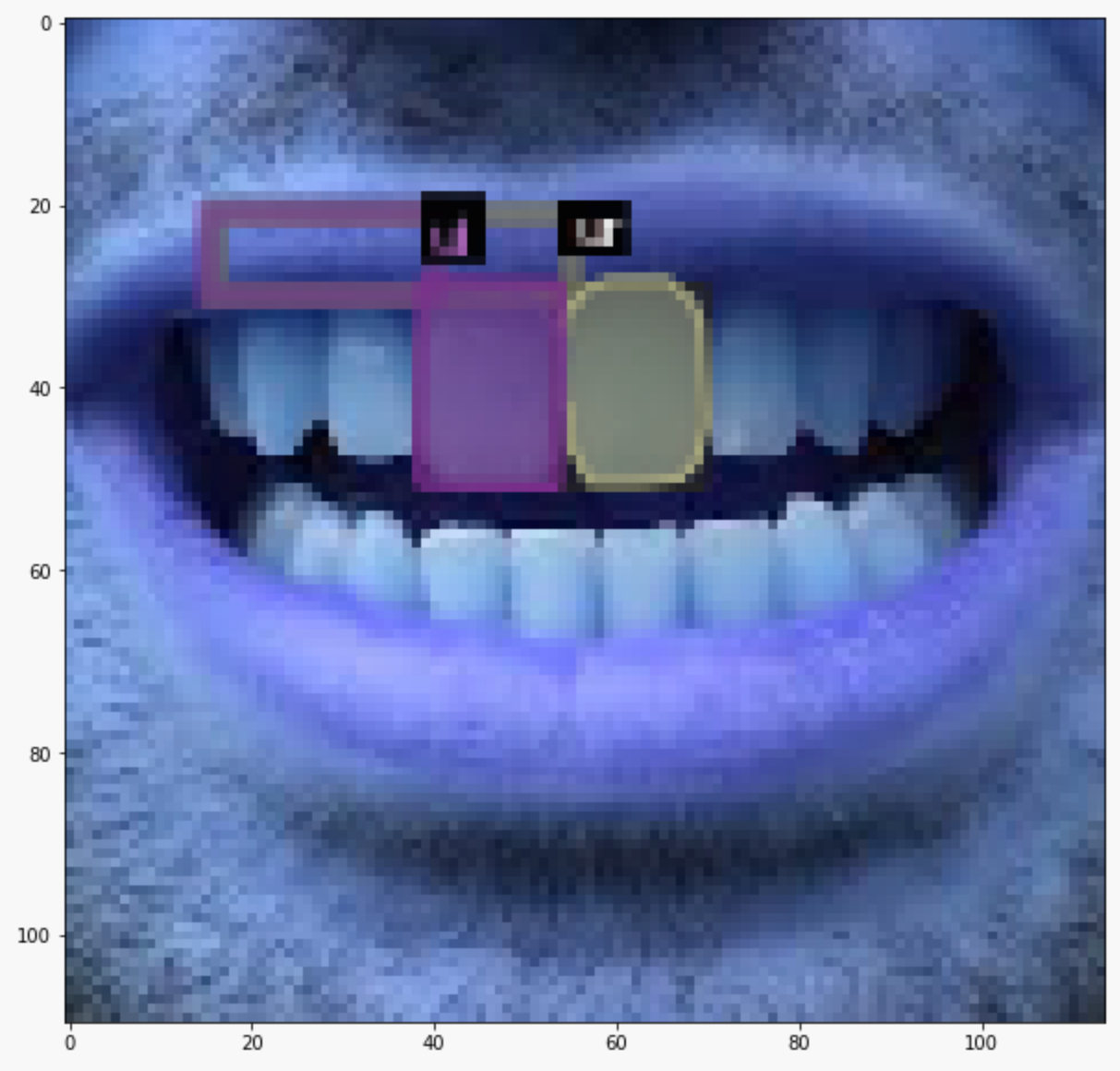
```
            # If the pixel is not black...
            if pixel != (0, 0, 0):
                # Check to see if we've created a sub-mask...
                pixel_str = str(pixel)
                sub_mask = sub_masks.get(pixel_str)
                if sub_mask is None:
                   # Create a 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_str] = Image.new('1', (width+2, height+2))

                # Set the pixel value to 1 (default is 0), accounting for padding
                sub_masks[pixel_str].putpixel((x+1, y+1), 1)

    return sub_masks
```

- The function takes mask images and sacn pixel by pixel for white rgb value which is mask of particular class from 28 total
- It returns a dictionary of maks generated

```
def create_sub_mask_annotation(sub_mask):

    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)
        segmentation =
np.array(poly.exterior.coords).ravel().tolist()
        segmentations.append(segmentation)

    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]
    area = multi_poly.area


    return segmentations,bbox,area
```

- This Function takes mask from previous function and plots contours on it
- Thus these contours are transformed in polygon and segmentation which is needed for standard detectron2 dataset


0
20
40
60
80
100
0
20
40
60
80
100