Meeting #5

10-7-21

Michael Lee

Deliverables

- Try to display 3D brain with coordinates

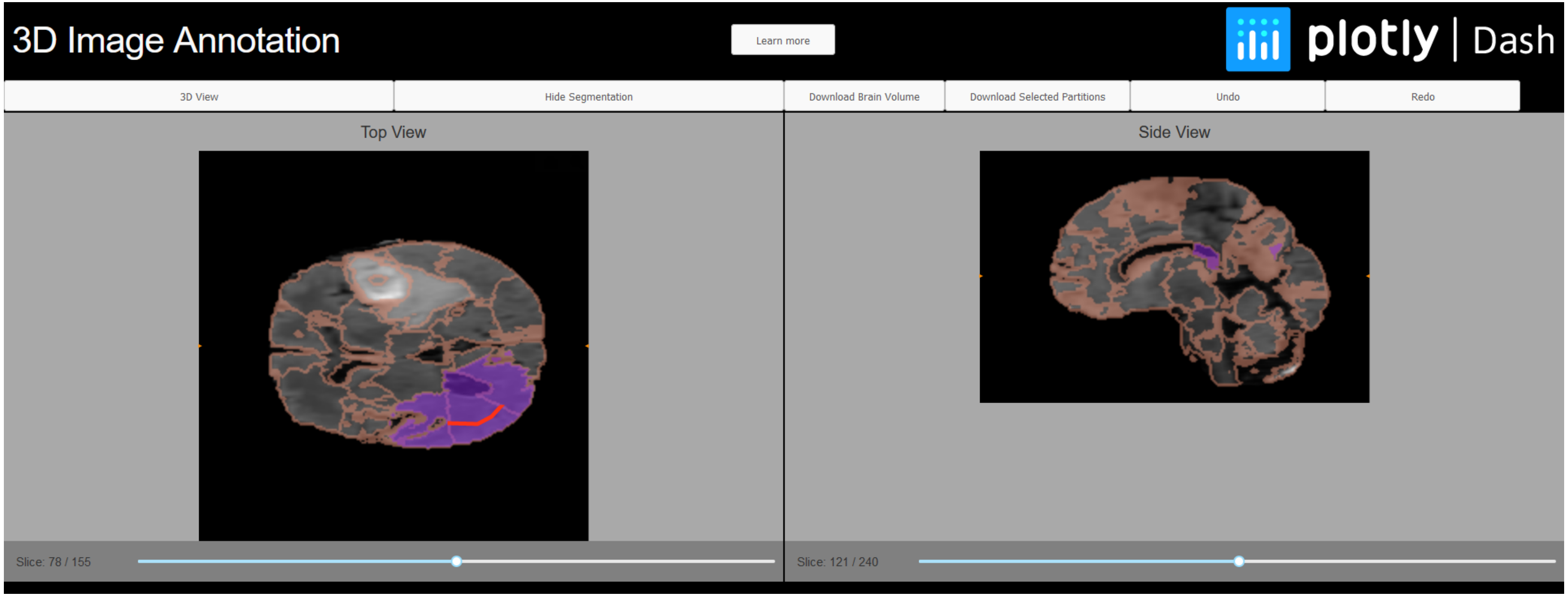
Methodology and Learnings

- Look at 3D brain visuals in plotly dash

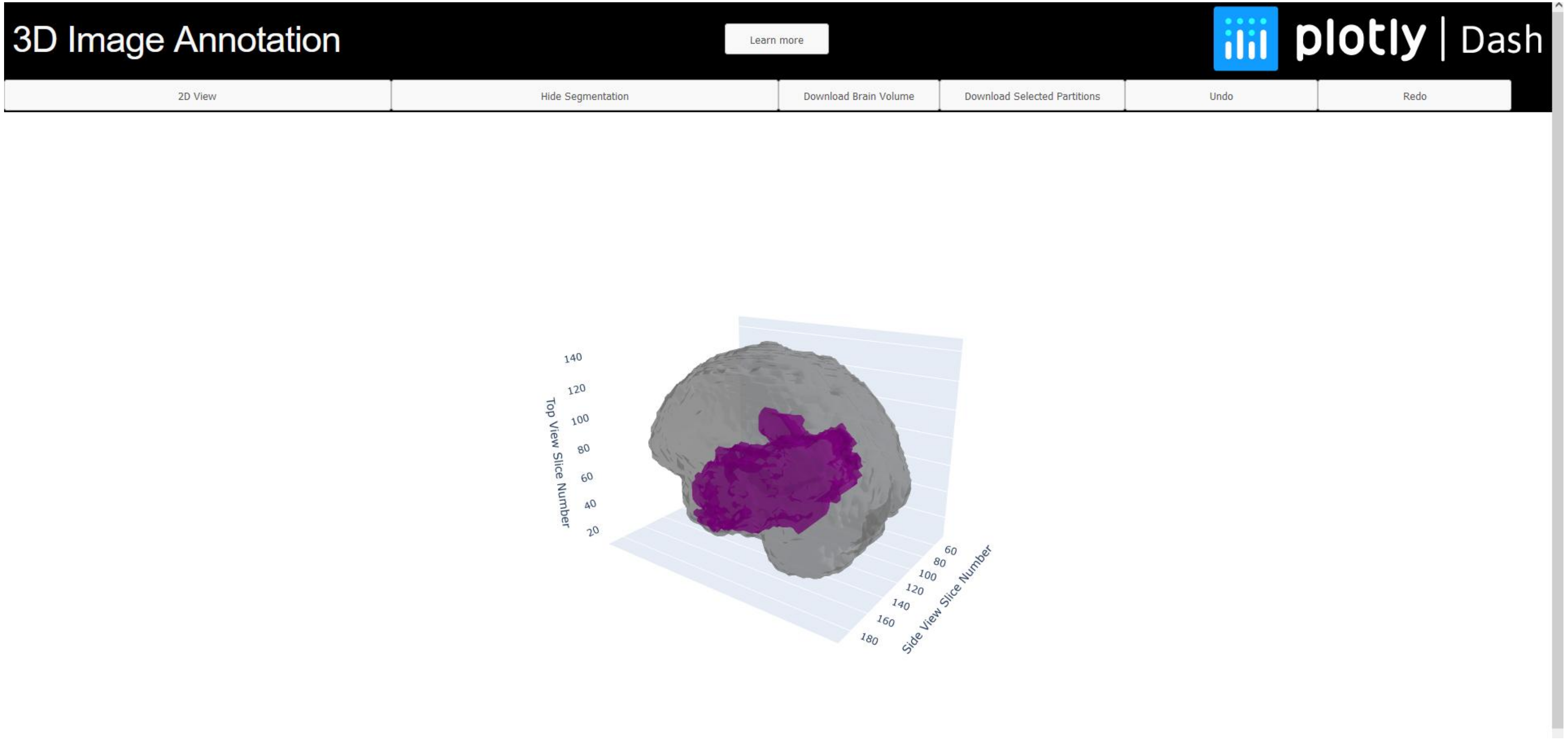
- https://github.com/plotly/dash-sample-apps/tree/main/apps/dash-3d-image-partitioning

- Tensors

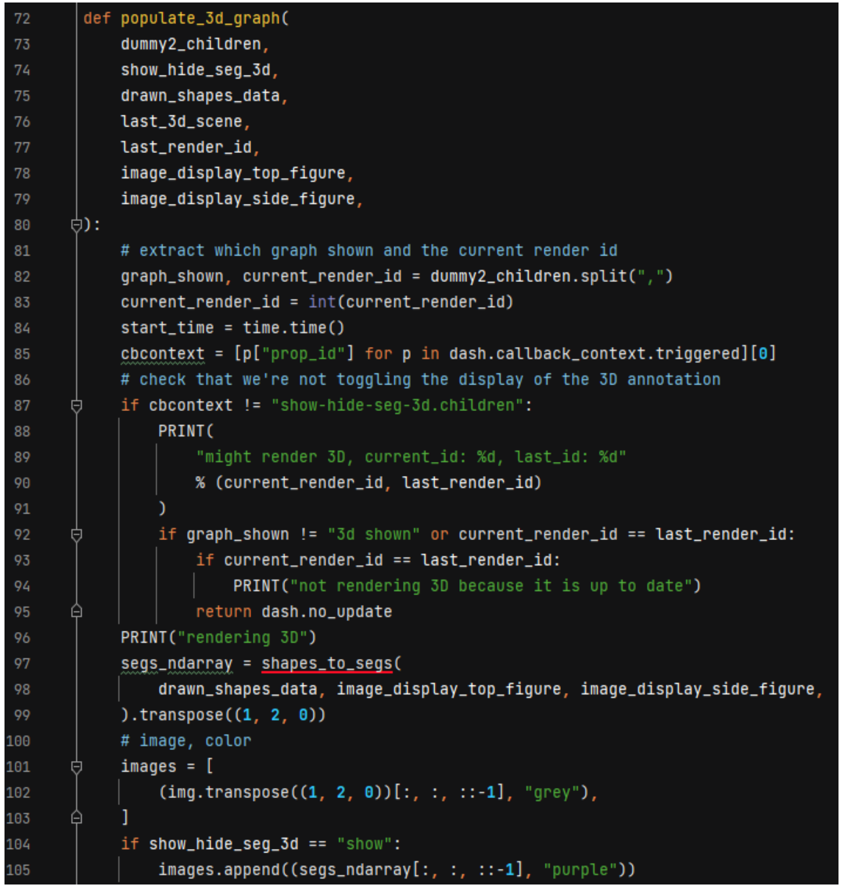
Results


3D Image Annotation
Learn more
plotly | Dash
3D View
Hide Segmentation
Download Brain Volume
Download Selected Partitions
Undo
Redo
Top View
Side View
Slice: 78 / 155
Slice: 121 / 240

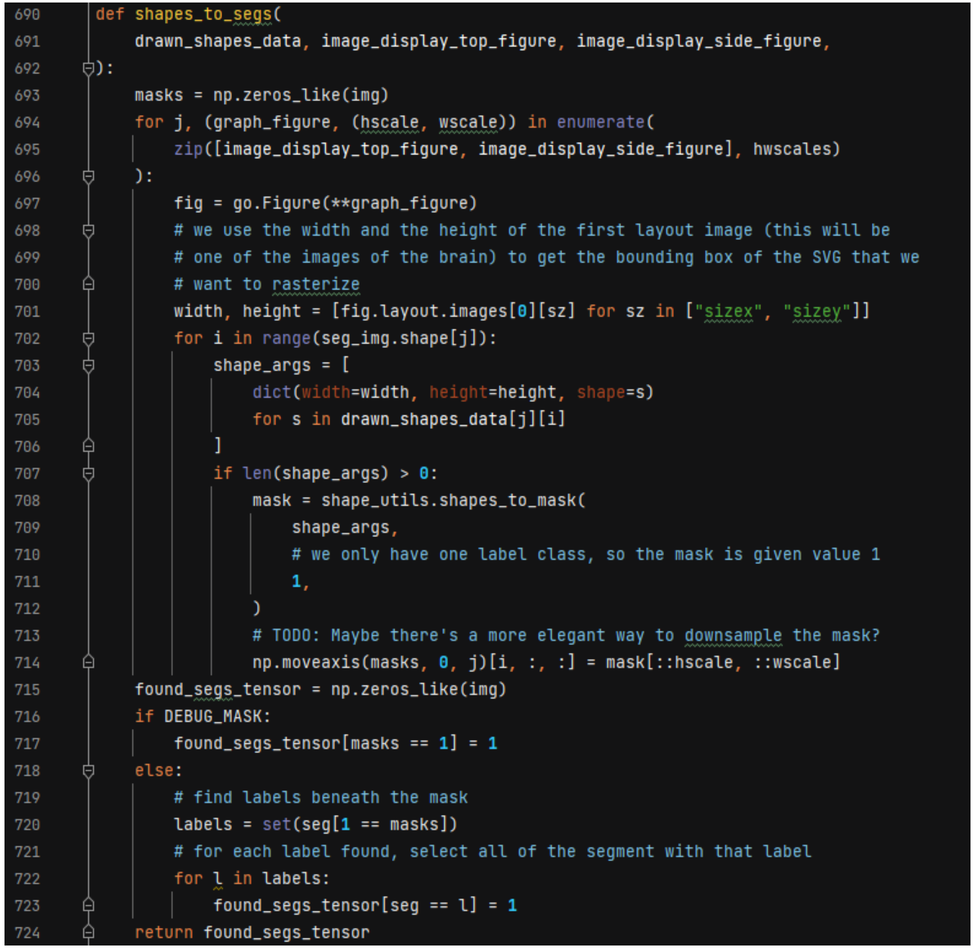
Results


3D Image Annotation
Learn more
plotly | Dash
2D View
Hide Segmentation
Download Brain Volume
Download Selected Partitions
Undo
Redo
Top View Slice Number
140
120
100
80
60
40
20
Side View Slice Number
60
80
100
120
140
160
180

Results



```python
def populate_3d_graph(
    dummy2_children,
    show_hide_seg_3d,
    drawn_shapes_data,
    last_3d_scene,
    last_render_id,
    image_display_top_figure,
    image_display_side_figure,
):
    # extract which graph shown and the current render id
    graph_shown, current_render_id = dummy2_children.split(",")
    current_render_id = int(current_render_id)
    start_time = time.time()
    cbcontext = [p["prop_id"] for p in dash.callback_context.triggered][0]
    # check that we're not toggling the display of the 3D annotation
    if cbcontext != "show-hide-seg-3d.children":
        PRINT(
            "might render 3D, current_id: %d, last_id: %d"
            % (current_render_id, last_render_id)
        )
        if graph_shown != "3d shown" or current_render_id == last_render_id:
            if current_render_id == last_render_id:
                PRINT("not rendering 3D because it is up to date")
            return dash.no_update
    PRINT("rendering 3D")
    segs_ndarray = shapes_to_segs(
        drawn_shapes_data, image_display_top_figure, image_display_side_figure,
    ).transpose((1, 2, 0))
    # image, color
    images = [
        (img.transpose((1, 2, 0))[:, :, ::-1], "grey"),
    ]
    if show_hide_seg_3d == "show":
        images.append((segs_ndarray[:, :, ::-1], "purple"))
```



```python
def shapes_to_segs(
    drawn_shapes_data, image_display_top_figure, image_display_side_figure,
):
    masks = np.zeros_like(img)
    for j, (graph_figure, (hscale, wscale)) in enumerate(
        zip([image_display_top_figure, image_display_side_figure], hwscales)
    ):
        fig = go.Figure(**graph_figure)
        # we use the width and the height of the first layout image (this will be
        # one of the images of the brain) to get the bounding box of the SVG that we
        # want to rasterize
        width, height = [fig.layout.images[0][sz] for sz in ["sizex", "sizey"]]
        for i in range(seg_img.shape[j]):
            shape_args = [
                dict(width=width, height=height, shape=s)
                for s in drawn_shapes_data[j][i]
            ]
            if len(shape_args) > 0:
                mask = shape_utils.shapes_to_mask(
                    shape_args,
                    # we only have one label class, so the mask is given value 1
                    1,
                )
                # TODO: Maybe there's a more elegant way to downsample the mask?
                np.moveaxis(masks, 0, j)[i, :, :] = mask[::hscale, ::wscale]
    found_segs_tensor = np.zeros_like(img)
    if DEBUG_MASK:
        found_segs_tensor[masks == 1] = 1
    else:
        # find labels beneath the mask
        labels = set(seg[1 == masks])
        # for each label found, select all of the segment with that label
        for l in labels:
            found_segs_tensor[seg == l] = 1
    return found_segs_tensor
```