ParallelDots Recruitment Assignment

This assignment will require you to use SAM2 model by Facebook AI to do Zero shot product detection. To get started, install SAM2 using instrictions here : https://github.com/facebookresearch/segment-anything-2

Once installed, you should be able to import SAM2 like so :

{Please use the tiny model}, it will be able to work on light GPUs and even CPU. The code we provide will assume you have a GPU available.

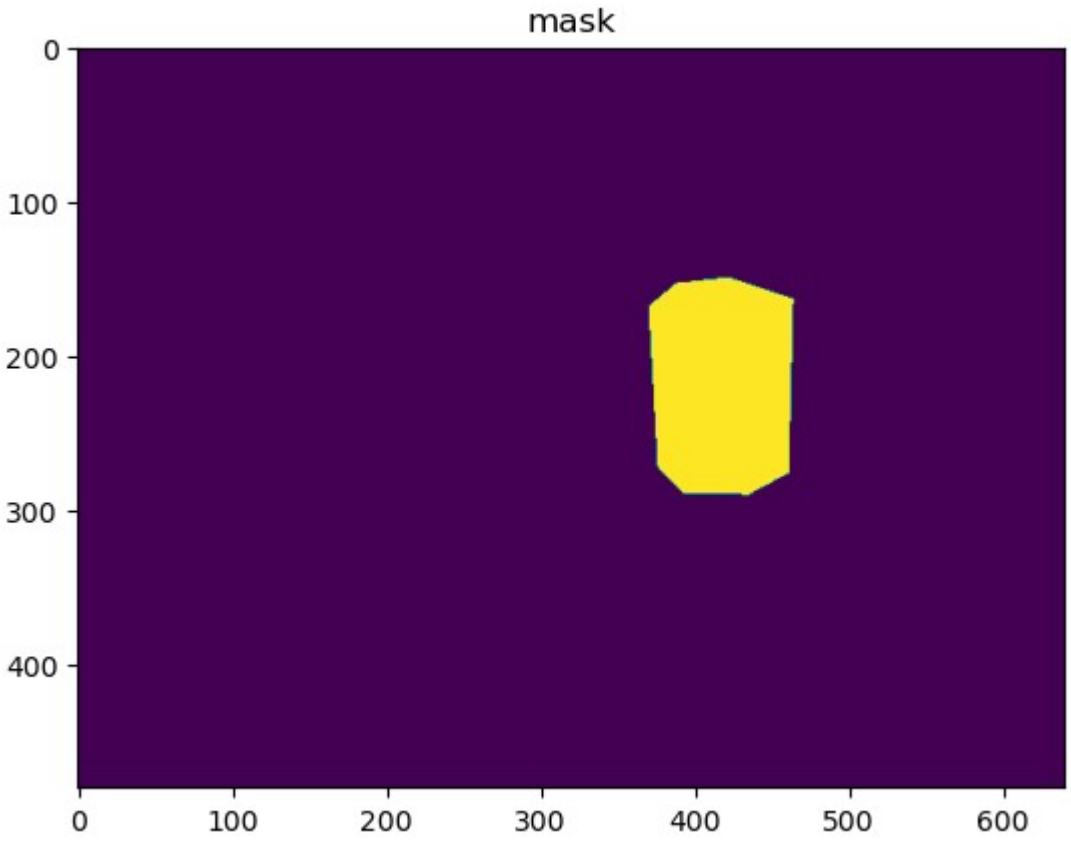
In [1]:
```
from sam2_extract import *
```

Now you will require to download CMU dataset from : http://www.cs.cmu.edu/~ehsiao/3drecognition/CMU10_3D.zip You just need to concern yourself with data_2D directory in dataset. Each directory has multiple images of a product with a mask of the product provided for each image. What you need to do is to use the mask of the first image and the first image itself to setup SAM2 to detect the product in other images and calculate its accuracy for each product. We are providing code to handle the SAM2 part and the applicant is thus required to work on image file loading/unloading and calculating accuracy metrics.

So for example, in folder data_2d , you have multiple images of product can chowder. In this only the first image and its mask { can_chowder_000001.jpg and can_chowder_000001_1_gt.png } will be used to setup SAM2 model for this product. One will require x1,x2,y1,y2 of the object in this image to share data with SAM2 for detection. You will be expected to implement the function to get these points from the setup/first image yourself using provided mask.

In [2]:
```
firstimgpath = '.\\CMU10_3D\\CMU10_3D\\data_2D\\can_chowder_000001.jpg'
firstimgmaskpath = '.\\CMU10_3D\\CMU10_3D\\data_2D\\can_chowder_000001_1_gt.png'
[xmin,xmax,ymin,ymax] = process_img_png_mask(firstimgpath,firstimgmaskpath,visualiz
```


full image
EAGLE
0
100
200
300
400
0
100
200
300
400
500
600


mask
0
100
200
300
400
0
100
200
300
400
500
600

One has to set the above image and mask in SAM2 to detect can_chowder in all other images. So for example, we provide code to detect the product in can_chowder_000002.jpg below. This is how the image looks like ::

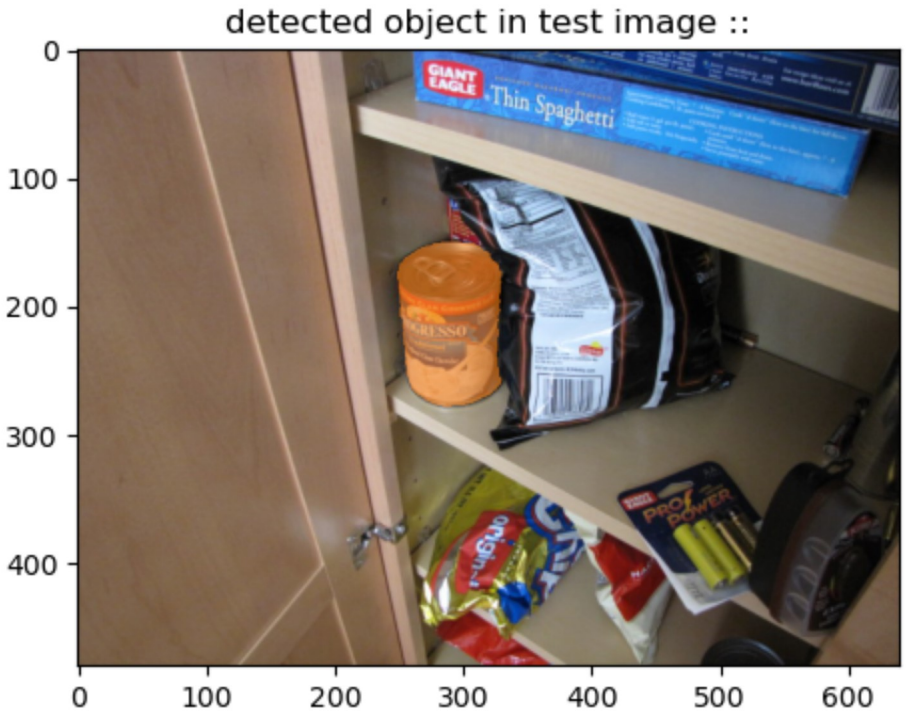
In [3]:
```
secondimgpath =  '.\\CMU10_3D\\CMU10_3D\\data_2D\\can_chowder_000002.jpg'
secondimg = Image.open(secondimgpath)
plt.imshow(secondimg)
plt.show()
```

So now you can call the function track_item_boxes to detect the product in can_chowder_000002.jpg . The 1 passed to this function is a dummy number to denote the object, the output mask will have the same object number. See how the object gets tracked automatically in this case ::

In [4]:
```
op = track_item_boxes(firstimgpath,secondimgpath,[([xmin,xmax,ymin,ymax],1)],visual
```

```
frame loading (JPEG): 100%|
████████████████████████████████████████████████████████████| 2/2 [00:00<00:00, 1
3.29it/s]
C:\projects\localpipinstalls\segment-anything-2\sam2\modeling\backbones\hieradet.p
y:68: UserWarning: 1Torch was not compiled with flash attention. (Triggered internal
ly at C:\cb\pytorch_1000000000000\work\aten\src\ATen\native\transformers\cuda\sdp_ut
ils.cpp:555.)
  x = F.scaled_dot_product_attention(
C:\projects\localpipinstalls\segment-anything-2\sam2\modeling\sam\transformer.py:27
0: UserWarning: Memory efficient kernel not used because: (Triggered internally at
C:\cb\pytorch_1000000000000\work\aten\src\ATen\native\transformers\cuda\sdp_utils.cp
p:718.)
  out = F.scaled_dot_product_attention(q, k, v, dropout_p=dropout_p)
C:\projects\localpipinstalls\segment-anything-2\sam2\modeling\sam\transformer.py:27
0: UserWarning: Memory Efficient attention has been runtime disabled. (Triggered int
ernally at C:\cb\pytorch_1000000000000\work\aten\src\ATen/native/transformers/sdp_ut
ils_cpp.h:495.)
  out = F.scaled_dot_product_attention(q, k, v, dropout_p=dropout_p)
C:\projects\localpipinstalls\segment-anything-2\sam2\modeling\sam\transformer.py:27
0: UserWarning: Flash attention kernel not used because: (Triggered internally at C:
\cb\pytorch_1000000000000\work\aten\src\ATen\native\transformers\cuda\sdp_utils.cp
p:720.)
  out = F.scaled_dot_product_attention(q, k, v, dropout_p=dropout_p)
C:\projects\localpipinstalls\segment-anything-2\sam2\modeling\sam\transformer.py:27
0: UserWarning: CuDNN attention kernel not used because: (Triggered internally at C:
\cb\pytorch_1000000000000\work\aten\src\ATen\native\transformers\cuda\sdp_utils.cp
p:722.)
  out = F.scaled_dot_product_attention(q, k, v, dropout_p=dropout_p)
C:\projects\localpipinstalls\segment-anything-2\sam2\modeling\sam\transformer.py:27
0: UserWarning: The CuDNN backend needs to be enabled by setting the enviornment var
iable`TORCH_CUDNN_SDPA_ENABLED=1` (Triggered internally at C:\cb\pytorch_10000000000
00\work\aten\src\ATen\native\transformers\cuda\sdp_utils.cpp:497.)
  out = F.scaled_dot_product_attention(q, k, v, dropout_p=dropout_p)
C:\Users\macoa\miniconda3\envs\mluni\Lib\site-packages\torch\nn\modules\module.py:15
62: UserWarning: Flash Attention kernel failed due to: No available kernel. Aborting
execution.
Falling back to all available kernels for scaled_dot_product_attention (which may ha
ve a slower speed).
  return forward_call(*args, **kwargs)
C:\projects\localpipinstalls\segment-anything-2\sam2\sam2_video_predictor.py:873: Us
erWarning: cannot import name '_C' from 'sam2' (C:\projects\localpipinstalls\segmen
t-anything-2\sam2\__init__.py)

Skipping the post-processing step due to the error above. You can still use SAM 2 an
d it's OK to ignore the error above, although some post-processing functionality may
be limited (which doesn't affect the results in most cases; see https://github.com/f
acebookresearch/segment-anything-2/blob/main/INSTALL.md).
  pred_masks_gpu = fill_holes_in_mask_scores(
propagate in video: 100%|
████████████████████████████████████████████████████████████| 2/2 [00:00<00:00,
7.13it/s]
```



Now you can see the output mask for the image here ::

In [8]:
```
output_masks = op[1] # Mask for output image is always on op[1] for this example
print(output_masks)
```

```
{1: array([[[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]]])}
```

The 1 in the key is the dummy number you shared to denote the object {1 above}, so you can get the mask out

In [9]:
```
relevant_mask = output_masks[1]
print(relevant_mask)
```

```
[[[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]]]
```

Problem statements ::

1. Using the first image of each object in data_2d folder and its corresponding mask {"can_chowder_000001.jpg", "can_chowder_000001_1_gt.png"} {"can_soymilk_000001.jpg" , "can_soymilk_000001_1_gt.png"} {"can_tomatosoup_000001.jpg" , "can_tomatosoup_000001_1_gt.png"} {"carton_oj_000001.jpg", "carton_oj_000001_1_gt.png"} {"carton_soymilk_000001.jpg","carton_soymilk_000001_1_gt.png"} {"diet_coke_000001.jpg" "diet_coke_000001_1_gt.png"} {"hc_potroastsoup_000001.jpg" "hc_potroastsoup_000001_1_gt.png"} {"juicebox_000001.jpg" "juicebox_000001_1_gt.png"} {"rice_tuscan_000001.jpg" "rice_tuscan_000001_1_gt.png"} {"ricepilaf_000001.jpg" "ricepilaf_000001_1_gt.png"} detect the objects in all other images of the object. That is {"can_chowder_000001.jpg", "can_chowder_000001_1_gt.png"} should be used to setup SAM2 to locate can_chowder in all other images of can_chowder . Similarly first image/mask pair of juicebox to locate juicebox in all other images of type juicebox_{number}.jpg .
2. You get masks as output, use a closely fitting bounding box as the final output corresponding to output mask you receive
3. Install pycocotools and use the boxes you get in step 2 from the mask like output of SAM2 as prediction and boxes you have as mask for each image { so groound truth for can_chowder_000002.jpg is box around mask in can_chowder_000002_1_gt.png while prediction is box around the mask relevant_mask calculated by SAM2 above } as ground truth and calculate object detection preformance for each product individually.

As promised, the code for the function track_item_boxes which detects objects zero shot using SAM2 is provided below, you are allowed to use this, but the rest of the code should

be written by the applicant ::

In [1]:
```
import torch
from sam2.build_sam import build_sam2
from sam2.automatic_mask_generator import SAM2AutomaticMaskGenerator
from sam2.sam2_image_predictor import SAM2ImagePredictor
from sam2.build_sam import build_sam2_video_predictor
from PIL import Image, ImageOps
import numpy as np
import matplotlib.pyplot as plt
import os,glob,shutil
import matplotlib.patches as patches

checkpoint = "./sam2_hiera_tiny.pt"
model_cfg = "sam2_hiera_t.yaml"
predictor_prompt = SAM2ImagePredictor(build_sam2(model_cfg, checkpoint))
sam2 = build_sam2(model_cfg, checkpoint, device='cuda', apply_postprocessing=False)
mask_generator = SAM2AutomaticMaskGenerator(sam2)
predictor_vid = build_sam2_video_predictor(model_cfg, checkpoint, device='cuda')

tempfolder = "./tempdir"

def create_if_not_exists(dirname):
    if not os.path.exists(dirname):
        os.mkdir(dirname)

def cleardir(tempfolder):
    filepaths = glob.glob(tempfolder+"/*")
    for filepath in filepaths:
        os.unlink(filepath)

def show_mask(mask, ax, obj_id=None, random_color=False):
    if random_color:
        color = np.concatenate([np.random.random(3), np.array([0.6])], axis=0)
    else:
        cmap = plt.get_cmap("tab10")
        cmap_idx = 0 if obj_id is None else obj_id
        color = np.array([*cmap(cmap_idx)[:3], 0.6])
    h, w = mask.shape[-2:]
    mask_image = mask.reshape(h, w, 1) * color.reshape(1, 1, -1)
    ax.imshow(mask_image)


def show_points(coords, labels, ax, marker_size=200):
    pos_points = coords[labels==1]
    neg_points = coords[labels==0]
    ax.scatter(pos_points[:, 0], pos_points[:, 1], color='green', marker='*', s=mar
    ax.scatter(neg_points[:, 0], neg_points[:, 1], color='red', marker='*', s=marke


def show_box(box, ax):
    x0, y0 = box[0], box[1]
    w, h = box[2] - box[0], box[3] - box[1]
    ax.add_patch(plt.Rectangle((x0, y0), w, h, edgecolor='green', facecolor=(0, 0,
```

```
def track_item_boxes(imgpath1,imgpath2,img1boxclasslist,visualize=True):
    # imgpath1 :: Image where object is known
    # imgpath2 :: Image where object is to be tracked
    # img1boxclasslist :: [ ([xmin,xmax,ymin,ymax],objectnumint) ,....] for all obj
    create_if_not_exists(tempfolder)
    cleardir(tempfolder)
    shutil.copy(imgpath1,tempfolder+"/00000.jpg")
    shutil.copy(imgpath2,tempfolder+"/00001.jpg")
    inference_state = predictor_vid.init_state(video_path="./tempdir")
    predictor_vid.reset_state(inference_state)
    ann_frame_idx = 0
    for img1boxclass in img1boxclasslist:
        ([xmin,xmax,ymin,ymax],objectnumint) = img1boxclass
        box = np.array([xmin, ymin, xmax, ymax], dtype=np.float32)
        _, out_obj_ids, out_mask_logits = predictor_vid.add_new_points_or_box(
            inference_state=inference_state,
            frame_idx=ann_frame_idx,
            obj_id=objectnumint,
            box=box,
        )
    video_segments = {}  # video_segments contains the per-frame segmentation resul
    for out_frame_idx, out_obj_ids, out_mask_logits in predictor_vid.propagate_in_v
        video_segments[out_frame_idx] = {
            out_obj_id: (out_mask_logits[i] > 0.0).cpu().numpy()
            for i, out_obj_id in enumerate(out_obj_ids)
        }
    if visualize:
        fig, ax = plt.subplots()
        plt.title(f"original image object ::")
        ax.imshow(Image.open(tempfolder+"/00000.jpg"))
        rect = patches.Rectangle((xmin, ymin), xmax-xmin, ymax-ymin, linewidth=1, e
        ax.add_patch(rect)
        plt.show()
        out_frame_idx = 1
        plt.figure(figsize=(6, 4))
        plt.title(f"detected object in test image ::")
        plt.imshow(Image.open(tempfolder+"/00001.jpg"))
        for out_obj_id, out_mask in video_segments[out_frame_idx].items():
            show_mask(out_mask, plt.gca(), obj_id=out_obj_id)
    return video_segments
```

To submit the assignment :: Solve all questions and upload the code to your github. Please write a pdf doc explaining the code in case you arent working in a Jupyter notebook. Share the the link of your github repo to the HR / ParallelDots employee you received this assignment from. We will go through your code and will get back to schedule the next round if the solution looks 1. Satisfactory 2. Original and 3. Precise in that order.

In [ ]: