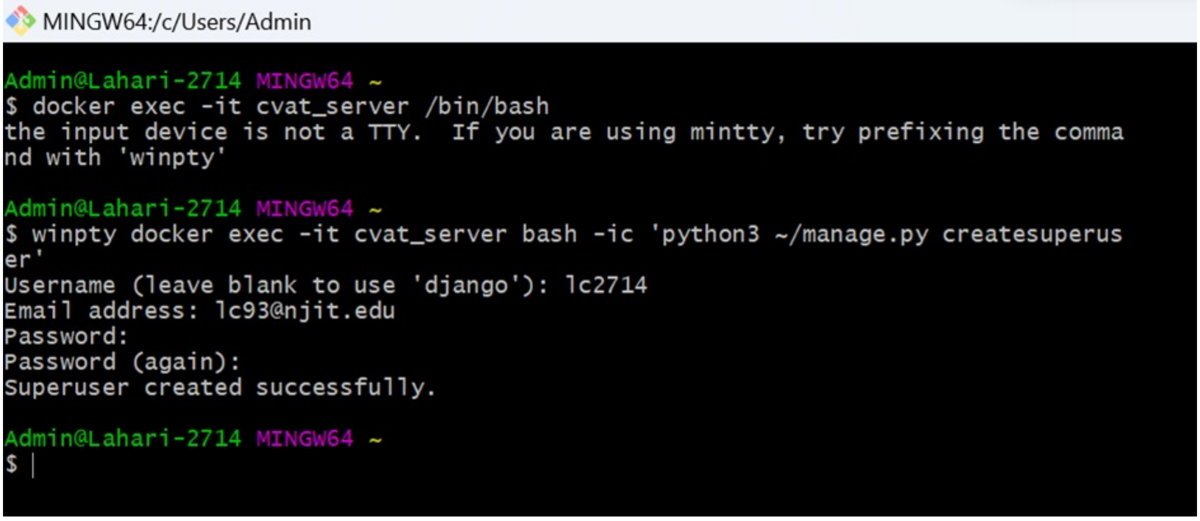
GARAGE

BATCH-19

Environment Preparation

Steps :

Install the wsl using the command wsl --install in commond prompt/ power shell after installing change the version from 1 to version 2 Then install the docker window and signup the docker window before that restart the computer Now install the the gitbash for windows And type the command cvat inorder to open the cvat. the commands are as follows git clone https://github.com/opencv/cvat cd cvat docker-compose up -d winpty docker exec -it cvat_server bash -ic 'python3 ~/manage.py createsuperuser' enter the username and password Make sure the docker window is ruuning parallel. now we can find that cvat is ruuning and is shiwn in docker window. Installation is done .

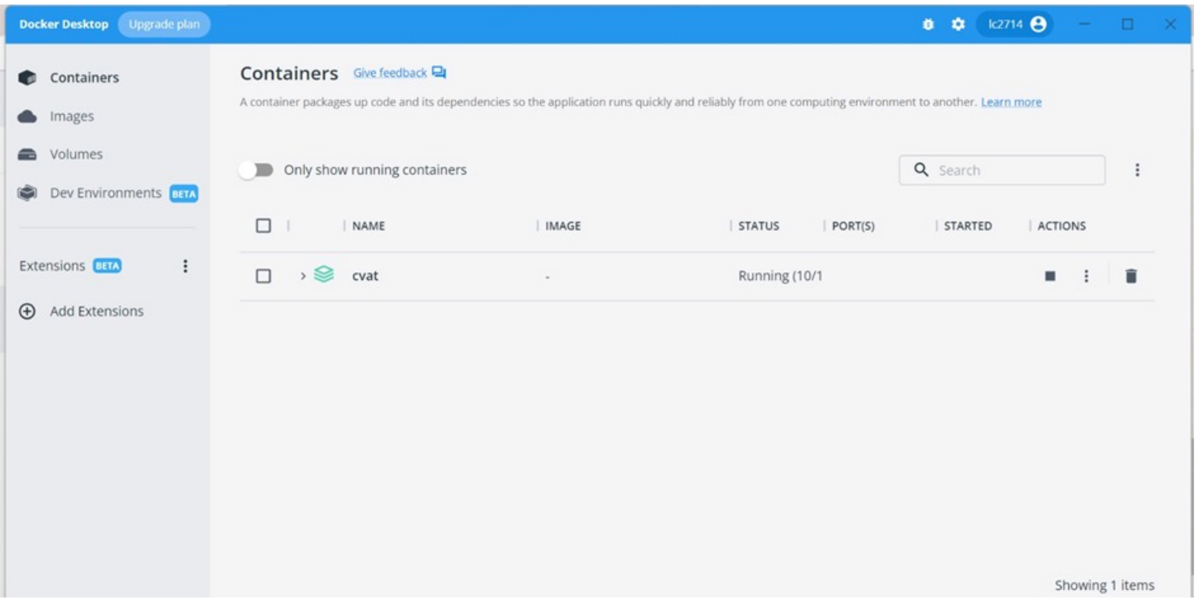


```
MINGW64:/c/Users/Admin

Admin@Lahari-2714 MINGW64 ~
$ docker exec -it cvat_server /bin/bash
the input device is not a TTY.  If you are using mintty, try prefixing the comma
nd with 'winpty'

Admin@Lahari-2714 MINGW64 ~
$ winpty docker exec -it cvat_server bash -ic 'python3 ~/manage.py createsuperus
er'
Username (leave blank to use 'django'): lc2714
Email address: lc93@njit.edu
Password:
Password (again):
Superuser created successfully.

Admin@Lahari-2714 MINGW64 ~
$ |
```



Containers

A container packages up code and its dependencies so the application runs quickly and reliably from one computing environment to another. Learn more

Only show running containers

□	NAME	IMAGE	STATUS	PORT(S)	STARTED	ACTIONS
□	cvat	-	Running (10/1			

Showing 1 items

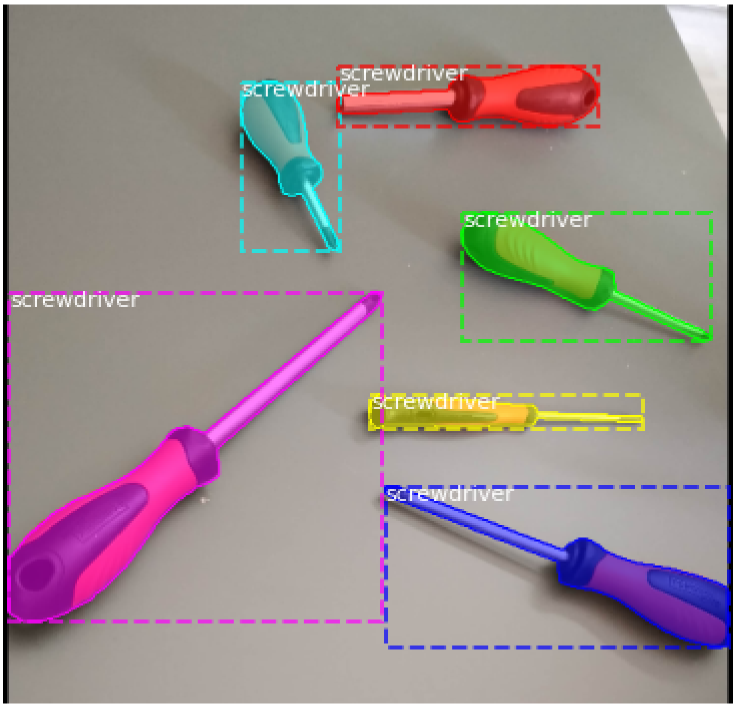
Data Acquisition

Here the google API is used for Data Acquisition. We generate a key for the data acquistion and start the Acquistion . There are 10 Categories. For each category , we aquire 100 images https://colab.research.google.com/drive/1dj_ss4Gk6k9nGV948NC9VP_DKI8Ra8JO#scrollTo=iYLc7IB4

Annotation

Here we take the images and covert them into a dataset having Training and testing data. We Peform Object Segmentation using Deep Extreme Cut (DEXTR). First we create a project in cvat and add the labels to it.

1. The label are named as Wrenches, Screwdrives,shleves, wallpanels,pliers,storage cabinets,workbenches,totes,hammer.
2. then we create a task.
3. Now upload a zip file having a 1000 images
4. Now fit the image in the rectangle and assign the image under correct label
5. Repeat the process for all the 1000 images.
6. make sure each label has a 100 images under it.
7. Now create an coount in roboflow
8. Now convert the fitted images into a MScoco file.
9. Download the zip file.
10. The Zip file is the Dataset
11. It contains test,train,valid and readme file and annotation for the images
12. link for roboflow https://universe.roboflow.com/njit-mhr5p/garage-co4hb 13.link to access the dataset: https://github.com/lc2714/milestone3-



Annotated image of screwdriver

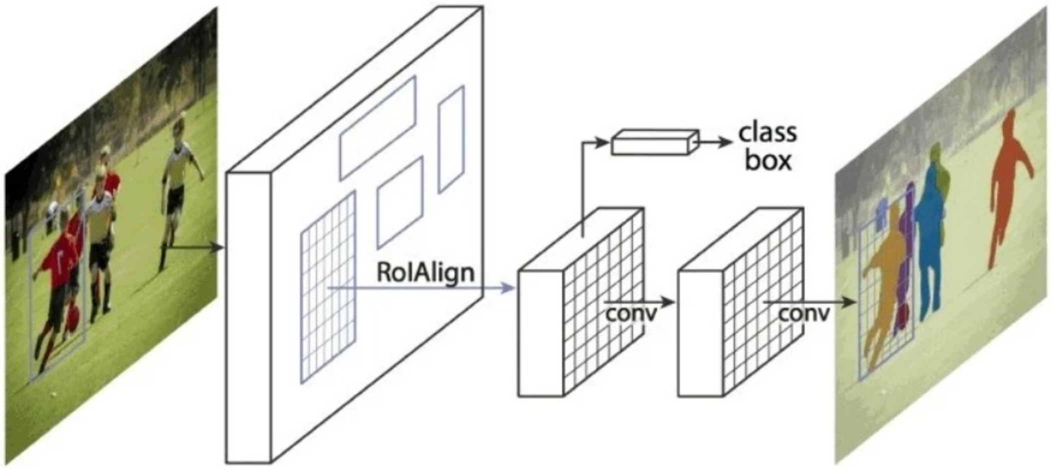
IMPLEMENTATION

Segmentation

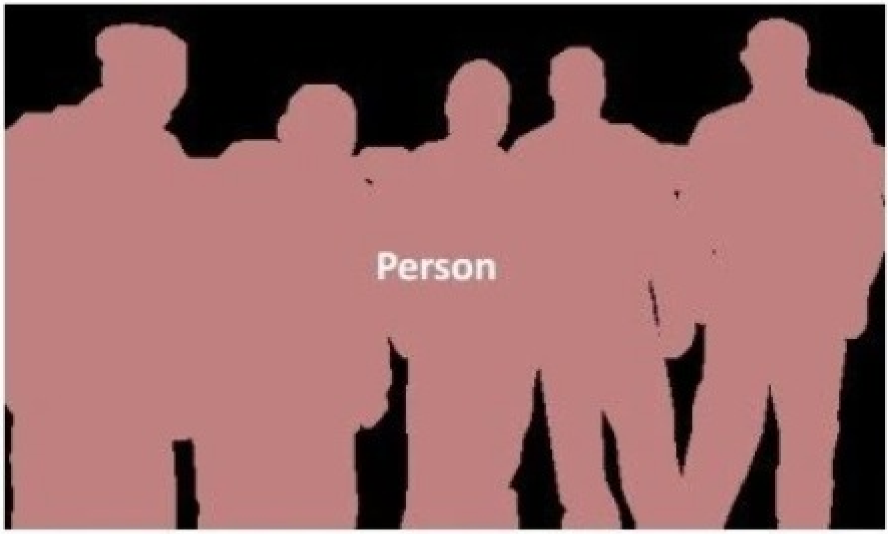
Here inorder to perform segmentation , we use MaskrCNN on garage dataset.

MaskRCNN

This architecture is used for instance image segmentation which extends Faster R-CNN (an architecture proposed by Shaoqing Ren et al to eliminate selective search and allow the network to learn region proposals) by adding an object mask predictor as a parallel branch to bounding box recognition.The architecture of MASKRCNN is



Semantic Segmentation



```
In [ ]:
#segmentation
 !git clone https://github.com/matterport/Mask_RCNN.git
import os
os.chdir('Mask_RCNN/samples')
!pip install mrcnn
import os
import sys
import skimage.io
import matplotlib
import matplotlib.pyplot as plt

ROOT_DIR = os.path.abspath("./")
sys.path.append(ROOT_DIR)
from mrcnn import utils
import mrcnn.model as modellib
from mrcnn import visualize
sys.path.append(os.path.join(ROOT_DIR, "samples/coco/"))
```

```
from samples.coco import coco
import cv2
from matplotlib import pyplot as plt
MODEL_DIR = os.path.join(ROOT_DIR, "logs")
COCO_MODEL_PATH = os.path.join(ROOT_DIR, "mask_rcnn_coco.h5")

if not os.path.exists(COCO_MODEL_PATH):
    utils.download_trained_weights(COCO_MODEL_PATH)
```

In [ ]:

```
IMAGE_DIR = os.path.join(ROOT_DIR, "images")
!wget http://images.cocodataset.org/zips/train2014.zip
!unzip -q train2014.zip
!wget http://images.cocodataset.org/zips/val2014.zip
!wget http://images.cocodataset.org/annotations/annotations_trainval2014.zip
!unzip -q val2014.zip
!unzip -q annotations_trainval2014.zip


! pip install 2to3
!git clone https://github.com/cocodataset/cocoapi.git
%cd cocoapi
!2to3 . -w
%cd PythonAPI
!python3 setup.py install
class_names = ['screwdrivers', 'workbench', 'pliers', 'hammer', 'wallpanel', 'storag
               'totes']
class InferenceConfig(coco.CocoConfig):
    # Set batch size to 1 since we'll be running inference on
    # one image at a time. Batch size = GPU_COUNT * IMAGES_PER_GPU
    GPU_COUNT = 1
    IMAGES_PER_GPU = 1
config = InferenceConfig()
config.display()
model = modellib.MaskRCNN(mode="inference", model_dir=MODEL_DIR, config=config)
model.load_weights(COCO_MODEL_PATH, by_name=True)
dataset_train= InferenceConfig(path_input = "/content/train2014" , path_mask = "/con
dataset_val = InferenceConfig(path_input =  "/content/val2014", path_mask =  "/conte

model.train(dataset_train, dataset_val,
            learning_rate=config.LEARNING_RATE,
            epochs=1,
            layers='heads')

model.train(dataset_train, dataset_val,
            learning_rate=config.LEARNING_RATE / 10,
            epochs=2,
            layers="all")

image_id = random.choice(dataset_val.image_ids)
original_image, image_meta, gt_class_id, gt_bbox, gt_mask =\
    modellib.load_image_gt(dataset_val, inference_config,
                           image_id, use_mini_mask=False)

log("original_image", original_image)
log("image_meta", image_meta)
log("gt_class_id", gt_class_id)
log("gt_bbox", gt_bbox)
log("gt_mask", gt_mask)

visualize.display_instances(original_image, gt_bbox, gt_mask, gt_class_id,
                            dataset_train.class_names, figsize=(8, 8))
 original= cv2.imread('/content/p2.jpeg')
results= cv2.imread('/content/p1.jpeg')
```

```
results = model.detect([original_image], verbose=1)

r = results[0]
visualize.display_instances(original_image, r['rois'], r['masks'], r['class_ids'],
                            dataset_val.class_names, r['scores'], ax=get_ax
```

In [ ]:

```
FOR REAL TIME USING CAMERA

!pip install pixellib
import pixellib
from pixellib.instance import instance_segmentation

segment_image = instance_segmentation()
segment_image.load_model("mask_rcnn_coco.h5")
segment_image.segmentImage("d1.jpg", output_image_name = "image_new.jpg")
cap = cv2.VideoCapture(0)
while cap.isOpened():
    ret, frame = cap.read()

    # Apply instance segmentation
    res = segment_image.segmentFrame(frame, show_bboxes=True)
    image = res[1]

    cv2.imshow('Instance Segmentation', image)

    if cv2.waitKey(10) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

RESULT

input image

OUTPUT IMAGE