Group 17 - Final Project

Introduction

We implemented two projects, at the upper half of the week, we worked on hand gesture recognition work and encountered the following problems. Then we start a new project based on OpendataCam called Pick up and Go, which is a customer flow and popular food detection project to empower retail stores for more sales with business intelligence. And finally, we carry on with the gesture recognition work and implemented new model to make it work on this address: http://192.168.85.67:30391.

Group member of group 17: Qi Jiadun, Li Yonghui, Tian Hao, Hong Ziyang.

Project1 - Pick up and Go: Customer Flow and Popular Food Detection in Retail Store

Background and Key Elements

Background

Retail stores receive millions of visitors every year. The primary objective of a store is to **attract visitors and make sales**. However, this constant stream of traffic can be leveraged to create an alternate source of value for retailers. You can collect traffic data and **extract key retail analytics** from it. Based on the information collected from our project, you can **create a lot of sales opportunities and optimize store operations**.

Definitions

Customer Traffic : visitor traffic and purchase traffic

Where to set the cameras

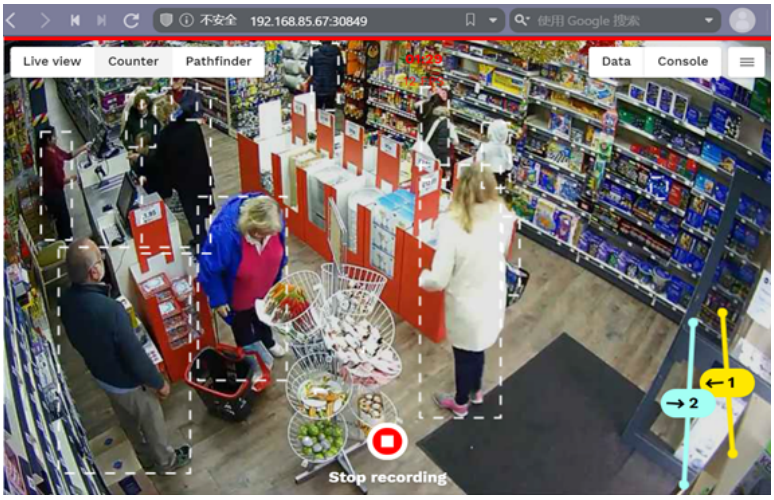
- **Entrance of the store** – Collect visitor traffic and street traffic
- **Cashier** – Collect purchase traffic data
- **Shelves** – Theft protection

Key Elements

- **Peak/off-peak periods**
- **Draw-in rate & Purchase rate**

- Draw-in rate: **Visitor traffic / Street traffic**

  Combined pedestrians traffic data and visitor traffic data collected from the entrance of store, to determine the draw in rate of store. – How appealing is your store to the pass-bys.



- Purchase rate: **Purchase traffic / Visitor traffic**

  Combine the traffic data of **cashier** and traffic data in **store** – How many customers want to buy after entering your store.

Implementation

Step1 - Build container based on opendatacam

```
FROM opendatacam/opendatacam:v3.0.2-xavier
COPY market1.mp4 /var/local/darknet/opendatacam_videos/market1.mp4
COPY market2.mp4 /var/local/darknet/opendatacam_videos/market2.mp4
CMD ./launch.sh
```

```
sudo docker build --tag market-detect:v1.7 .
sudo docker login
sudo docker tag market-detect:v1.7 jadenqi/market-detect:v1.7
sudo docker image push jadenqi/market-detect:v1.7
```

- **Step2 - Configure the opendatacam**

```
apiVersion: v1
data:
  config.json: |
    {
      "OPENDATACAM_VERSION": "3.0.2",
      "PATH_TO_YOLO_DARKNET" : "/var/local/darknet",
      "VIDEO_INPUT": "file",
      "NEURAL_NETWORK": "yolov4",
      "VIDEO_INPUTS_PARAMS": {
        "file": "opendatacam_videos/market1.mp4",
        "usbcam": "v4l2src device=/dev/video0 ! video/x-raw, framerate=30/1,
width=640, height=360 ! videoconvert ! appsink",
        "raspberrycam": "nvarguscamerasrc ! video/xraw(memory:NVMM),width=1280,
height=720, framerate=30/1, format=NV12 ! nvvidconv ! video/x-raw, format=BGRx,
width=640, height=360 ! videoconvert ! video/x-raw, format=BGR ! appsink",
"remote_cam": "YOUR IP CAM STREAM (can be .m3u8, MJPEG ...), anything supported
by opencv",
        "remote_hls_gstreamer": "souphttpsrc
location=http://YOUR_HLSSTREAM_URL_HERE.m3u8 ! hlsdemux ! decodebin !
videoconvert ! videoscale ! appsink"
      },
      "TRACKER_SETTINGS": {
        "objectMaxAreaInPercentageOfFrame": 80,
        "confidence_threshold": 0.2,
        "iouLimit": 0.05,
        "unMatchedFrameTolerance": 5
      },
      "COUNTER_SETTINGS": {
        "minAngleWithCountingLineThreshold": 5,
        "computeTrajectoryBasedOnNbOfPastFrame": 5
      },
      "VALID_CLASSES": ["*"],
      "DISPLAY_CLASSES": [
        { "class": "person", "hexcode": "1F9CD"},
        { "class": "hot dog", "hexcode": "1F32D"},
        { "class": "cake", "hexcode": "1F370"},
        { "class": "cup", "hexcode": "1F964"},
        { "class": "sandwich", "hexcode": "1F96A"},
        { "class": "donut", "hexcode": "1F369"}
      ],
      "PATHFINDER_COLORS": [
        "#1f77b4",
        "#ff7f0e",
```

```
        "#2ca02c",
        "#d62728",
        "#9467bd",
        "#8c564b",
        "#e377c2",
        "#7f7f7f",
        "#bcbd22",
        "#17becf"
      ],
      "COUNTER_COLORS": {
        "yellow": "#FFE700",
        "turquoise": "#A3FFF4",
        "green": "#a0f17f",
        "purple": "#d070f0",
        "red": "#AB4435"
      },
      "NEURAL_NETWORK_PARAMS": {
        "yolov4": {
          "data": "cfg/coco.data",
          "cfg": "cfg/yolov4-416x416.cfg",
          "weights": "yolov4.weights"
        },
        "yolov4-tiny": {
          "data": "cfg/coco.data",
          "cfg": "cfg/yolov4-tiny.cfg",
          "weights": "yolov4-tiny.weights"
        }
      },
      "TRACKER_ACCURACY_DISPLAY": {
        "nbFrameBuffer": 300,
        "settings": {
          "radius": 3.1,
          "blur": 6.2,
          "step": 0.1,
            "gradient": {
              "0.4":"orange",
              "1":"red"
            },
          "canvasResolutionFactor": 0.1
        }
      },
      "MONGODB_URL": "mongodb://opendatacam-mongo:27017",
      "PORTS": {
        "app": 8080,
        "darknet_json_stream": 8070,
        "darknet_mjpeg_stream": 8090
      }
    }
kind: ConfigMap
metadata:
  creationTimestamp: null
  name: opendatacam-market1
```

```
kubectl apply -f config.yaml
```

Step3 - YAML files

YAML file for OpendataCam

`opendatacam-deployment.yaml`

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: opendatacam
  name: opendatacam
spec:
  replicas: 1
  selector:
    matchLabels:
      app: opendatacam
      tier: frontend
  template:
    metadata:
      labels:
        app: opendatacam
        tier: frontend
    spec:
      containers:
      - image: jadenqi/market-detect:v1.7
        command: ["/bin/bash"]
        args: ["-c", "/var/local/opendatacam/launch.sh"]
        name: opendatacam
        ports:
        - containerPort: 8080
        - containerPort: 8070
        - containerPort: 8090
        resources: {}
        securityContext:
          privileged: true
        volumeMounts:
        - mountPath: /var/local/opendatacam/config.json
          name: opendatacam-config
          subPath: "config.json"
      restartPolicy: Always
      volumes:
      - name: opendatacam-config
        configMap:
          name: opendatacam-market1
```

`opendatacam-service.yaml`

```yaml
apiVersion: v1
kind: Service
metadata:
  labels:
    app: opendatacam
  name: opendatacam
spec:
  ports:
  - name: "8070"
    port: 8070
    targetPort: 8070
  - name: "8090"
    port: 8090
    targetPort: 8090
  - name: "8080"
    port: 8080
    targetPort: 8080
  selector:
    app: opendatacam
    tier: frontend
  type: LoadBalancer
```

YAML file for mongoDB

`opendatacam-mongo-pvc.yaml`

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mongodb-pv-claim
  labels:
    app: opendatacam
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
```

`opendatacam-mongo-deployment.yaml`

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: opendatacam
```

```
  name: opendatacam-mongo
spec:
  replicas: 1
  selector:
    matchLabels:
      app: opendatacam
      tier: mongo
  template:
    metadata:
      labels:
        app: opendatacam
        tier: mongo
    spec:
      containers:
      - image: mongo
        name: mongo
        ports:
        - containerPort: 27017
        resources: {}
        volumeMounts:
        - mountPath: /data/db
          name: mongodb-persistent-storage
      restartPolicy: Always
      volumes:
      - name: mongodb-persistent-storage
        persistentVolumeClaim:
          claimName: mongodb-pv-claim
```

`opendatacam-mongo-service.yaml`

```
apiVersion: v1
kind: Service
metadata:
  labels:
    app: opendatacam
  name: opendatacam-mongo
spec:
  ports:
  - name: "27017"
    port: 27017
    targetPort: 27017
  selector:
    app: opendatacam
    tier: mongo
  type: ClusterIP
```

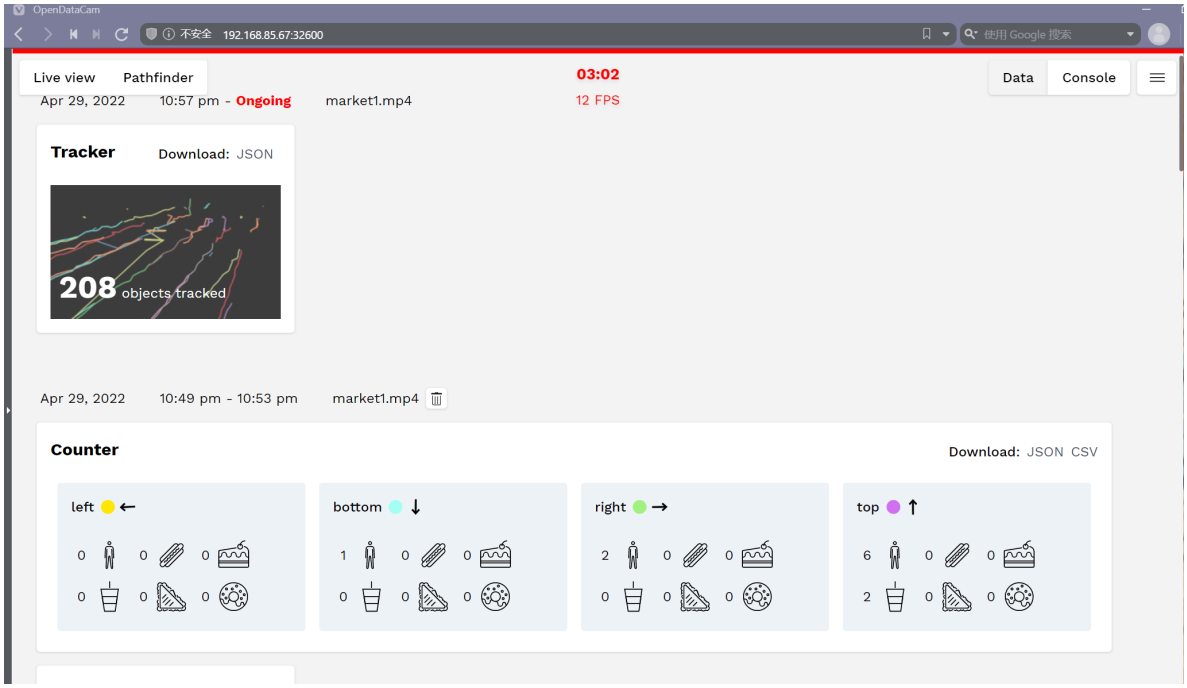
Deploy

`deployments.sh`

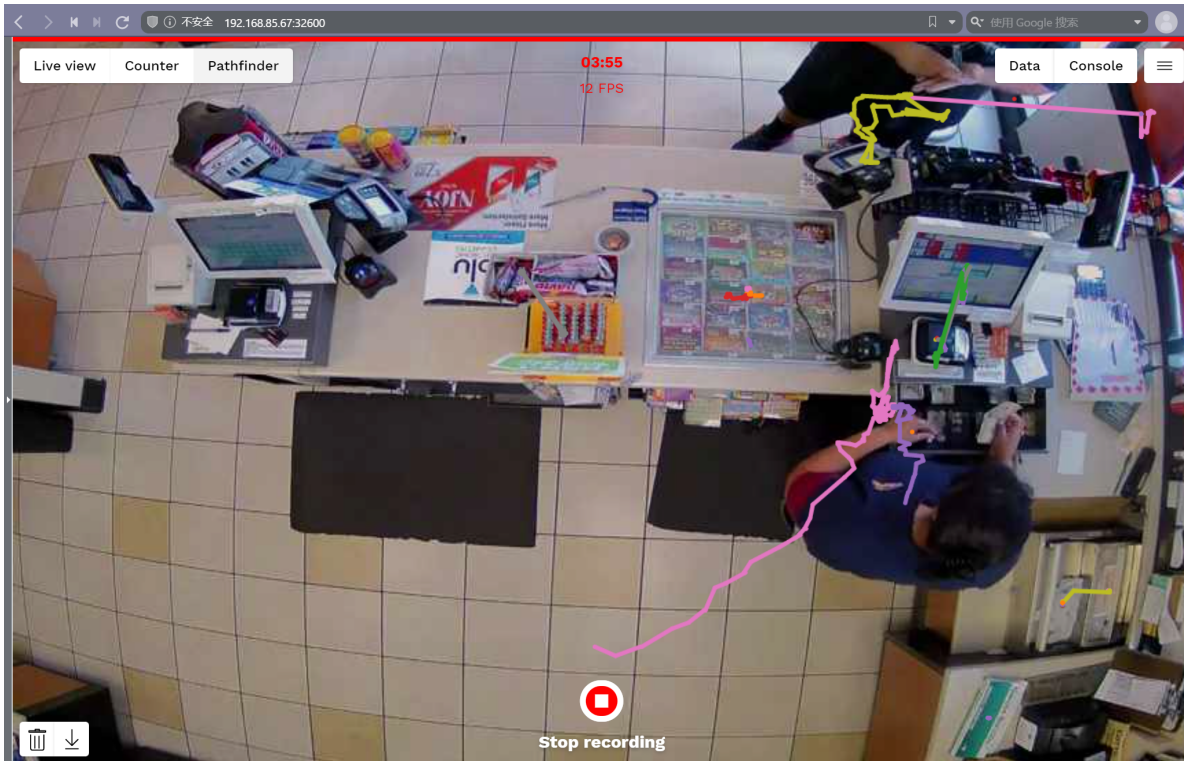
```
kubectl apply -f opendatacam-deployment.yaml
kubectl apply -f opendatacam-service.yaml
kubectl apply -f opendatacam-mongo-pvc.yaml
kubectl apply -f opendatacam-mongo-deployment.yaml
kubectl apply -f opendatacam-mongo-service.yaml
kubectl get svc
```

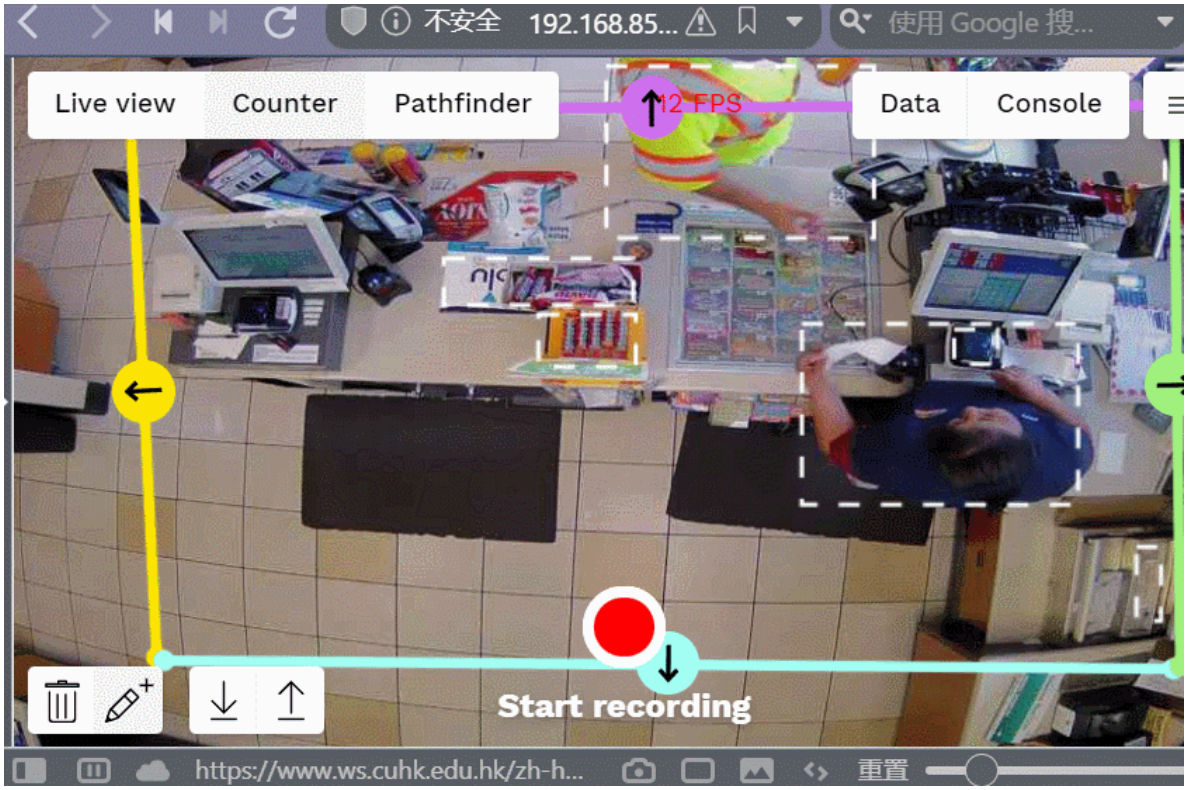
`stop-deployments.sh`

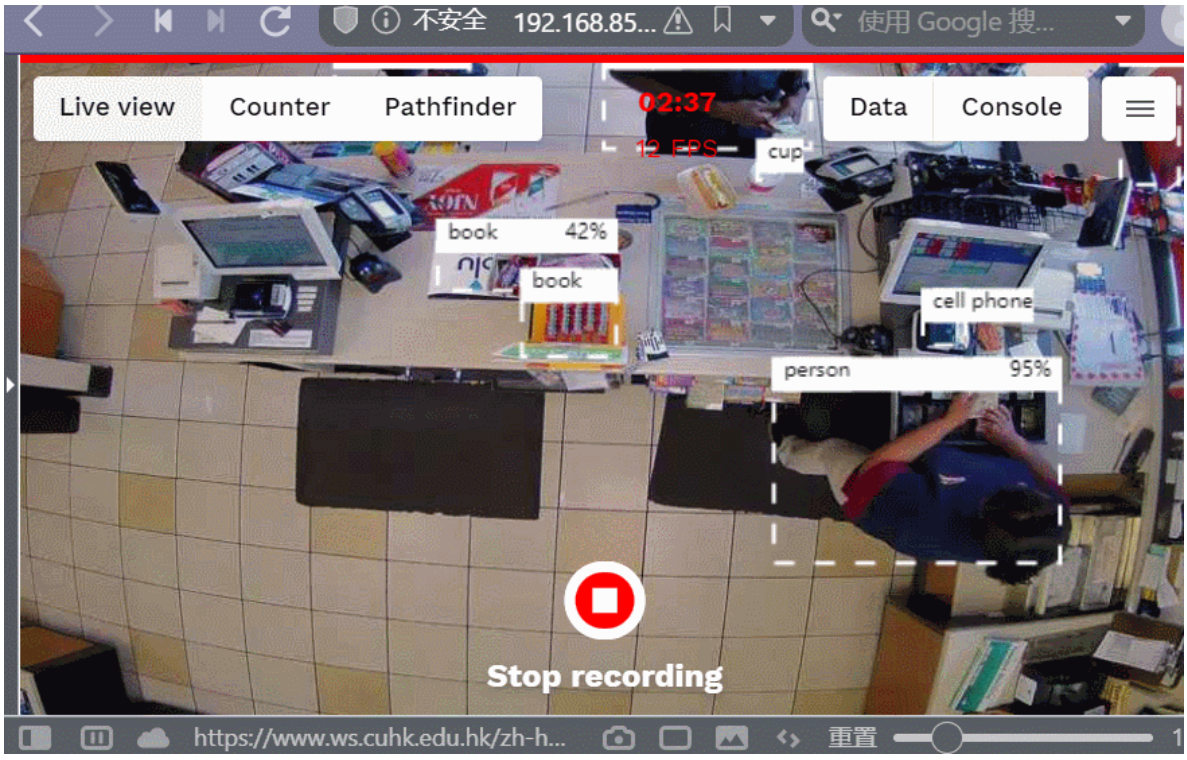
```
kubectl delete deployment opendatacam
kubectl delete deployment opendatacam-mongo
kubectl delete svc opendatacam
kubectl delete svc opendatacam-mongo
```

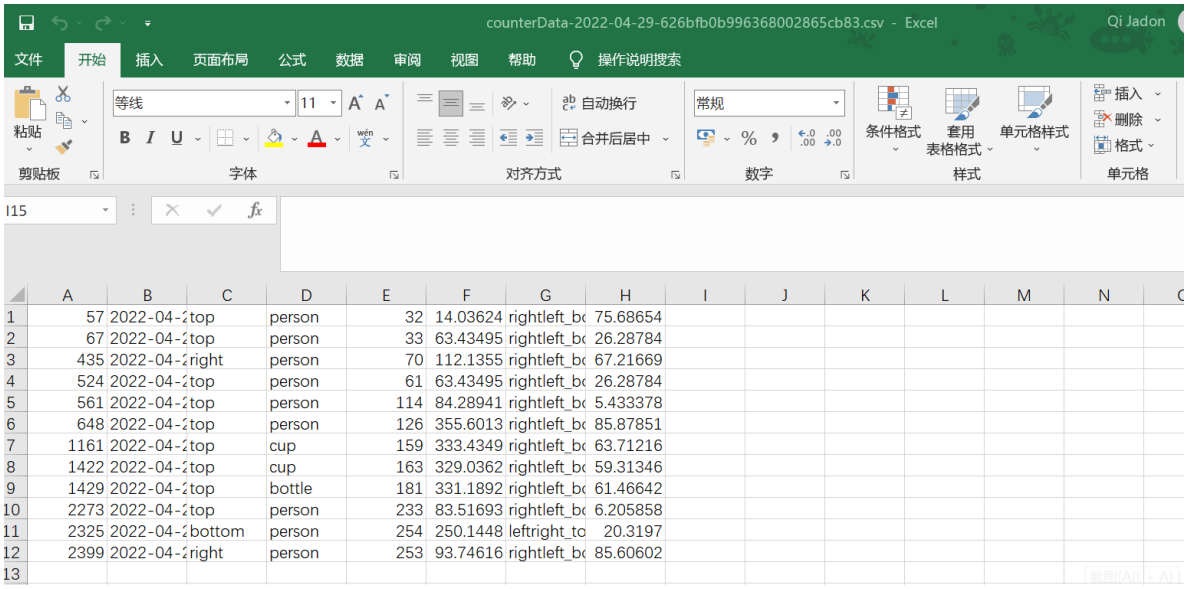



不安全 192.168.85.67:32600
使用 Google 搜索
Live view
Counter
Pathfinder
03:31
12 FPS
Data
Console
10
0
2
1
Stop recording


不安全 192.168.85.67:32600
使用 Google 搜索
Live view
Counter
Pathfinder
03:55
12 FPS
Data
Console
Stop recording


不安全
192.168.85...
使用 Google 搜...
Live view
Counter
Pathfinder
12 FPS
Data
Console
Start recording
https://www.ws.cuhk.edu.hk/zh-h...
重置


不安全
192.168.85...
使用 Google 搜...
Live view
Counter
Pathfinder
02:37
12 FPS
Data
Console
cup
book 42%
book
cell phone
person 95%
Stop recording
https://www.ws.cuhk.edu.hk/zh-h...
重置



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	57	2022-04-2	top	person	32	14.03624	rightleft_bc	75.68654						
2	67	2022-04-2	top	person	33	63.43495	rightleft_bc	26.28784						
3	435	2022-04-2	right	person	70	112.1355	rightleft_bc	67.21669						
4	524	2022-04-2	top	person	61	63.43495	rightleft_bc	26.28784						
5	561	2022-04-2	top	person	114	84.28941	rightleft_bc	5.433378						
6	648	2022-04-2	top	person	126	355.6013	rightleft_bc	85.87851						
7	1161	2022-04-2	top	cup	159	333.4349	rightleft_bc	63.71216						
8	1422	2022-04-2	top	cup	163	329.0362	rightleft_bc	59.31346						
9	1429	2022-04-2	top	bottle	181	331.1892	rightleft_bc	61.46642						
10	2273	2022-04-2	top	person	233	83.51693	rightleft_bc	6.205858						
11	2325	2022-04-2	bottom	person	254	250.1448	leftright_to	20.3197						
12	2399	2022-04-2	right	person	253	93.74616	rightleft_bc	85.60602						
13														

• Future Work

Find better models to:

1. Differentiate more feature of customers

   Gender, Age, Occupation…

2. Differentiate more kinds of products/foods

   More than Hot dog, cake, chips, coffee etc.

Project2 - Understand in Silent: Hand Gesture Recognition

– Introduction

We also propose a Hand Gesture Recognition application aiming at helping people with disabilities in speaking and listening to understand each other in silent, since the model is still have room to improve, we currently present a presentation that can only recognize gestures that indicating numbers.

– Step1 - Build container Image

Use opendatacam as base image and darknet as model framework, we find a new model from GitHub and implement it.

https://github.com/ShahrozTanveer/Hand-Gestures-Recognition, this is a **customized model** based on YOLO-v3 that is very fast in detection hand gesture inference.

```
FROM opendatacam/opendatacam:v3.0.2-xavier
COPY gesture.mp4 /var/local/darknet/opendatacam_videos/gesture.mp4

RUN mkdir gesture

COPY yolov3_custom_last.weights /var/local/darknet/gesture/

COPY ./Hand-Gestures-Recognition/Yolo/obj.data /var/local/darknet/cfg
COPY ./Hand-Gestures-Recognition/Yolo/yolov3_custom.cfg /var/local/darknet/cfg
COPY ./Hand-Gestures-Recognition/Yolo/obj.names /var/local/darknet/data

CMD ./launch.sh
```

```
sudo docker build --tag gesture-detect:v1.9 .
sudo docker login
sudo docker tag gesture-detect:v1.9 jadenqi/gesture-detect:v1.9
sudo docker image push jadenqi/gesture-detect:v1.9
```

Check out the running container.

```
docker exec -i -t 93c94xxxxx /bin/bash
```

Here we configure the model to use `yolov3-custom`.

```
apiVersion: v1
data:
  config.json: |
    {
      "OPENDATACAM_VERSION": "3.0.2",
      "PATH_TO_YOLO_DARKNET" : "/var/local/darknet",
      "VIDEO_INPUT": "file",
      "NEURAL_NETWORK": "yolov3-custom",
      "VIDEO_INPUTS_PARAMS": {
        "file": "opendatacam_videos/gesture.mp4",
        "usbcam": "v4l2src device=/dev/video0 ! video/x-raw, framerate=30/1,
width=640, height=360 ! videoconvert ! appsink",
        "raspberrycam": "nvarguscamerasrc ! video/xraw(memory:NVMM),width=1280,
height=720, framerate=30/1, format=NV12 ! nvvidconv ! video/x-raw, format=BGRx,
width=640, height=360 ! videoconvert ! video/x-raw, format=BGR ! appsink",
"remote_cam": "YOUR IP CAM STREAM (can be .m3u8, MJPEG ...), anything supported
by opencv",
        "remote_hls_gstreamer": "souphttpsrc
location=http://YOUR_HLSSTREAM_URL_HERE.m3u8 ! hlsdemux ! decodebin !
videoconvert ! videoscale ! appsink"
      },
      "TRACKER_SETTINGS": {
```

```
    "objectMaxAreaInPercentageOfFrame": 80,
    "confidence_threshold": 0.2,
    "iouLimit": 0.05,
    "unMatchedFrameTolerance": 5
  },
  "COUNTER_SETTINGS": {
    "minAngleWithCountingLineThreshold": 5,
    "computeTrajectoryBasedOnNbOfPastFrame": 5
  },
  "VALID_CLASSES": ["*"],
  "DISPLAY_CLASSES": [
    { "class": "person", "hexcode": "1F9CD"},
    { "class": "hot dog", "hexcode": "1F32D"},
    { "class": "cake", "hexcode": "1F370"},
    { "class": "cup", "hexcode": "1F964"},
    { "class": "sandwich", "hexcode": "1F96A"},
    { "class": "donut", "hexcode": "1F369"}
  ],
  "PATHFINDER_COLORS": [
    "#1f77b4",
    "#ff7f0e",
    "#2ca02c",
    "#d62728",
    "#9467bd",
    "#8c564b",
    "#e377c2",
    "#7f7f7f",
    "#bcbd22",
    "#17becf"
  ],
  "COUNTER_COLORS": {
    "yellow": "#FFE700",
    "turquoise": "#A3FFF4",
    "green": "#a0f17f",
    "purple": "#d070f0",
    "red": "#AB4435"
  },
  "NEURAL_NETWORK_PARAMS": {
    "yolov4": {
      "data": "cfg/coco.data",
      "cfg": "cfg/yolov4-416x416.cfg",
      "weights": "yolov4.weights"
    },
    "yolov4-tiny": {
      "data": "cfg/coco.data",
      "cfg": "cfg/yolov4-tiny.cfg",
      "weights": "yolov4-tiny.weights"
    },
    "yolov3-voc":{
     "data": "cfg/voc.data",
     "cfg": "cfg/yolov3-voc.cfg",
     "weights": "gesture/yolov3-voc_final.weights"
```

```
        },
        "yolov3-custom":{
          "data": "cfg/obj.data",
          "cfg": "cfg/yolov3_custom.cfg",
          "weights": "gesture/yolov3_custom_last.weights"
        }
      },
      "TRACKER_ACCURACY_DISPLAY": {
        "nbFrameBuffer": 300,
        "settings": {
          "radius": 3.1,
          "blur": 6.2,
          "step": 0.1,
            "gradient": {
              "0.4":"orange",
              "1":"red"
            },
          "canvasResolutionFactor": 0.1
        }
      },
      "MONGODB_URL": "mongodb://opendatacam-mongo:27017",
      "PORTS": {
        "app": 8080,
        "darknet_json_stream": 8070,
        "darknet_mjpeg_stream": 8090
      }
    }
kind: ConfigMap
metadata:
  creationTimestamp: null
  name: opendatacam-gesture
```

Step2 - YAML files

YAML file for OpendataCam

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: opendatacam
  name: opendatacam
spec:
  replicas: 1
  selector:
    matchLabels:
      app: opendatacam
      tier: frontend
  template:
    metadata:
      labels:
```

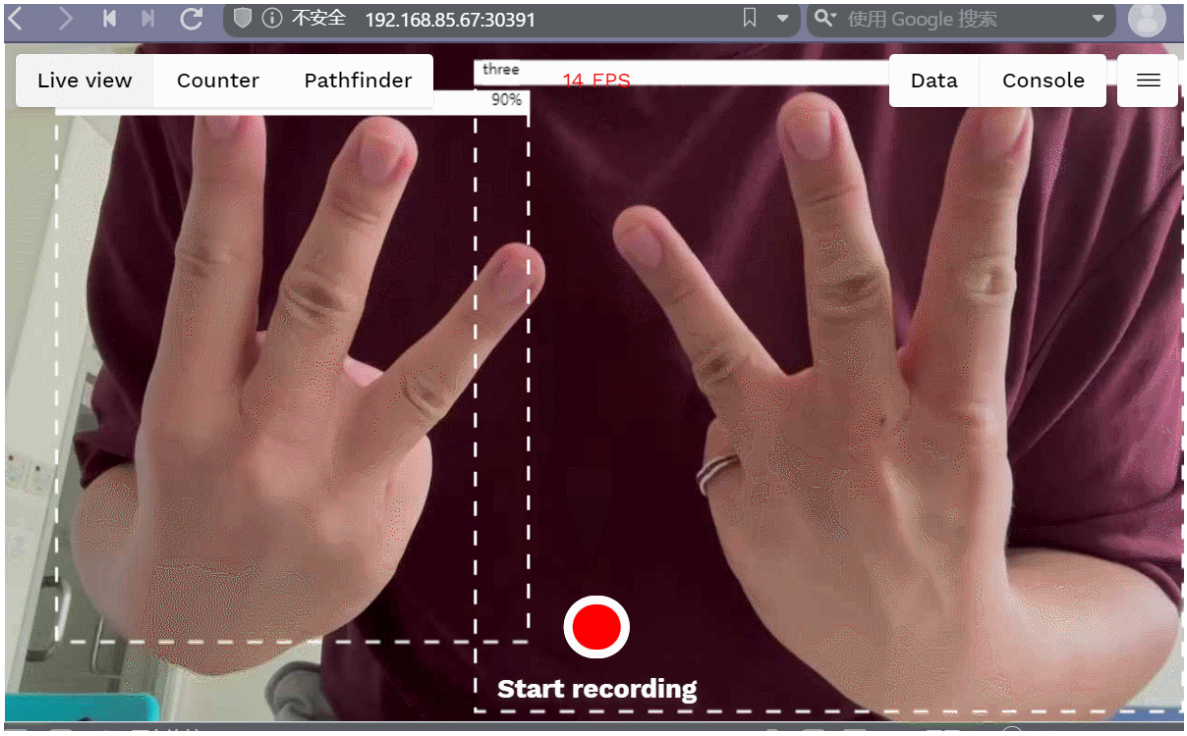
```
        app: opendatacam
        tier: frontend
    spec:
      containers:
      - image: jadenqi/gesture-detect:v1.9
        command: ["/bin/bash"]
        args: ["-c", "/var/local/opendatacam/launch.sh"]
        name: opendatacam
        ports:
        - containerPort: 8080
        - containerPort: 8070
        - containerPort: 8090
        resources: {}
        securityContext:
          privileged: true
        volumeMounts:
        - mountPath: /var/local/opendatacam/config.json
          name: opendatacam-config
          subPath: "config.json"
      restartPolicy: Always
      volumes:
      - name: opendatacam-config
        configMap:
          name: opendatacam-gesture
```

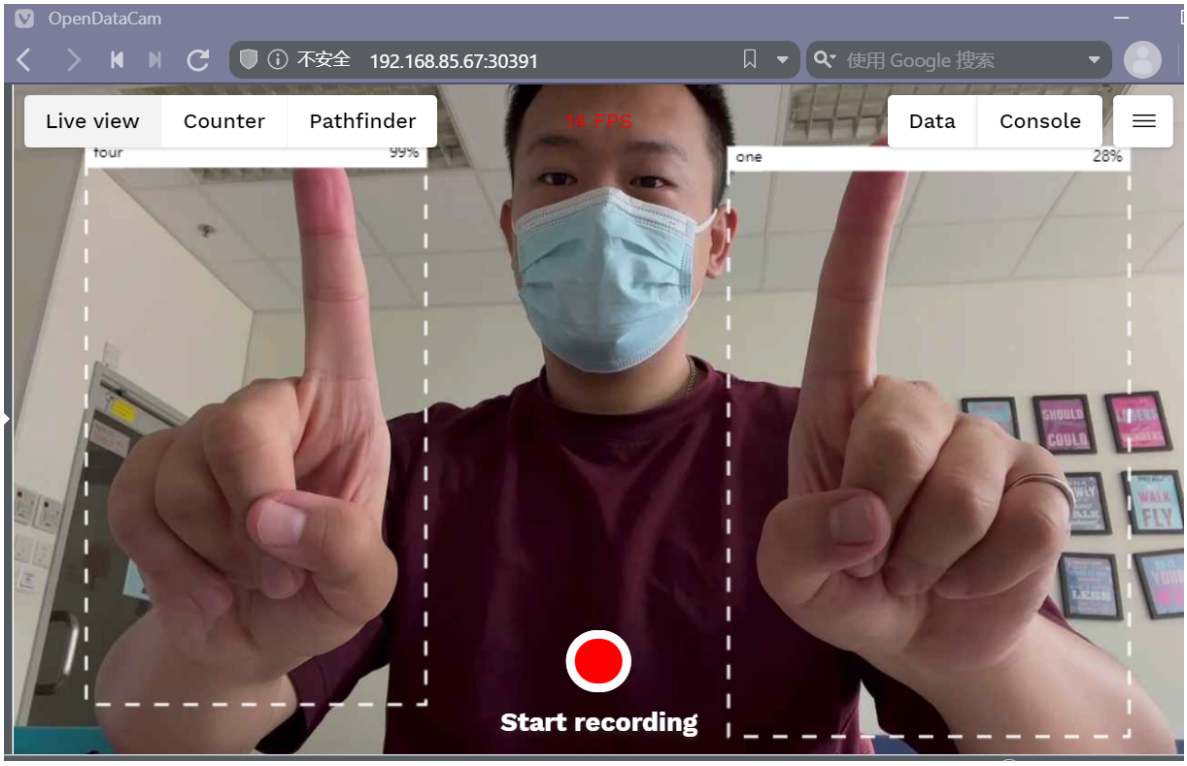
YAML files for MongoDB

The YAML file is the same as Project1, so we choose skip this part.

Deployment and Result

Since the **result of Project1 was shown on class** presentation, so we choose to **deploy Project2 using K8s** and keep it running for you to check the running result. You can see the result using this address (with): http://192.168.85.67:30391 .





Conclusion

We start with gesture recognition project but encountered many problems, so we start a new project called 'Pick up and Go' to try to use simple technology and directly use OpendataCam container image to inference and provide many strategies including draw-in rate and purchase rate, to empower the retail stores with business intelligence. After class demonstration, we continue working on the gesture recognition project we left, we implemented new model on the based on darknet framework to make this work, however, this model is now not powerful enough to help deaf-mute people to communicate only based on hand gesture. More models need to be trained to help in this area.