Collect data

Note: This course is only applicable to the R2L car and Yahboom's autopilot map.

If you use other models or other maps, you need to develop and debug the code by yourself, and the code provided in this chapter cannot be used directly.

1. Install the cvui library

Open the system terminal and enter the following command to install the cvui library.

```
sudo pip3 install cvui
```

2. Code analysis

Code path:Rosmaster/auto_drive/data_collection.py

1. Initialize the cvui library. The cvui library is based on the image operation library of opencv and can display multiple windows.

```
# initial cvui
WINDOW_NAME = 'CVUI DATA COLLECTION'
cvui.init(WINDOW_NAME)
```

2. Create a new data table named road_following_A/B, which is used to store data and picture information. A/B are two independent data sets, as long as A is used in practical applications.

```
TASK = 'road_following'
CATEGORIES = ['apex']
DATASETS = ['A', 'B']

TRANSFORMS = transforms.Compose([
    transforms.ColorJitter(0.2, 0.2, 0.2, 0.2),
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
])

datasets = {}
for name in DATASETS:
    datasets[name] = XYDataset(TASK + '_' + name, CATEGORIES, TRANSFORMS, random_hflip=True)
```

3. Initialize the camera, where 0 in cv2.VideoCapture(0) is the device number of the camera /dev/video0. Generally, it is 0 when only one camera is connected,if multiple cameras are connected, obtain the camera number by searching ls /dev/video*. Since the image size required by the model dataset is 224*224, it is necessary to modify the image size to 224*224.

```
cap = cv2.VideoCapture(0)
if cap.isOpened():
    cap.set(cv2.CAP_PROP_FOURCC, cv2.VideoWriter.fourcc('M', 'J', 'P', 'G'))
    cap.set(cv2.CAP_PROP_FRAME_WIDTH, 640)   # 640
    cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)  # 480
    cap.set(cv2.CAP_PROP_FPS, 30)            # 30
ret, frame = cap.read()
frame = cv2.resize(frame, (int(224), int(224)))
# image size
img_h = 224
img_w = 224
```

4. Initialize the data table and print the number of currently existing data pictures.

```
# 初始化数据表 initialize active dataset
category_value = CATEGORIES[0]
dataset = datasets[DATASETS[0]]
count_value = dataset.get_count(category_value)
print("count_value:", count_value)
```

5. Create a while loop to read the camera image and display it on the cvui window. Then listen for mouse click events.

```
while(1):
    # process image
    ret, frame = cap.read()
    frame = cv2.resize(frame, (int(224), int(224)))
    total_frame[0:img_h, 0:img_w] = frame
    cv2.setMouseCallback(WINDOW_NAME, onMouse, 0)
    # show
    cvui.imshow(WINDOW_NAME, total_frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
```

6. When it is detected that the left mouse button is clicked on the window, the onMouse function event is triggered. If it is judged to be in the camera area, record the X Y coordinates of the marked point and save the image.

```
# 鼠标事件处理 Mouse event handling
def onMouse(event, x, y, flags, param):
    if event == 1: # 鼠标按下 The mouse click
        if x <= 224 and y <= 224:
            global frame
            save_snapshot(x, y, frame)
```

7. Save the current camera picture to the local and data table, and print out the current X Y value and the number of data pictures at the same time.

```python
# 保存当前摄像头画面到本地和数据表 Saves the current camera frame to local
def save_snapshot(x, y, snapshot):
    global total_frame, count_value
    # 保存到本地 save to disk
    dataset.save_entry(category_value, snapshot, x, y)
    # 画一个小圆并显示图像 Draw a small circle and display the image
    snapshot = snapshot
    snapshot = cv2.circle(snapshot, (x, y), 8, (0, 255, 0), 3)
    total_frame[0:img_h, img_w+10:img_w*2+10] = snapshot
    count_value = dataset.get_count(category_value)
    print("x=", x,"y=", y)
    print("count_value:", count_value)
```

3. Start collecting data

Note: Since opencv needs to open the camera screen, it needs to run on the actual desktop. It can be run on the remote VNC desktop, but it cannot be run by remote SSH login. Please close other programs that open the camera before running the command, otherwise conflicts and errors will be reported.

Run the following commands

```
cd ~/Rosmaster/auto_drive
python3 data_collection.py
```

If you use the Jetson Xavier NX master car, due to internal multi-version issues, you need to use python3.7 to run the program.

```
cd ~/Rosmaster/auto_drive
python3.7 data_collection.py
```

At this point, the number of currently existing image datasets will be printed out, and one window and two screens will be displayed. The picture on the left is the real-time display picture of the camera, and the picture on the right is the image picture captured after the mouse is clicked, and a green circle with XY coordinates is added. Every time you click on the left camera screen, the count_value will automatically increase by 1 and save the picture and coordinate information.

Note: When collecting data sets, you need to put the car on the map, and click on the optimal path point on the camera screen of the car to push the car slowly along the track, and record every small step, and record the entire map. The data set can be recorded multiple times to improve the recognition accuracy. You can also open another terminal to run the handle control program, and use the handle to control the movement of the car.

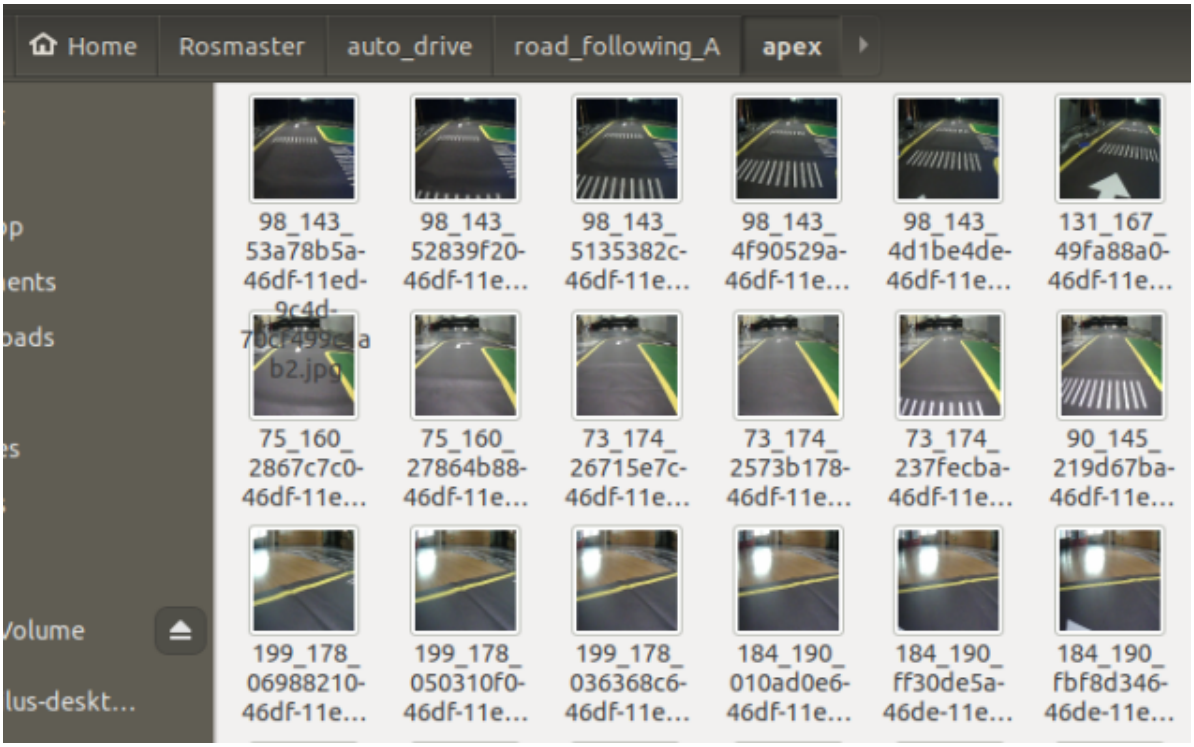
```
cd ~/Rosmaster/auto_drive/
python3 joystick_R2L.py
```

Finally, press the keyboard q key to exit the data collection function.

4. View data pictures

Picture generation path:

```
Rosmaster/auto_drive/road_following_A/apex
```



Note: If you find that there is an incorrectly entered picture and you need to cancel it, please find the corresponding picture in this folder and delete it. Then re-run the data_collection.py program.