# 金 华 职 业 技 术 学 院

JINHUA POLYTECHNIC

# 工作室成果报告

（2019 级）

项目名 Trajectory tracking system based on object detection

学　院 信息工程学院

工 作 室 自动化运维工作室

团队名称 第八组

执　笔 马浩璐，沈江源　The author of the paper: Haolu Ma, Jiangyuan shen

Team menber 成　员 柯亦威，蔡易澄，郑锦凌，朱煜洲

完成时间 2021 年 6 月 24 日

# Contents

## Abstract

In this paper, we propose a new solution to track the trajectory of object. By using the RCNN model to locate objects and frame-by-frame analysis the trajectories. We trained RCNN by using COCO 2017 image dataset, it includes 90 types of objects and user can set the object's type by manually if they do not use all of types. The tracking system through object detection results of two adjacent frames and two thresholds to track it.

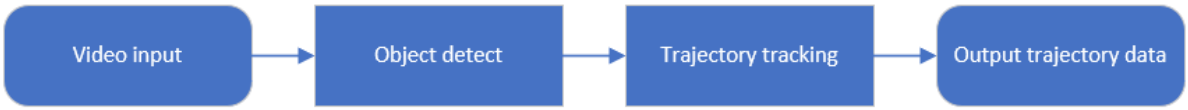
**Keywords**: trajectory tracking, object detection, computer vision

## 摘要

在本文中，我们提出了一个新的解决方案来追踪物体的轨迹。通过使用 RCNN 模型来定位物体并逐帧分析其轨迹。RCNN 模型训练数据来自 COCO 2017 数据集，数据集包括 90 种类型的物体，如果用户不使用所有类型的物体，可以通过手动设置需要检测的物体类型。追踪系统通过相邻两帧的物体检测结果和两个阈值来完成追踪。

# 1 The main structure of project

This project will receive a video stream which can come from local videos or camera. Then it will apart each frame and put it into RCNN model and the model will return all the objects' information which include the locations, classes, and confidence. The part of trajectory detection will analyze two adjacent frames and link all trajectories of objects. Finally, the trajectory data will be returned.



# 2 The object detection module

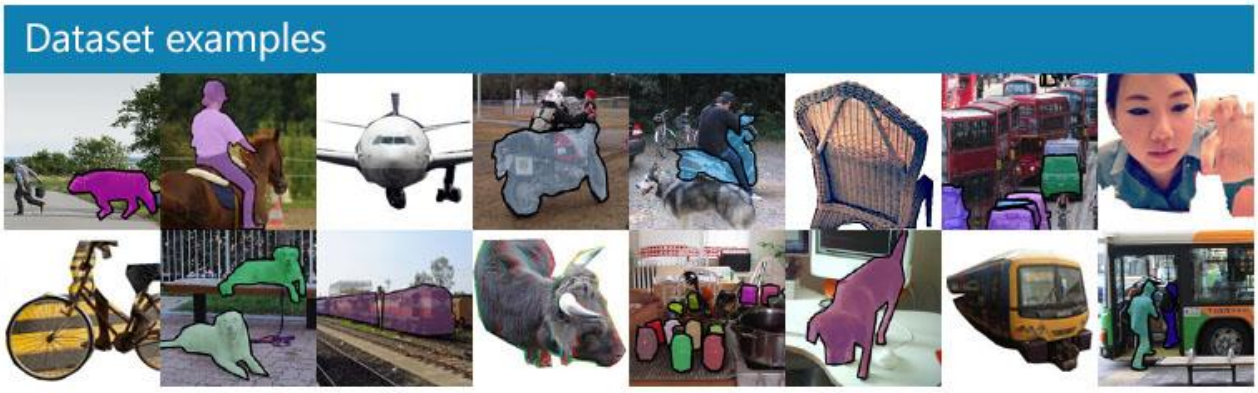
## 2.1 Introduction

This project is to detect the motion trajectory of the object in the image. We decided to use the "Convolution Neural Network", especially the "Region Convolution Neural Network", AKA "RCNN", to detect the object in a frame.

It is essential to get all the features in the image of the input video stream (include live camera and local video file) without losing track of the object. The traditional stream detection will not function in the Saturated traffic or while the objects are doing cross-movement because the color difference will be confused, so it is necessary to detect an object's edge and separate each object. And it is also impossible to use the origin OpenCV to do this job because the IO costing will be unacceptable. Therefore, we chose the RCNN model.

## 2.2 Data filter and Label

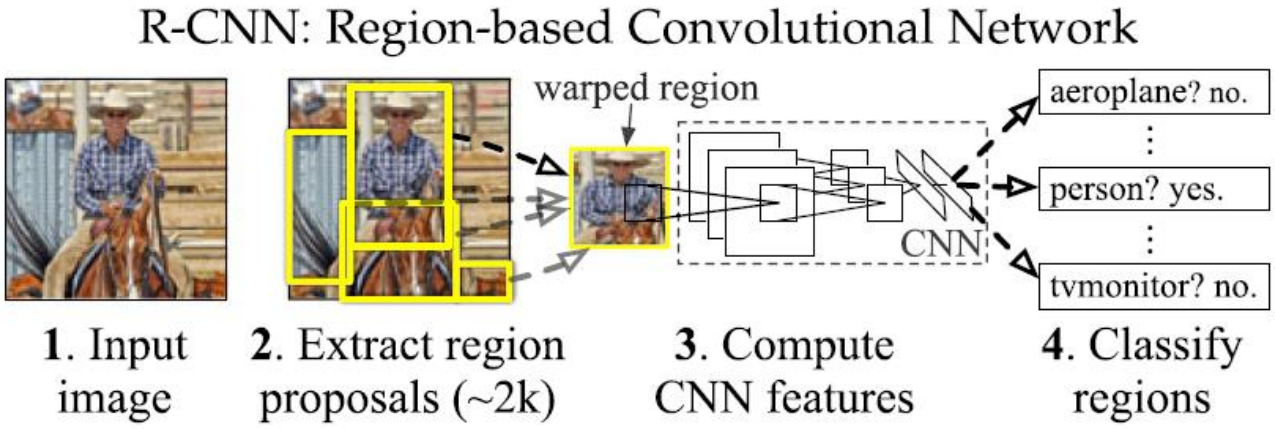
We used the COCO2017 dataset to train, test and evaluate, then we would use the TensorFlow Object detection API to train the model.




## 2.3 Structure of RCNN

A Fast R-CNN network takes as input an entire image and a set of object proposals. The network first processes the whole image with several convolutional (conv) layers and max-pooling layers to produce a conv feature map. Then, each feature vector is fed into a sequence of fully connected layers and be output as two very similar consequence on the output layer. Moreover, according to the output, one of the forecasts over softmax(the K value), and that region will be **classed** and the "Background", so after that, another layer of output will generate ***4 "trustable" vales*** and every 4 of the "trustable" vale will be

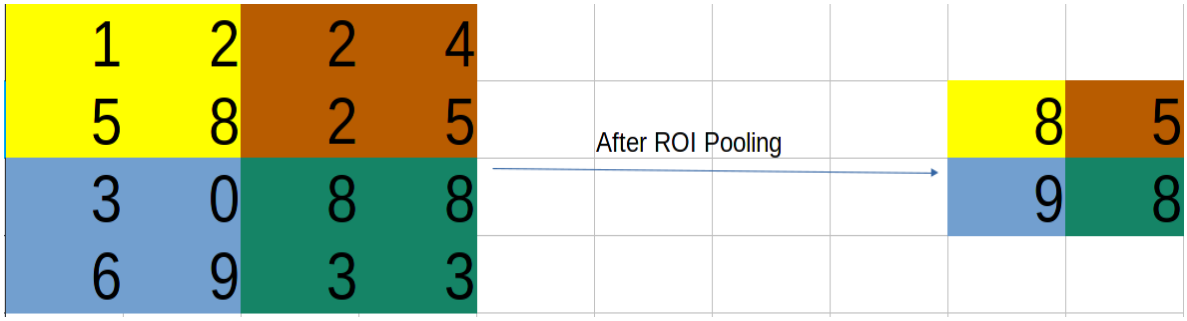
set into a Bbox, and it's positions of a **K class**. This means the softmax selector is used to output the ***probability*** of the region. And the Bbox is used to define the region, and each of the ***four values*** can generate ***one*** region.




## 2.4 The ROI (Region of Interest) Pooling Layer

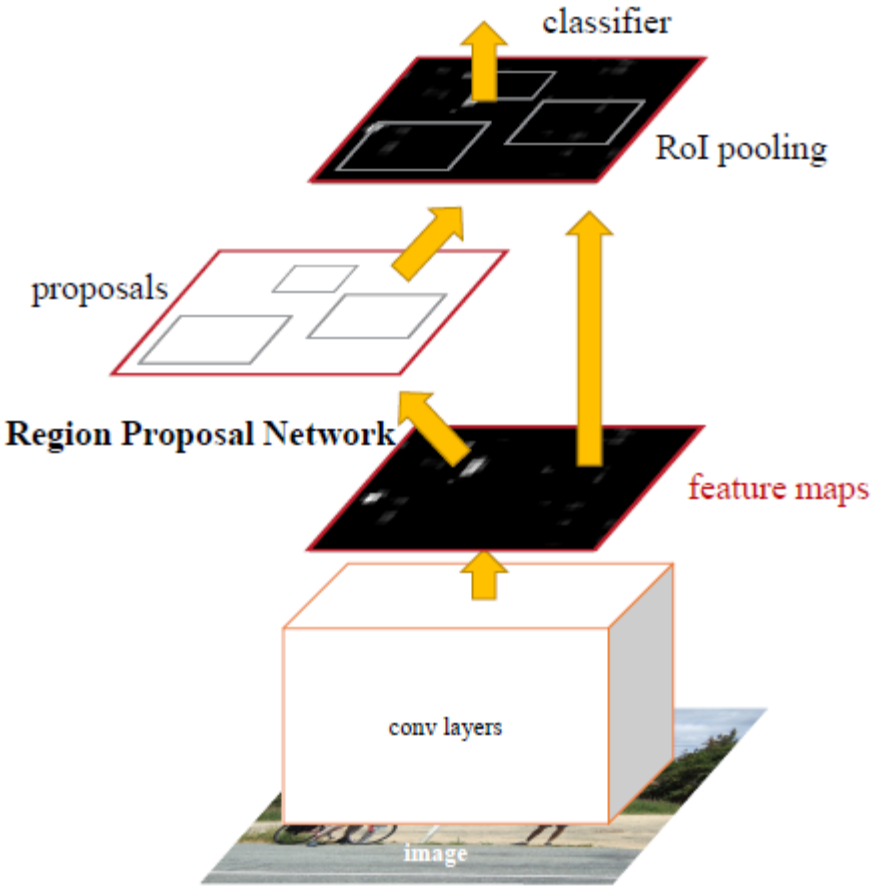
In brief, the function of this layer is to maximize the value and put the data in a fixed image, which mainly means mapping a large image into a smaller image with the same proportion. (H *x* W)→ (h *x* w)

e.g. (4x4 → 2x2)




Then the data after pooling will be sent into multiply of "Fully Connected Layer" to convert into a Single dimension vector, which called the feature vector.

Fast-RCNN mainly include two network, RPN network and R-CNN network. The RPN network charges the generation of ***candidate region***, and the RCNN network charges Identify and locate.




These two networks are very analogous. The difference is RPN will automatically draw the ***candidate region*** from feature maps**.** But the RCNN draw the ***candidate region*** directly from the input image and mapping to feature maps.

The lose function in RPN network:

$$Lloc(t\,u\,,v)\;=\;X\,i\in\{x,y,w,h\}\,smoothL1\,(t\,u\,i\;-\;vi)$$

## 2.5 Training process

### 2.5.1 Training environment

Our training environment based on *Ubuntu 18.04*. This table shows specific detail of training machine.

| | |
|---|---|
| CPU | Intel E3 1231-v3 |
| Memory | 32GB |
| GPU | GTX 970, P106-100 |
| CUDA | 9.0 |
| cuDNN | 7.0.5 |
| Python | 3.6.8 |
| TensorFlow | 1.8.0 |

### 2.5.2 Choose the data

We chose the coco 2017 dataset because it includes the 90 types of objects, it has 121,408 images, 883,331 object annotations, and median image ratio is 640 x 480.

### 2.5.3 Pre-process

First, we converted all images to grayscale, used the *mean value* method

$$\text{Gray} = \frac{(\text{R} + \text{G} + \text{B})}{3}$$

Then, we followed the mothed from *TensorFlow Document* and converted images and annotations to tfrecord structure. We set the 103,196 (85% of total) images randomly for training data, 12,140 (20% of total) images randomly for testing data, and 6,072 (5% of total) images randomly for evaluating data. And prepared the training configure file from *TensorFlow GitHub*. We set the batch size to 24, learning rate to 0.004.

Eventually, we got 3 files (training data, test data, and training configure file).

### 2.5.4 Training

We used TensorFlow object detection API to let the training processes easier. Through 4 times training and 72 hours later, we adjusted some training parameters to optimize the effect. After more than 26,000 steps the total loss was less than 5, we decided to finish the training.


It can work well in most of situation

## 2.6 Usage

As same as pre-processing we need to convert the source image to grayscale, and let it through the RCNN model, the model will return three arrays: boxes, classes, scores. The boxes array includes every object location. The classes array includes every object's type. The scores array includes every object's confidence the higher the score, the more likely the detection is to be correct. In the end, we will set a threshold to extract objects which confidence is higher than this threshold.

Then these objects will flow to the next module.

## 2.7 Section summary

This part shows that the reason of we chose RCNN model for this project, how we trained it, and how we use it. The final model can work in most senses.

# 3 The trajectory tracking module

## 3.1 Introduction

Every object has 3 statuses: *Appearance*, *Motion*, and *Disappearance*. This module can detect all of statuses of any object.

## 3.2 Theory

This model is designed for detect the same object in 2 adjacent frames. The way is to locate every object's center and compare it in 2 adjacent frames, find the minimum distance in the screen of each point, if it is lower than the set threshold these two points will be identified as an object and the line segment between these two points is the trajectory of this object in that frame.

The specific steps are as follows:

1. Get all first frame object's coordinate information from object detection module then calculator the center locations of objects, store it in the array $\boldsymbol{P_0}$.
2. Get the next frame object's center locations in the same way, store it in the array $\boldsymbol{P_1}$.
3. Find the minimum distance form every point in $\boldsymbol{P_0}$ to the point in $\boldsymbol{P_1}$. If the minimum distance is higher than the set threshold, this point will not be record (it means the point is at the *Appearance* or *Disappearance* status).
4. Link it into the array $\boldsymbol{D}$.

$\boldsymbol{D}$ is the trajectory data.

## 3.3 Section summary

This module solves how to identify same object in 2 adjacent frames. And the segment of these center point is the trajectory of this object.

# 4 Demo program

We made a demo program to evaluate our algorithm.

## 4.1 Lab environment

Evaluation environment based on *Windows 10*. Also, we install the GPU for accelerating speed of object detection model.

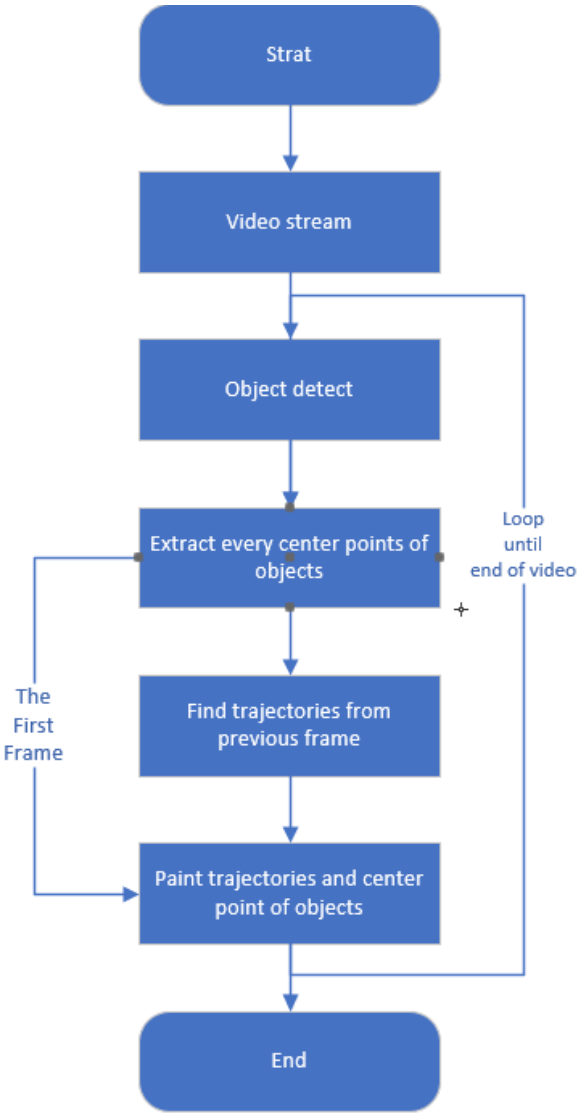
Evaluation environment

| | |
|---|---|
| CPU | Intel core i5-1135G7 |
| Memory | 40GB |
| GPU | MX450 |
| CUDA | 9.0 |
| cuDNN | 7.0.5 |
| Python | 3.6.8 |
| TensorFlow | 1.8.0 |

In this computer we can process videos in 2-4 frames per second.

## 4.2 The main structure of demo program

This demo program can read a local video. Then it will apart each frame and put it into RCNN model and the model will return all the objects' information which include the locations, classes, and confidence. The part of trajectory detection will analyze two adjacent frames and link the all the trajectories of objects. Finally, the trajectory will be painted on the video and output to the file.



## 4.3 Tracking result


# 5 Conclusion and Prospect

## 5.1 Conclusion

In this paper, we presented a general trajectory tracking system. It through object detection results of two adjacent frames and two thresholds to track it. Compared with the traditional mean shift algorithm it has better adaptability in complex situation. This system can adopt alterable shape of object not only changeless object.

## 5.2 Prospect

This project uses deep learning model and own algorithm and improve their robustness, but it still can optimize:

1. In the case of dense objects, our system sometimes could not normally track, like some different objects in two adjacent frames was identity to same. We will use optical flow method and direction of motion to assist identify.
2. Our demo program is running very slow, we will rebuild it using OOP method and add multithread processing.
3. Because we have not enough time, all the tests are lack of data to support, and we could not compare with other algorithm by data.

## Reference

[1] Ren, S., He, K., Girshick, R., & Sun, J. (2016). Faster R-CNN: towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*, *39*(6), 1137-1149.

[2] TensorFlow Document. (2021). Retrieved 24 June 2021, from https://www.tensorflow.org/

[3] Chen, W., & Chang, S. F. (1999, December). Motion trajectory matching of video objects. In *Storage and Retrieval for Media Databases 2000* (Vol. 3972, pp. 544-553). International Society for Optics and Photonics.

[4] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... & Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

[5] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 779-788).