**Fasterrcnn-python检测统计系统技术说明书**

目录

[1项目开发目的和意义 2](#_Toc12231)

[2 总体设计 3](#_Toc11467)

[3 算法设计 3](#_Toc16885)

[3.1 人群数据集 3](#_Toc4420)

[3.2 模型训练 4](#_Toc7326)

[3.3 模型转换 4](#_Toc23981)

[3.4 密集人数统计统计 4](#_Toc17501)

[3.4 流程设计 4](#_Toc22818)

[4 结果 5](#_Toc11528)

[5 关键代码说明 6](#_Toc29861)

[5.1 视频解码部分核心代码 6](#_Toc9746)

[5.2 人数统计推理的核心代码 7](#_Toc17007)

[5.3 人数统计的后处理核心代码 9](#_Toc16155)

[6 重要问题及解决 9](#_Toc26821)

[7 后续可扩展性 9](#_Toc3563)

## 1项目开发目的和意义

目标检测就是将对目标的分割和识别合二为一，是一种基于目标几何和统计特征的图像分割。当需要在一个复杂场景中对多个目标进行实时处理时，目标的自动提取和识别就尤其重要。简而言之，目标检测与识别就是指从一张图片或一个场景中找出目标，而这其中包含检测和识别两个过程。

近几年，随着目标检测的兴起，越来越多的相关研究成果被发表在各大期刊，研究该领域的人逐渐增多，目标检测也相应地不断发展。由于传统的目标检测方法复杂度较高，窗口冗余，且手工设计的特征对于多样性的变化没有很好的鲁棒性。正好，基于深度学习的目标检测和识别能较好的解决这些问题，因此，该方法更受人推广和应用，是目标检测的主流方法。其中主要涉及到深度神经网络模型以及卷积神经网络CNN。目前大致可将现有的基于深度学习的目标检测算法大致分为以下三类：

（1）基于候选区域的深度学习目标检测算法，如RCNN、 Fast-RCNN、 Faster-RCNN；

（2）基于回归的深度学习目标检测算法，如YOLO、SSD；

（3）基于搜索的深度学习目标检测算法，如基于视觉注意的AttentionNet，基于强化学习的算法。

由于深度学习的广泛运用，目标检测算法得到了较为快速的发展。当前，YOLO、SSD和RFPN属于较新且效果较好的方法，RFCN的准确度最高，但是速度最快的还是YOLO和SSD一体化的方法。本文对目标检测的研究主要采用基于候选区域建议的深度学习目标检测算法，该类算法中较知名的为RCNN、 Fast-RCNN和 Faster-RCNN。从顺序上，Fast-RCNN弥补了RCNN的一些不足之处，而Faster-RCNN则是基于Fast-RCNN进行了改进，三者在非实时水平上，精度和速度明显得到了提高。Faster-RCNN和YOLO、SSD相比，鲁棒性更强，但是速度会比较慢。Faster-RCNN是由RCNN不断演化得来的，重点提升部分是对于RCNN的效率提升，占用空间的减小。

本项目在华为Atlas 200 DK开发者板上实现对本地mp4文件或者RTSP视频流进行解码，对视频帧中的图像进行检测，生成结构化信息发送至Presenter Server端进行保存、展示。

## 2 总体设计

本系统可以分为三个部分：数据处理部分、模型构建部分和物体检测计数部分。各部分相互独立，但是存在数据关联。为了说明各系统之间的结构关系，细化结构如下图：

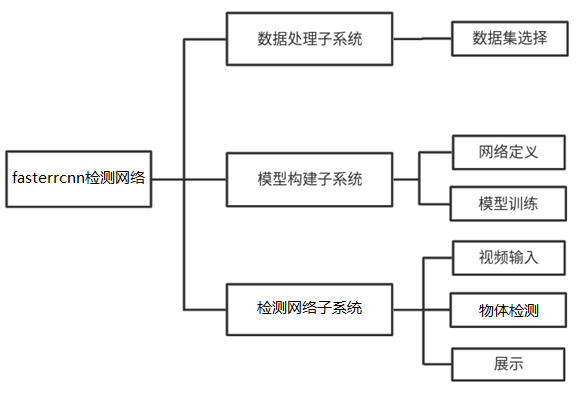


图1 系统整体功能结构图

## 3 算法设计

### 3.1 模型训练

模型的训练过程来源于<https://github.com/rbgirshick/py-faster-rcnn/blob/781a917b378dbfdedb45b6a56189a31982da1b43/models/pascal_voc/VGG16/faster_rcnn_end2end/test.prototxt>

### 3.2模型转换

训练得到的模型是caffe模型，由caffe模型在MindStudio中转换成om模型。

### 3.4 fasterrcnn-python检测网络

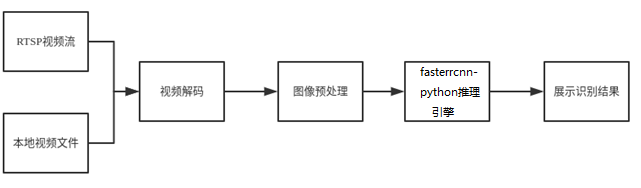


图2 fasterrcnn-python检测网络流程图

Fasterrcnn-python流程：通过对输入的RTSP视频流或MP4视频文件，读取到BGR格式的视频帧数据。当输入图片的分辨率与网络模型要求的分辨率不匹配时，使用OpenCV的resize函数对图片进行预处理，经过预处理后的图片被送入fasterrcnn网络模型进行推理，并输出原始图片及计数的结果送到presenter server进行展示。

### 3.4 流程设计

Fasterrcn-python系统总共设计了三个部分，分别为视频解码部分、推理部分、后处理部分。

1. **视频解码部分**

视频解码部分获取的是输入的RTSP视频流或MP4视频文件，读取成BGR格式的视频帧数据。

1. **fasterrcnn推理部分**

fasterrcnn推理部分是当输入图片的分辨率与网络模型要求的分辨率不匹配时，使用OpenCV的resize函数对图片进行预处理，经过预处理的图片会被送入推理部分中，并输出推理的结果。

1. **后处理部分**

后处理部分使用上一个部分的原始图片和推理结果，推理的结果是房前图片上的物体框和置信度，后处理将这些框和置信度信息打包成一个结果，把这个结果和原始图片信息通过调用Presenter Agent的API发送到UI Host上部署的Presenter Server服务进程。Presenter Server根据接收到的推理结果，并将图像信息和检测结果发送给Web UI。

## 4 结果

输入RTSP视频时，推理结果展示如图3下：

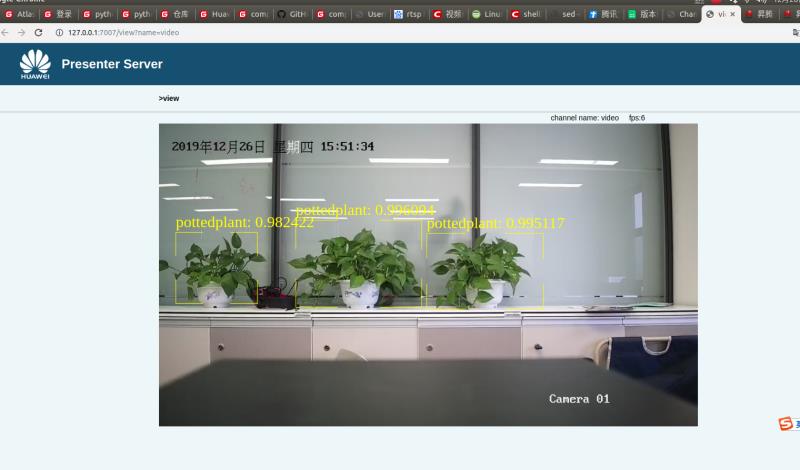


图3 推理结果展示图

可以看出已经检测到植物并且给出了置信度。

## 5 关键代码说明

### 5.1 视频解码部分核心代码

import detection\_inference

import cv2

import numpy as np

import client

import threading

import ChannelManager

from presenter\_types import \*

import time

import os

import sys

import re

from graph import \*

#判断输入是否正确（通过rtsp://和.MP4后缀判断）

lenofUrl = len(sys.argv)

if lenofUrl <= 1:

print("[ERROR] Please input mp4/Rtsp URL")

exit()

elif lenofUrl >= 3:

print("[ERROR] param input Error")

exit()

URL = sys.argv[1]

URL1 = re.match('rtsp://', URL)

URL2 = re.search('.mp4', URL)

if URL1 is None:

if URL2 is None:

print("[ERROR] should input correct URL")

exit()

else:

mp4\_url = True

else:

mp4\_url = False

detection\_app = detection\_inference.detectionInference()

detection\_app.clientsocket = client.PresenterSocketClient((presenter\_ip, presenter\_port), 5, None)

thread\_1 = threading.Thread(target=detection\_app.clientsocket.start\_connect)

thread\_1.setDaemon(True)

thread\_1.start()

time.sleep(0.1)

if detection\_app.graph is None:

print('Create Graph Failed!')

exit()

channel\_manager = ChannelManager.ChannelManager()

data = channel\_manager.OpenChannel()

if detection\_app.clientsocket is None:

print('detection\_app.clientsocket is None')

exit()

detection\_app.clientsocket.send\_data(data)

cap = cv2.VideoCapture(URL)

ret, frame = cap.read()

if mp4\_url:

try:

while ret:

detection\_inference.dowork(frame,detection\_app)

ret, frame = cap.read()

except Exception as e:

print("ERROR",e)

finally:

detection\_app.dispose()

else:

rtsp\_queue = client.Queue()

sub\_thread = threading.Thread(target=detection\_inference.sqEngine,args=(rtsp\_queue,detection\_app))

sub\_thread.setDaemon(True)

sub\_thread.start()

try:

while ret:

rtsp\_queue.put(frame)

ret, frame = cap.read()

except Exception as e:

print("ERROR",e)

finally:

cv2.destroyAllWindows()

cap.release()

detection\_app.dispose()

### 5.2 人数统计推理的核心代码

class detectionInference(object):

def \_\_init\_\_(self):

self.model = ModelManager.ModelManager()

self.width = 800

self.height = 600

self.clientsocket = None

self.graph = None

self.channel\_manager=ChannelManager.ChannelManager()

self.\_getgraph()

def dispose(self):

hiai.hiai.\_global\_default\_graph\_stack.get\_default\_graph().destroy()

def \_getgraph(self):

inferenceModel = hiai.AIModelDescription('faster\_rcnn', detection\_model\_path)

#print inferenceModel

self.graph = self.model.CreateGraphWithoutDVPP(inferenceModel)

if self.graph is None:

print "CreateGraph failed"

def Inference(self,input\_image):

inputImageTensor = hiai.NNTensor(input\_image, self.width, self.height, 3, 'testImage', DataType.UINT8\_T,self.width \* self.height \* 3)

imageinfo=np.array([800,600,3]).astype(np.float32)

imageinfo=np.reshape(imageinfo,(1,3))

infoTensor = hiai.NNTensor(imageinfo, 1,3,1,'testinfo',DataType.FLOAT32\_T, imageinfo.size)

#print('inputImageTensor is :', inputImageTensor)

datalist=[inputImageTensor, infoTensor]

nntensorList = hiai.NNTensorList(datalist)

#print('nntensorList', nntensorList)

resultList = self.model.Inference(self.graph, nntensorList)

#print('inference over')

return resultList

### 5.3 人数统计的后处理核心代码

def GetDetectionImage(self, input\_image, detection\_result):

if detection\_result is None:

return None

imageList=[]

h,w,c = input\_image.shape

for resultList in detection\_result:

imageList.append(input\_image[w\*resultList[1]:w\*resultList[3],h\*resultList[2],h\*resultList[4]])

return imageList

def GetImageFrameData(self, fact\_info,input\_image):

image\_frame = ImageFrame()

image\_frame.format = 0

image\_frame.width = input\_image.shape[1]

image\_frame.height = input\_image.shape[0]

#cv2.imwrite('1-2.jpg',input\_image)

image\_frame.data = cv2.imencode(".jpg",input\_image)[1].tobytes()

#with open('1-3.jpg','wb') as f:

# f.write(image\_frame.data)

image\_frame.size = 0

#print "fact\_info in inference",fact\_info

if fact\_info :

for result in fact\_info:

dr = DetectionResult()

dr.lt.x = (int)(result[0] \* image\_frame.width / 800.0)

if dr.lt.x < 0:

dr.lt.x = 0

dr.lt.y = (int)(result[1] \* image\_frame.height / 600.0)

if dr.lt.y < 0:

dr.lt.y = 0

dr.rb.x = (int)(result[2] \* image\_frame.width / 800.0)

if dr.rb.x < 0:

dr.rb.x = 0

dr.rb.y = (int)(result[3] \* image\_frame.height / 600.0)

if dr.rb.y < 0:

dr.rb.y = 0

dr.result\_text = kind[result[4]] + ": " + str(result[5])

image\_frame.detection\_results.append(dr)

#print "result:",result

all\_data = self.channel\_manager.PackRequestData(image\_frame)

return all\_data

def dowork(src\_image,detection\_app):

#img = cv2.imread("./1.jpg")

input\_image = cv2.cvtColor(src\_image, cv2.COLOR\_BGR2RGB)

input\_image = cv2.resize(input\_image, (detection\_app.width, detection\_app.height))

resultList = detection\_app.Inference(input\_image)

#print("resulylist[0] max:",max(resultList[0][0][0][0]))

#if resultList:

# print resultList[0]

fact\_info = detection\_app.GetDetectionInfo(resultList)

all\_data = detection\_app.GetImageFrameData(fact\_info, src\_image)

if all\_data:

detection\_app.clientsocket.send\_data(all\_data)

def sqEngine(rtsp\_queue,detection\_app):

while True:

frame = rtsp\_queue.get()

if frame is None:

time.sleep(0.1)

continue

dowork(frame,detection\_app)

## 6 重要问题及解决

问题1：presenter server上fps展示过低

解答：由于原图过大，图片发送到present server上展示的需要的时间比较长，在图片分辨率允许的情况下，可以把原图resize后再发送。

## 7 后续可扩展性

物体检测可以继续丰富模型，加强准确率。后续在安全驾驶或者物体识别领域有很大的作用。