**密集人数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

从图像或视频中准确估计人群已成为计算机视觉技术在人群控制和公共安全中有着日益重要的应用。在某些情况下，如公众集会和体育赛事，参与人数或密度是未来赛事规划和空间设计的重要信息。好的人群计数方法也可以扩展到其他领域，例如，从显微图像中计数细胞或细菌，野生动物保护区中的动物人群估计，或估计交通枢纽或交通堵塞的车辆数量等。本文将对密集人群的人数统计的相关内容做一个较为详细的综述。

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，对视频帧中的人群图像进行检测并对其进行预测，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、模型构建部分和密集人群计数部分。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：

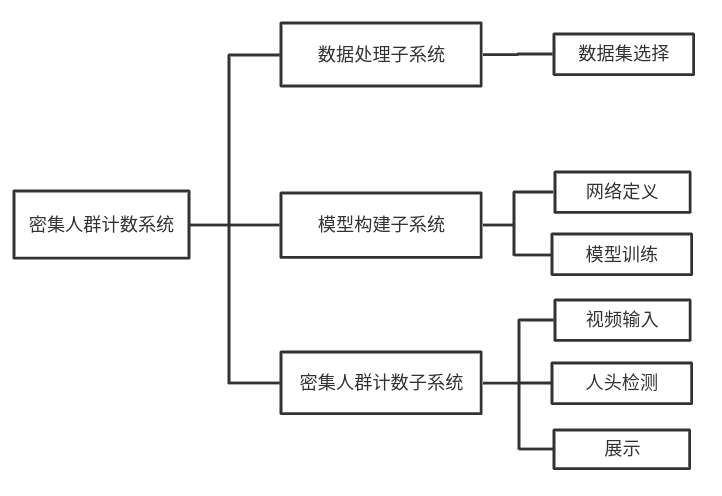


图1 系统整体功能结构图

## 3 算法设计

### 3.1 人群数据集

这里使用了大规模人群数据集，名为“上海科技”，包含近1200幅图像，其中约33万个准确标记的头部。据我们所知，它是最大的人群计数数据集方面的数字注释头。此数据集中没有两个图像是从同一个视点拍摄的。该数据集由两部分组成：A部分和B部分。A部分中的图像是从互联网上随机抓取的，其中大部分都有大量的人。B部分取自上海市区繁华的街道。

### 3.2 模型训练

模型的训练过程来源于<https://github.com/uestcchicken/crowd-counting-MCNN>

### 3.3 模型转换

训练得到的模型是h5和json文件，这里使用Keras to TensorFlow来转化成pb模型，再由pb模型在MindStudio中转换成om模型。

### 3.4 密集人数统计统计

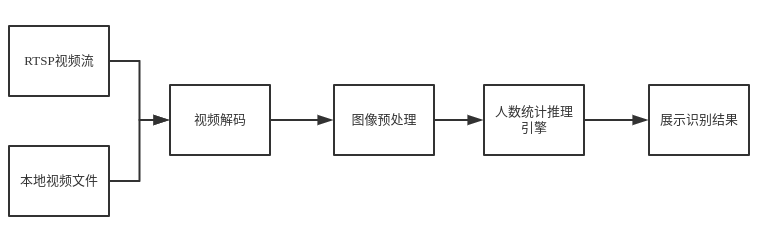


图2 人数统计流程图

人数统计流程：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密集人群计数统计网络模型进行推理，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

密集人群计数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. **人数统计推理部分**

人数统计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，推理的结果是得到每一个像素上的人头数量，后处理将这些结果全部相加后可以得到人群的计数统计结果，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人数的结果发送给Web UI。

## 4 结果

输入MP4视频时，推理结果展示如图3下：

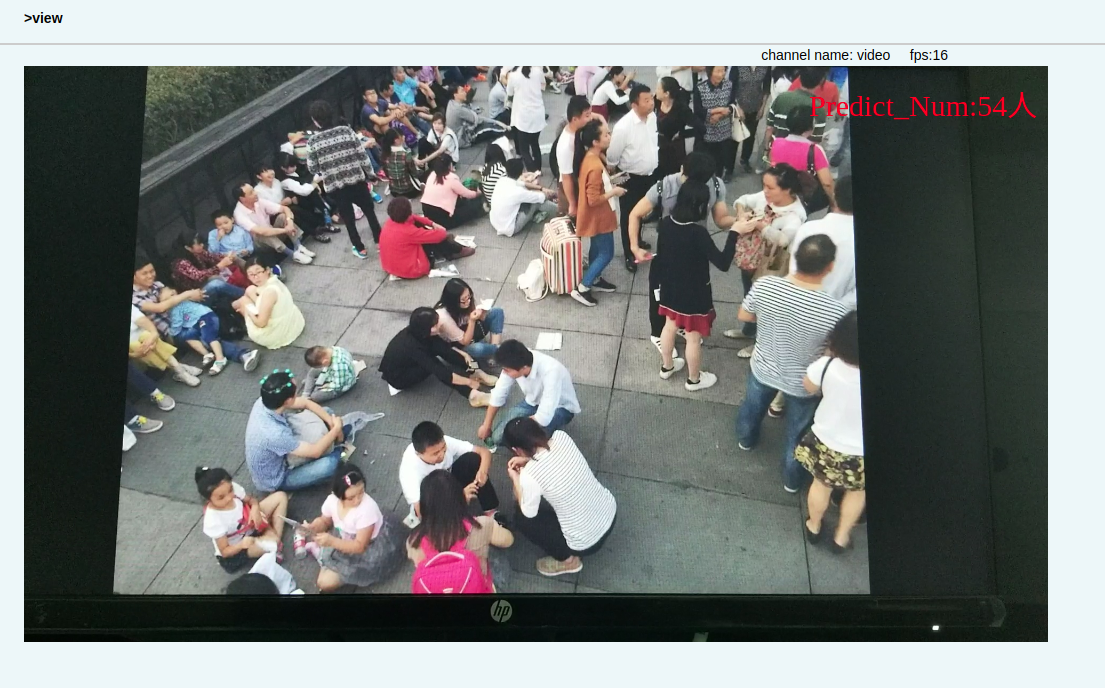


图3 推理结果展示图

可以看出，预测的人数54人和实际人数47人相差10个人以内，比较准确。

## 5 关键代码说明

### 5.1 视频解码部分核心代码

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

crowd\_counting\_app = crowd\_counting.CrowdCountingInference()

crowd\_counting\_app.clientsocket = client.PresenterSocketClient(("192.168.1.122", 7006), 5, None)

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

thread\_1.setDaemon(True)

thread\_1.start()

time.sleep(0.1)

if crowd\_counting\_app.graph is None:

print("creat graph fail")

sys.exit(1)

channel\_manager = ChannelManager.ChannelManager()

data = channel\_manager.OpenChannel()

if crowd\_counting\_app.clientsocket is None:

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

exit()

crowd\_counting\_app.clientsocket.send\_data(data)

cap = cv2.VideoCapture(URL)

ret, frame = cap.read()

if mp4\_url:

try:

while ret:

crowd\_counting.dowork(frame, crowd\_counting\_app)

ret, frame = cap.read()

except Exception as e:

print("ERROR",e)

finally:

crowd\_counting\_app.dispose()

else:

rtsp\_queue = client.Queue()

sub\_thread = threading.Thread(target=crowd\_counting.sqEngine,args=(rtsp\_queue,crowd\_counting\_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()

crowd\_counting\_app.dispose()

### 5.2 人数统计推理的核心代码

class CrowdCountingInference(object):

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

# 实例化模型管理类

self.model = ModelManager.ModelManager()

self.width = 1024

self.height = 768

self.clientsocket = None

self.channel\_manager = ChannelManager.ChannelManager()

self.graph = None

self.\_getgraph()

def dispose(self):

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

def \_\_del\_\_(self):

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

def \_getgraph(self):

# 描述推理模型

inferenceModel = hiai.AIModelDescription('crowd\_counting', crowd\_counting\_model\_path)

# 初始化Graph

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

if self.graph is None:

print ("Init Graph failed")

'''

1.定义输入Tensor的格式

2.调用推理接口

'''

def Inference(self,input\_image):

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

self.height \* self.width \* 3)

nntensorList = hiai.NNTensorList(inputImageTensor)

#print ('crowd\_counting\_nntensorList.get\_tensor\_num() = ',nntensorList.get\_tensor\_num())

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

if not resultList:

print("Inference fail")

sys.exit(1)

#返回推理结果

return resultList

def GetDetectionInfo(self, resultList, input\_image):

if not resultList:

return None

detection\_result = []

sum = 0.0

sum = np.sum(resultList)

#for i in range(len(resultList[0][0])):

# for j in range(len(resultList[0][0][0])):

# sum = sum + resultList[0][0][i][j][0]

#print('sum = ', sum)

detection\_result.append([0, 0, 1, 1, "number", sum])

return detection\_result

def GetImageFrameData(self, deteInfo, input\_image):

dr = DetectionResult()

image\_frame = ImageFrame()

image\_frame.format = 0

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

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

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

image\_frame.size = 0

sum = deteInfo[0][5]

sum = int(sum)

#print('\*\*\*\*\*\*\*\*\*\*\*\*\*\*sum = ', sum)

dr.result\_text = "Predict\_Num : " + str(sum)

dr.lt.x = 2

dr.lt.y = 2

dr.rb.x = 3

dr.rb.y = 3

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

resultData = self.channel\_manager.PackRequestData(image\_frame)

return resultData

def dowork(src\_image, crowd\_counting\_app):

input\_image = cv2.resize(src\_image, (crowd\_counting\_app.width, crowd\_counting\_app.height))

resultList = crowd\_counting\_app.Inference(input\_image)

resultinfo = crowd\_counting\_app.GetDetectionInfo(resultList, input\_image)

all\_data = crowd\_counting\_app.GetImageFrameData(resultinfo, input\_image)

if all\_data:

crowd\_counting\_app.clientsocket.send\_data(all\_data)

### 5.3 人数统计的后处理核心代码

def dowork(src\_image, crowd\_counting\_app):

input\_image = cv2.resize(src\_image, (crowd\_counting\_app.width, crowd\_counting\_app.height))

resultList = crowd\_counting\_app.Inference(input\_image)

resultinfo = crowd\_counting\_app.GetDetectionInfo(resultList, input\_image)

all\_data = crowd\_counting\_app.GetImageFrameData(resultinfo, input\_image)

if all\_data:

crowd\_counting\_app.clientsocket.send\_data(all\_data)

def sqEngine(rtsp\_queue,crowd\_counting\_app):

while True:

frame = rtsp\_queue.get()

if frame is None:

time.sleep(0.1)

continue

dowork(frame,crowd\_counting\_app)

## 6 重要问题及解决

问题1：presenter server上fps展示过低

解答：对推理得到的数据结果处理时不应该使用双重for循环，而是应该使用numpy自带的求和函数；以及由于原图过大，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

密集人群计数可以针对目前世界各地发生的大规模踩踏事件进行预防，从图像或者视频中准确估计人群已成为计算机视觉技术在人群控制和公共安全中日益重要的应用。在某些情况下，如公众集会和体育赛事，参与人数或密度是未来赛事规划和空间设计的重要信息。好的人群计数方法也可以扩展到其他领域，例如，从显微图像中计数细胞或细菌，野生动物保护区中的动物人群估计，或估计交通枢纽或交通堵塞的车辆数量等。