机器学习工程师Nanodegree

卡普斯通项目

拉希姆·卡塞纳6
月12日 th ，2019

I. 定义

项目概况

无人机以各种方式影响着我们的社会。 我们现在可以完成以前不可能完成的任务，或者需要大量的人类干预。 他们完全改变了我们拍摄照片或收集偏远地区信息的方式。 大多数无人机都是人类控制的，但随着人工智能的兴起，我们正在进入一个全新的世界。

在这个项目中，我创建了一个无人机，能够根据它所看到的跟踪一个人。 使用无人机摄像头，它能够检测到一个人，并相应地移动到那个人的位置。 除了相机本身，没有其他传感器被使用。

问题陈述

为了使无人机能够执行这样的动作，它必须能够有两个功能。 我会的
通过将其与人类逻辑推理进行比较来说明这一点。

让我们假设A的人想走向B的人。 首先，A人必须在他的环境中找到B人，并能够知道他在哪里。 让我们调用这个函数对象检测。 一旦发现感兴趣的对象，人A必须“激活”他的腿开始移动到人B。 在行走时，人A与人B的相对位置正在不断变化，因此如果他走得太慢或走得太多，就必须调整。 让我们把这个函数称为运动控制。

无人机有同样的问题需要解决，物体检测和运动控制。

让我们谈谈这两个问题及其各自的解决方案：

- 目标检测：无人机配备720p摄像机。 使用人工神经网络，特别是卷积网络，我们可以检测图像中物体的存在。 最近的体系结构如YOLO，我们甚至有一个额外的信息，告诉我们该对象位于图片中的位置。 重要的是要明白，计算机必须有一种方法来计算他在感兴趣的物体上的位置，才能知道无人机应该去哪里。
- 运动控制：一旦我们知道物体的位置是图片，我们必须告诉无人机移动，以便将该物体集中在他的相机中。 我们可以命令无人机顺时针，逆时针，向上，向下，向前或向后..

标准

用两个度量来确定解决方案模型：

- 无人机位置相对于物体中心的准确性。 无人机坐标与目标的平方距离..
- 时收敛到目标..

我选择这两个指标是因为它们代表了人类可以观察到的东西。 当无人机采取行动时，我们可以观察到它到达那里的速度有多快，当它到达最后的位置时，我们还可以观察到它对感兴趣的物体有多中心。 这个项目就像一个自拍无人机，在那里我们想中心一个人，同时做它很快。

II. 分析

数据探索和可视化

本节将分成两个小节，一个用于对象检测数据，另一个用于运动控制。

目标检测-VOC数据集

帕斯卡VOC挑战是一个非常流行的数据集，用于构建和评估图像分类、对象检测和分割的算法。

数据集包含20个对象类：.

- 个人

- 鸟，猫，牛，狗，马，羊
- 飞机，自行车，船，公共汽车，汽车，摩托车，火车
- 瓶子，椅子，餐桌，盆栽，沙发，电视/显示器

每个记录由一个图像和一个包含有关它的信息的xml文件组成。 注释是由此数据集的创建者手动完成的。

关于模型如何使用这些信息的详细信息将在后面的章节中描述。

这里是注释xml文件的示例：.

```
<注释>
                <文件夹>培训</文件夹>
                <文件名>000001.png</文件名>
                <路径>/my/path/Train/000001.png</path>
                 <来源>
                        <数据库>未知</数据库>
                 </来源>
                <尺寸>
                        <宽度>224</宽度>
                        <高度>224</高度>
                        <深度>3</深度>.
                </尺寸>
                <分段>0</分段>
                 <对象>
                        <姓名>21</姓名>.
                        <姿势>正面</姿势>
                        <截断>0</截断>
                        <困难>0</困难>
                        <闭塞>0</闭塞>.
                        <bndbox>
                               <xmin>82</xmin>.
                                <xmax>172</xmax>.
                                <ymin>88</ymin>.
                                <ymax>146</ymax>.
                        </bndbox>
                 </物体>
        </注释>
```

每个xml文件都是指一个图像。 一个xml文件将列出图片中存在的所有类，并清楚地指示图片中对象的包围框的坐标，如下图1所示。

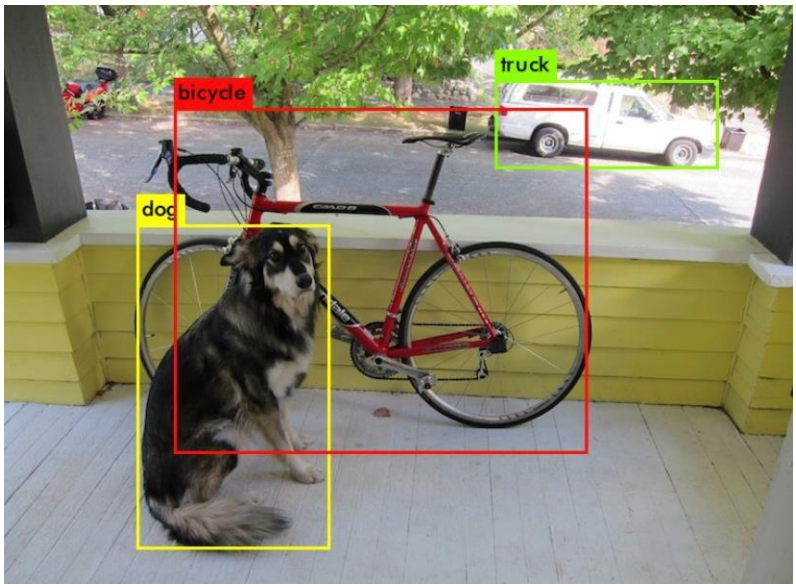


图1：对象检测

下表显示了关于不同类的统计数据：

表1：VOC数据集统计

	train		val		trainval		test	
	Images	Objects	Images	Objects	Images	Objects	Images	Objects
Aeroplane	327	432	343	433	670	865	-	-
Bicycle	268	353	284	358	552	711	-	-
Bird	395	560	370	559	765	1119	-	-
Boat	260	426	248	424	508	850	-	-
Bottle	365	629	341	630	706	1259	-	-
Bus	213	292	208	301	421	593	-	-
Car	590	1013	571	1004	1161	2017	-	-
Cat	539	605	541	612	1080	1217	-	-
Chair	566	1178	553	1176	1119	2354	-	-
Cow	151	290	152	298	303	588	-	-
Diningtable	269	304	269	305	538	609	-	-
Dog	632	756	654	759	1286	1515	-	-
Horse	237	350	245	360	482	710	-	-
Motorbike	265	357	261	356	526	713	-	-
Person	1994	4194	2093	4372	4087	8566	-	-
Pottedplant	269	484	258	489	527	973	-	-
Sheep	171	400	154	413	325	813	-	-
Sofa	257	281	250	285	507	566	-	-
Train	273	313	271	315	544	628	-	-
Tvmonitor	290	392	285	392	575	784	-	-
Total	5717	13609	5823	13841	11540	27450	-	-

对于这个项目，只有人类才会感兴趣。 正如我们在“训练”部分为“人”类，我们有4087张图片，其中包含8566人。

运动控制-环境

从无人机接收到的相框分辨率为960x720像素。 框架的中心总是x=480，y=360。

根据人在图片中的位置，我们可以计算模型找到的包围框的中心。

利用屏幕中心和包围盒中心之间的差异，我们得到了一个相对位置，我们可以用来训练强化学习代理。 请注意，x和y只跟踪无人机的方向和高度相对于目标。 使用x和y无法找到对象之间的距离(Z)。 为了获得距离的度量，我使用了包围盒的面积。 当物体越来越靠近时，盒子的面积会变大，反之亦然。 强化学习代理只被训练为给出x，y命令。 距离(Z)组件是单独控制的，有一个简单的if条件为本项目。

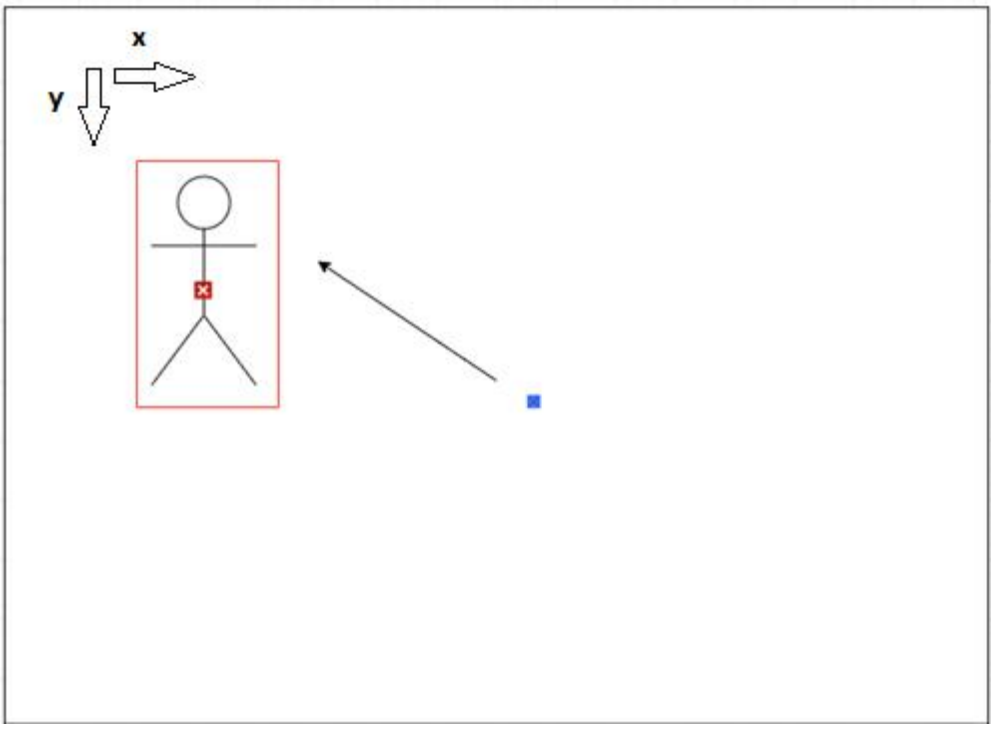


图2：无人机环境

对于培训阶段，我使用了一个具有以下参数的环境：

- 观测空间：x=0，y=0➔ x=960，y=720
- 连续动作空间：-60<A1<60和-60<A2<60。
- 动作1控制偏航运动，动作2控制高度上下运动..
- 无人机支持每个方向的速度从-100到100。
- 我定义了areap，它是包围框与全摄像机屏幕(are ap=面积/（960*720)）*100所占面积的百分比

算法和技术

同样，本节将分成两个小节，一个用于对象检测问题，另一个用于运动控制。

对象检测-YOLOV3体系结构

以下信息是JonathanHui关于YOLO算法的伟大文章的摘录[1] 有关YOLO的更多信息，请参阅参考链接。

你只看一次(YOLO)是一个目标检测系统，用于实时处理..

YOLO将输入图像划分为S×S网格. 每个网格单元只预测一个对象。 对于每个网格单元，

- 它预测B边界框，每个框有一个框置信度分数，
- 它只检测一个对象，而不考虑框B的数量，
- 它预测C条件类概率（对象类的相似性每类一个）。

为了评估VOC数据集，YOLO使用7×7个网格(S×S)、2个边界框(B)和20个类(C)。 每个边界框包含5个元素：(x，y，w，h)和一个框置信度评分。 置信度分数反映了框包含对象的可能性和边界框的准确性。 每个单元格有20个条件类概率。 总之，YOLO的预测具有(S，S，B×5+C)=（7，7，2×5，20)=(7，7，30）的形状。

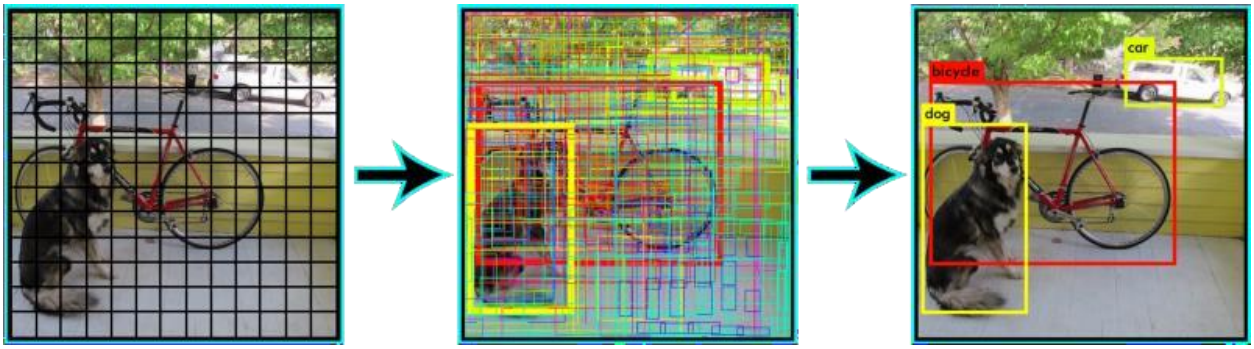


图3：YOLO预测步骤.

YOLO将采取每个7x7窗口，并发布每个类和框坐标的预测。 一旦所有的图片被看到，我们保持那些高框置信度分数（大于0.25)作为我们的最终预测(正确的图片）。 每个预测框的类置信度分数计算为：.

班级置信度分数=盒子置信度分数x条件班级概率.

它测量分类和定位（对象所在的位置）的置信度。

下表介绍了YOLOV3实现的体系结构：

表2：YOLOV3体系结构

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

YOLO是一种经过验证的复杂算法. 我们可以看到，在COCO数据集上，它的表现超过了他的大多数竞争对手：

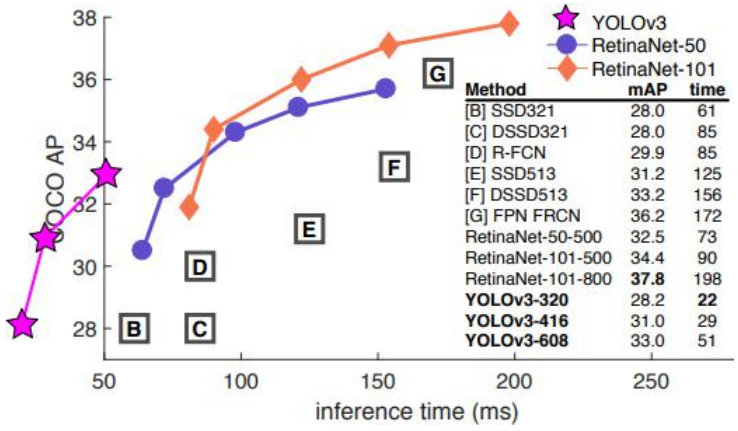


图4：YOLOV3的性能

运动控制-DDPG代理

“深度确定性策略梯度”(DDPG)是强化学习中的策略梯度算法。 它使用了如图3所示的演员批评模型。

简单地说，我们有一个演员模型，尝试不同的行动，被批评模型评级。 当选择该动作以允许探索和发现可能被忽略的状态时，会增加一些噪声。

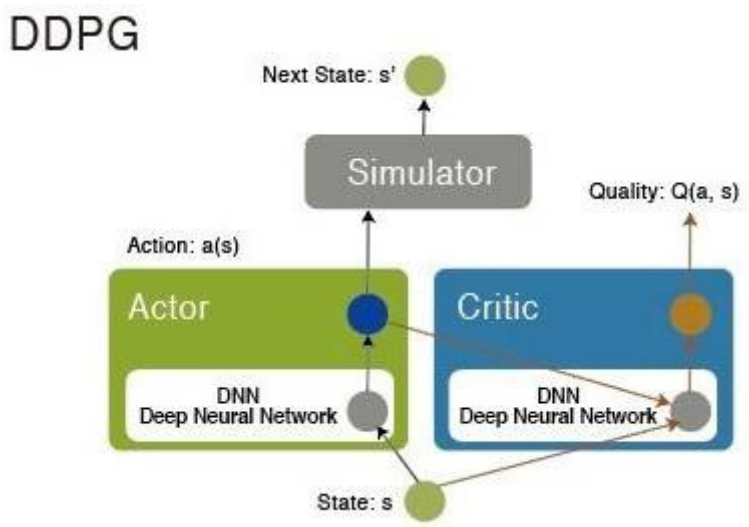


图5：DDPG算法

为了训练这个代理，我们需要定义一个奖励函数，让模型知道它是否朝着正确的方向发展。 奖励函数将作为输入检测到的包围框的坐标x和y。 奖励功能的详细信息将在后面的章节中添加。

以下是培训期间可用的超参数：

- 演员和评论家学习率：演员和评论家神经网络中的梯度学习率
- tau=定义了目标网络更新的速度
- 伽玛=折扣因子
- 泰塔=勘探噪音
- 亩=勘探噪声
- 西格玛=勘探噪声
- nb_max_episode_steps：一集前的步骤数被视为完成
- 集数

基准

建议中描述的基准是一个控制无人机的人。

提交的"视频/Benchmark_test.mp4"视频文件显示一名人类飞行员试图跟踪图片中的人类。 从人类被检测到的那一刻（红色的盒子）到它以相机为中心的时间大约需要15秒。

最后的立场是：.

x=500和y=420，area_p=20

我做这个测试的速度尽可能快，方向变化最小。

如果我们将其与x=960，y=720的目标位置进行比较，这些结果在x上和y上的误差分别为4%和16%。

III. 方法

本节将分成两个小节，一个用于对象检测问题，另一个用于运动控制。

数据预处理-对象检测

VOC数据集本身已经干净，每个图像都与注释文件相关联。

在YOLO网络的输入下，将以下预处理功能应用于图像：.

```
def preprocess_input(image, net_h, net_w):
    new_h, new_w, _ = image.shape

    # determine the new size of the image
    if (float(net_w)/new_w) < (float(net_h)/new_h):
        new_h = (new_h * net_w)//new_w
        new_w = net_w
    else:
        new_w = (new_w * net_h)//new_h
        new_h = net_h

    # resize the image to the new size
    resized = cv2.resize(image[:,:,::-1]/255., (new_w, new_h))

    # embed the image into the standard letter box
    new_image = np.ones((net_h, net_w, 3)) * 0.5
    new_image[(net_h-new_h)//2:(net_h+new_h)//2, (net_w-new_w)//2:(net_w+new_w)//2, :] = resized
    new_image = np.expand_dims(new_image, 0)

    return new_image
```

每个图像被调整到神经网络的输入大小，从最小输入大小到配置中定义的最大输入。每个像素值除以255，最后将得到的数组嵌入到字母框中。

数据被分割，80%用于培训，20%用于验证。

数据预处理-运动控制

在代理的训练过程中，生成边界框的随机位置，唯一的预处理是确保这些值在图片边界内如下：.

$$0<x<960和0<y<720$$

实现-对象检测

YOLO原始模型建立在C++中，只能通过命令行执行。 由于我想更深入地理解这个体系结构，所以我重用了一个YOLOv3Keras实现[3]

它是一个实现，具有训练和预测功能以及操作超参数的配置。

培训和完善

这是描述用于培训的不同参数的YOLO配置文件：

- 图像输入大小
- batch_size
- learning_rate
- nb_epochs

最小输入大小为220到最大416：

新纪元	批次大小	学习率	平均精度	早停（时代）
100	8	1e-3	0.51	20
100	16	1e-4	0.58	12
100	16	1e-5	0.55	23

最小输入大小为416：

新纪元	批次大小	学习率	平均精度	早停（时代）
100	8	1e-4	0.71	7

我使用GoogleCloud上的虚拟机只在“Person”类上训练这个模型，因为这是唯一的一个，我对这个项目感兴趣。 每次训练练习根据输入大小从4小时到24小时不等.. 我在我的VM上使用一个可预付费GPU，这样我的训练就会被打断，我必须重新开始。 在给出它之后，几次尝试使用描述的超参数，并将输入大小更改为416，我得到了0.71，这对于这个项目来说是足够体面的。 然而，我决定在整个20个类VOC数据集上使用预先训练的权重，它在人类上提供了稍微好一点的0.78平均精度。 唯一的调整是过滤掉“人”类的预测。

一旦训练完成，我就把重量从谷歌云虚拟机导入到我的笔记本电脑，它有一个中等的图形卡。 在GTX1051Ti上运行YOLO算法提供了大约250ms的检测时间，这对于实时项目来说是缓慢的。 为了防止检测减慢主循环，我从一个通过队列发送检测到的框的并行线程中执行了YOLO算法。 这些检测到的盒子出现在大约每250毫秒每秒帧在一个25FPS视频馈送。

实现-运动控制

无人机是通过Wifi使用UDP命令控制的。 我从达米恩·富恩特斯那里得到了我的密码 [3] GitHub，特洛无人机的python包装器。 我不会深入了解这个图书馆的细节，但它允许我们通过每50ms发送命令来控制无人机。 我修改了代码并调整了一些参数，以满足我对这个项目的需求。 每个方向可以用-100到100之间的值来命令。

培训和改进-模拟：

为了使代理可以训练，它需要一个具有基本功能步骤和重置的环境。 我重用了开放AI健身房的“连续山车”环境。

步骤功能：.

- 用输入操作更新位置以生成下一个状态。
- 计算报酬
- 确定是否最终状态（已完成）
- 返回下一个状态，奖励和完成状态重置功

能：

- 将位置重置为边界内的随机值。
- 初始化内部状态，已完成状态

对于DDPG代理，我使用了keras-rl库[4].

我使用包围框中心和屏幕中心（480，360）的平方距离(Dist)作为奖励函数的度量。

最终奖励功能：.

- >100➔ 奖励-=发行*0.25
- 发行<100➔ （100分）

这些是完成一集的条件：

- 步数超过10步.
- x或y失界(960x720以上)

为了获得好的结果，我不得不对奖励函数进行了大量的调整。 这主要是试错。 它确实帮助绘制了边界框的位置在训练期间使用健身房可视化功能。

下图显示了不同的尝试与完成的条件和距离。 橙色显示奖励移动平均超过100步。

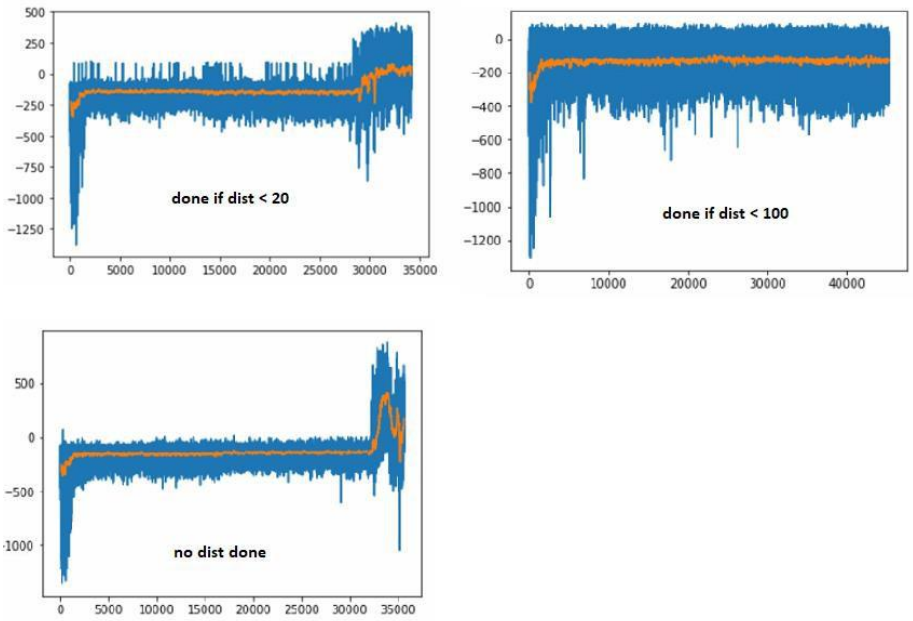


图6：奖励与步骤

当我添加了一个带有距离要求的已完成状态时，模型没有探索低于该数字的状态。 我需要模型能够选择行动，甚至真正接近目标。 我最终选择不添加一个已完成的状态与距离。

以下是训练期间使用的最终超参数：

- 演员和评论家学习率：LR=。 001
- 头=1e-3
- 伽玛=0.99.
- 泰塔=0.15
- 亩=0
- 西格玛=0.3
- 第10nb_max_episode_steps=
- 发作次数：10000次

距离控制

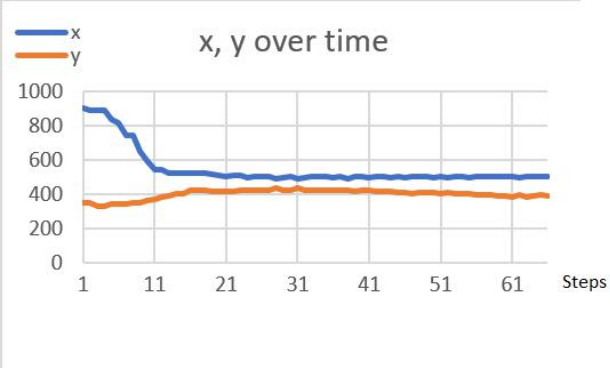
为了控制距离，我使用边界框的面积作为度量。 随着物体的靠近，包围盒变得更大，所以这是一个很好的指示无人机是如何接近目标。 如前所述，我用我简单的if条件来控制这个：

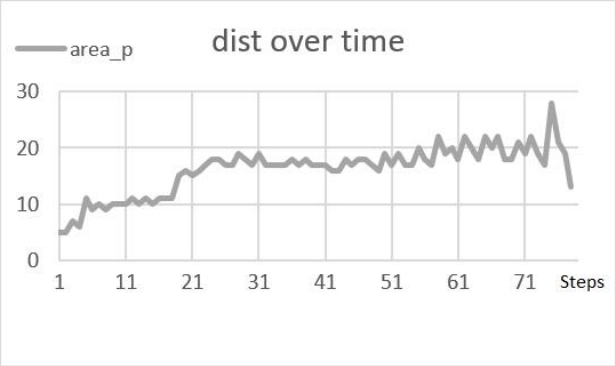
- 如果面积低于总屏幕尺寸的25%➔ 往前走
- 如果面积超过总屏幕尺寸的50%➔ 向后退
- 如果面积在25%到50%之间➔ 别动。

IV. 结果

模型评估和理由

我意识到在与基准完全相同的条件下进行了测试。 请参阅说明此测试的视频文件“视频/real_test.mp4”。 无人机迅速收敛到目标（7秒以下）。 最终误差在x上为4.3%，在y上为11%。 距离不是根据基准来衡量的，但我们可以看到，它是否低于前面定义的25。 我们可以说，这种模式比人类特工表现得更好。 我真的很满意这些结果，即使延迟造成了很多问题。





由于硬件引入的延迟，结果对无人机的速度很敏感。 我发现最好的最大速度是60。

由于硬件的限制，无人机的行为并不完美，但结果仍然令人印象深刻，无人机实际上能够识别一个人并跟踪它。

V. 结论

我在提交中添加了一个免费的测试视频文件(视频/free_test.mp4)。 我在不同的条件和角度下测试了无人机。

正如我们在视频中看到的，对象检测是准确的，但由于我的图形卡不好，有一些延迟。 无人机很快就会聚到目标上，但由于反应缓慢，无法实时调整，经常越过目标回来。 无人机有时在目标上振荡，试图尽可能地最大化回报。 如果我们消除了抖动，因为结果是满足这一水平的硬件。

反思

本项目运行情况可总结如下：.

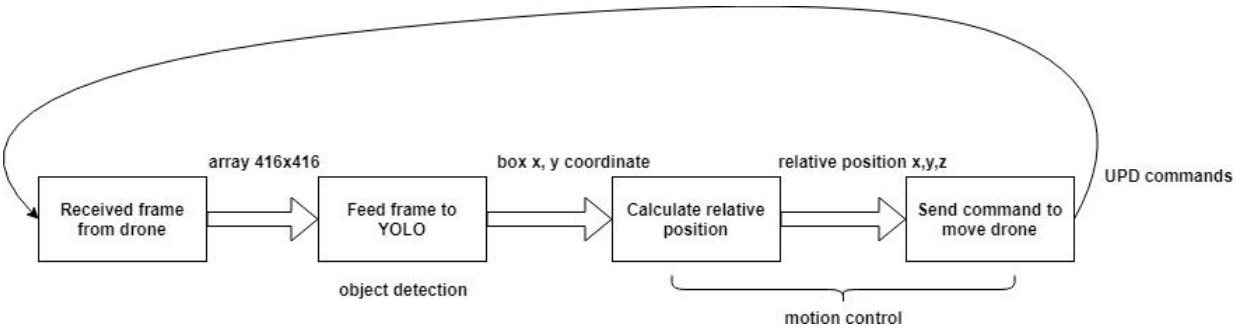


图7：系统流程图

其中一个困难的部分，如果这个项目是设置。 因为无人机没有
足够的处理能力，“大脑”必须在笔记本电脑上，这会带来延迟。

本项目教授了如何使用Linux云计算机器，以及如何为AI培训目的设置一个完整的CUDA-Keras GPU环境。

看到YOLO算法是如何执行的，以及它如何改变了目标检测项目的游戏，这是令人印象深刻的。

使用强化学习对于这个项目来说似乎有点过火，因为它可以用简单的if条件来完成。DDPG算法基本上建立了一个将相对位置与相应动作关联起来的表，它很有趣地看到了使用keras-rl库实现它是多么容易。

改进

我用了100$无人机做这个项目。 这个项目可以得到更好的结果与高精度的无人机。 如前所述，无人机和笔记本电脑之间的延迟确实影响了这个项目，所以这里有两个方面可以改进这个项目：

- 实时无人机，即更高的网络速度和计算速度。
- 更好的图形卡在笔记本电脑上，以减少推断时间的检测。

随着这两个变化，无人机将更快地知道一个动作的效果。 目前，它需要大约0.5秒的反馈来，这是不被认为是实时的。

参考资料

[1] *YOLO算法解释： https://medium.com/@jonathan_hui/real-time-object 检测-用-yolo-yolov2-28b1b93e2088

[2] *Keras YOLOv3实现：Damien Fuentes Tello SDK：

https://github.com/experiencor/keras-yolo3

https://github.com/damiafuentes/DJITelloPy

[4]：Keras强化学习图书馆 https://keras rl.readthedocs.io/en/latest/agents/ddpg/