Real Time Human Detection & tracking via a quadcoptor and SSD-MobileNet neural network

Daksh Shukla, Ben Liu, Patrick McNamee, Tim Fox, Caio Vigo

OUTLINE

- Background & motivation
- System architecture – hardware
- System architecture – software
- ROS architecture - 2 packages communication
- Neural network architecture
- Controller
- Challenges - implementation (frequency, TF+ROS, Oscillations)
- Non drone videos - lab testing
- Real time drone video – outdoor flying
- Limitations & future work
- References

Background and motivation

What We Would Like To Do?

We want to be able to **track** and **follow a human being** using low cost drone for surveillance applications.

What Is The Challenge?

We want to do it in real-time.

We need to deal with **real-time computation costs** related to **vision processing**.

- Classify a human from real-time drone images
- As the person moves, send commands to the drone to follow the classified object (human)
- Algorithms (neural networks) need to interact with other operating systems (drone, ROS, etc.)

How ML can help?

Neural networks are well-suited for object detection and classification within images.

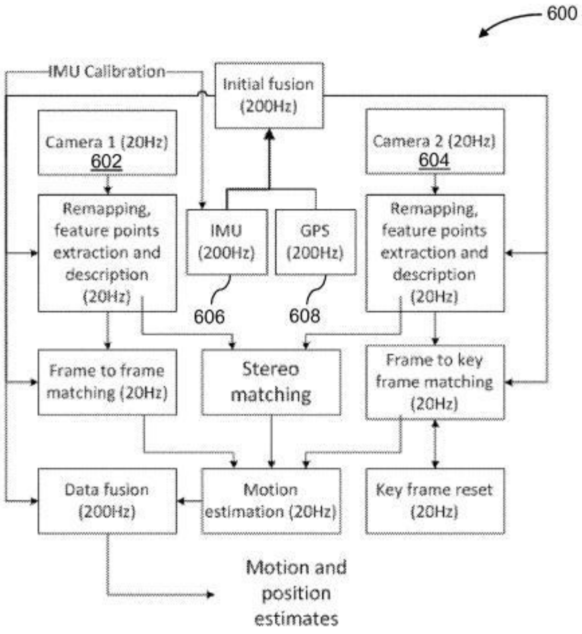
Deep neural networks (DNNs)

Due to their inherent parallel processing and robust object detection characteristics provide a good framework for this type of application.

Who Is Currently Doing Similar Things?

- DJI and others use two cameras for stereo matching [1]
  - Part of obstacle avoidance and position tracking
  - Doesn't classify objects

Cameras for stereo matching

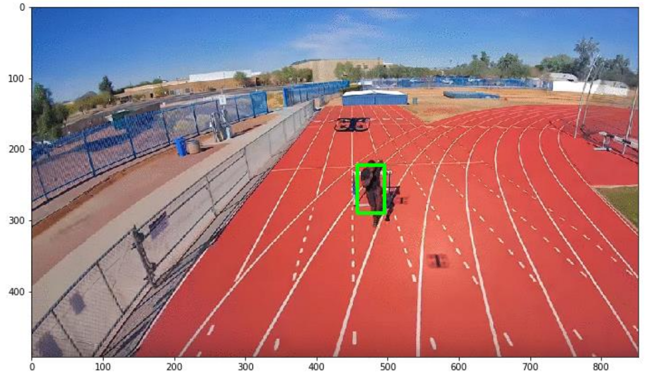


[1] *U.S. Patent No. 10,240,930*. (2019). Washington, DC: U.S. Patent and Trademark Office.

Who is currently doing similar things?

- Researchers have used SLAM (**simultaneous localization and mapping**) techniques to recognize objects (no classification) in the environment to follow humans [1].

- Norman Di Palo has a good guide to human tracking using tensor for a quadcopter [2]. Unsure if he has implemented it.



[1] K. K. Lekkala and V. K. Mittal, "Simultaneous aerial vehicle localization and Human tracking," *2016 IEEE Region 10 Conference (TENCON)*, Singapore, 2016, pp. 379-383. doi: 10.1109/TENCON.2016.7848025

[2] https://medium.com/nanonets/how-i-built-a-self-flying-drone-to-track-people-in-under-50-lines-of-code-7485de7f828e

System architecture – hardware

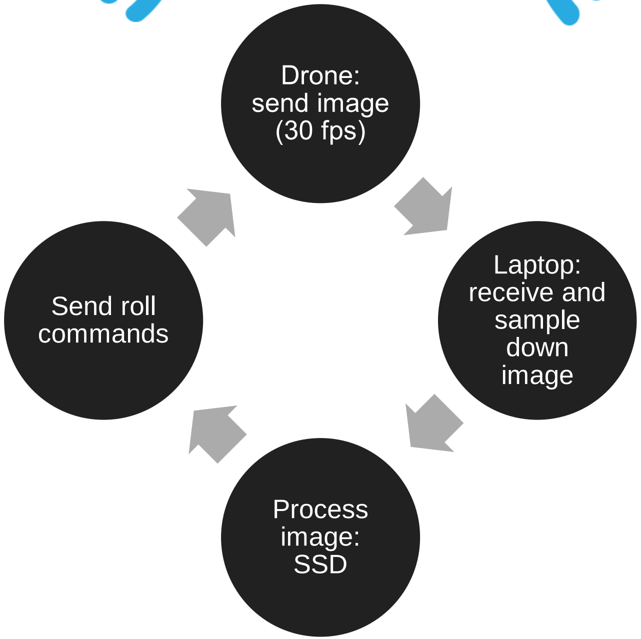
Parrot AR Drone 2.0

Characteristics:

- Front camera: 720p
- WiFi connection: acts as router
- Inertial measurement unit: Gyroscope, Accelerometer, Magnetometer
- Altitude ultrasound sensor
- Vertical (bottom) camera
- Take-off weight: 0.93 pounds

System architecture - hardware

Parrot AR DRONE 2.0



System architecture – software

An Overview of the Process

- Robot Operating System (ROS) software on Ubuntu 16.04
- ROS Kinetic Kame version
- Ardrone autonomy - ROS driver for the drone

  --> *Autonomy lab - Simon Fraser University*

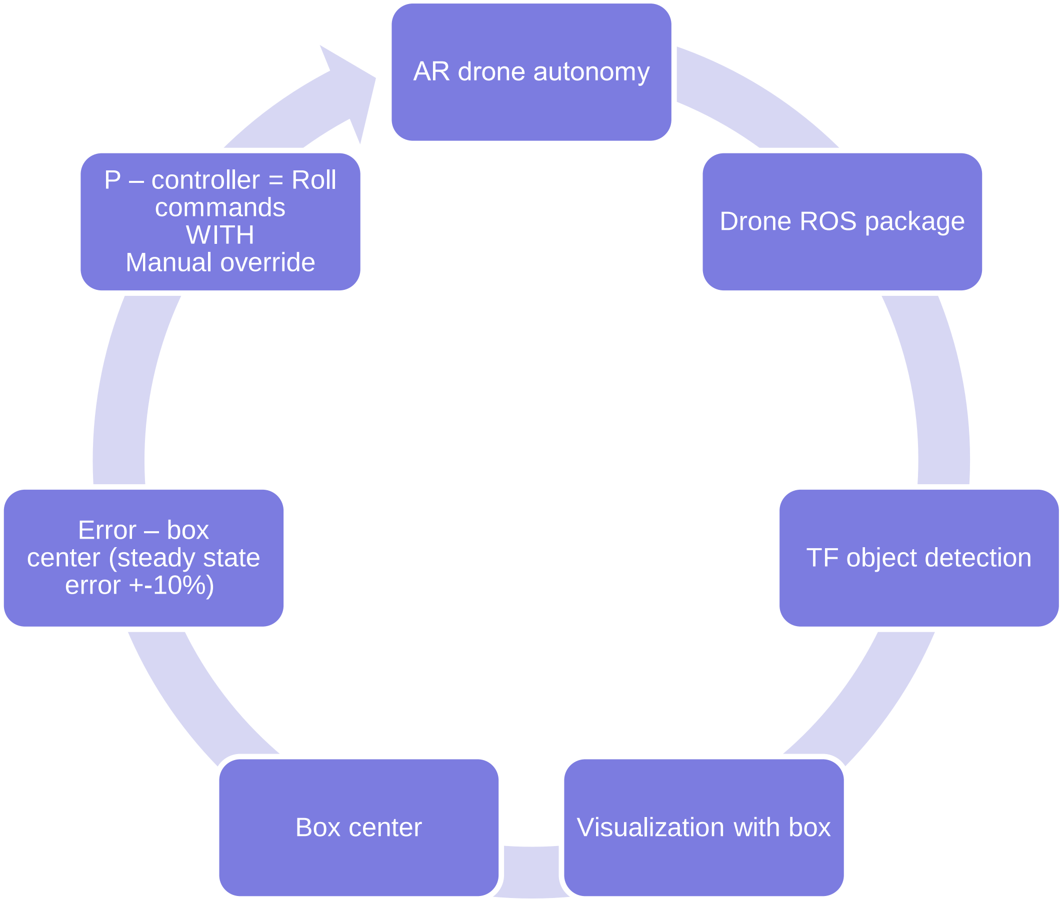
  --> Publishes navigation data and sends commands
- Ardrone tutorials ROS package

  https://robohub.org/up-and-flying-with-the-ar-drone-and-ros-getting-started/

  --> Uses Ardrone autonomy driver to subscribe to sensor data
- Tensorflow object detection API

  https://github.com/tensorflow/models/tree/master/research/object_detection

System architecture - software



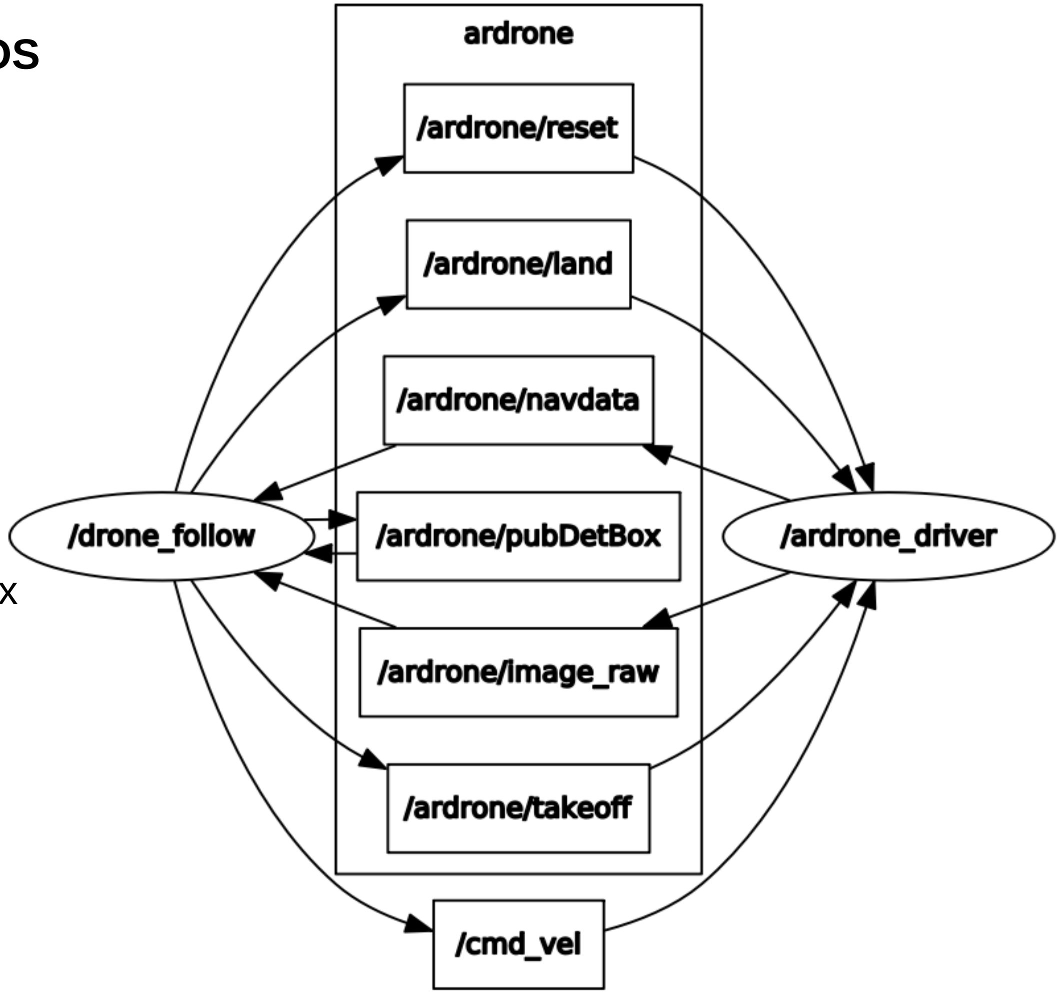
ROS architecture - 2 packages communication

ROS rqt graph:

Visualization of ROS nodes communication

ROS rqt graph: Visualization of ROS topics communication – data exchange

1. Subscribe to image
2. Pass every "nth" (6th) image
   frequency = 30/n (= 5 Hz)
3. Convert ROS image to CV image
4. Detect classes = ANN(image)
5. If class == 1 && det_score > 30%
      find max score index
6. Use index
   - get box = [y_min, x_min,y_max, x
7. _max]
   - visualize(image + box)
   - Make ROS message
   - Publish ROS topic
8. Subscribe to box data
9. Compute error w.r.t center
10. Compute & Publish roll commads



Neural networks architecture

Net Architecture

- We used the pre-trained model from TensorFlow object detection API, ssd_mobilenet_v1_coco_2017_11_17
- 3,191,072 parameters
- It was pretrained on Microsoft COCO (Common objects in context) data set (available at http://cocodataset.org/#home)
  Training images = 200,000
  Validation images = 8000
- Training algorithm: RMSProp (Root Mean Square Propagation)
  batch size = 32
  learning rate = 0.004
  learning rate decay = 0.95 every 800k steps

J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama et al., *Speed/accuracy trade-offs for modern convolutional object detectors*, 2016
T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ra- manan, P. Dolla ŕ, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 1 May 2014.1, 4

Net Architecture-SSD

- **SSD** stands for **Single Shot MultiBox Detector**
- **Feed forward network:** image through one-time step
- Convolutional feature layers at the end of the base network and they get smaller as the layers move forward, this allows for predictions at different scales
- Default bounding boxes at each feature map location, the SSD will make 4 box predictions and keep the highest score
- SSD will name a ‘positive match’ if the intersection over union is above 50 percent
- SSD is generally fast and accurate
  58 FPS with mAP 72.1% on VOC2007 test
  Faster R-CNN: 7 FPS with mAP 73.2%
  YOLO: 45 FPS with mAP 63.4%

W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, A.C. Berg, Ssd: Single shot multibox detector, 2015

Net Architecture-SSD

- SSD creates **many predictions** for all the objects, there are many negative scores also and the model can hurt from too many negative scores

- SSD sorts negative scores, keeps only 3 negative scores for every positive score
  --> keeps the negative scores so that the model can know what a ‘bad prediction’ is

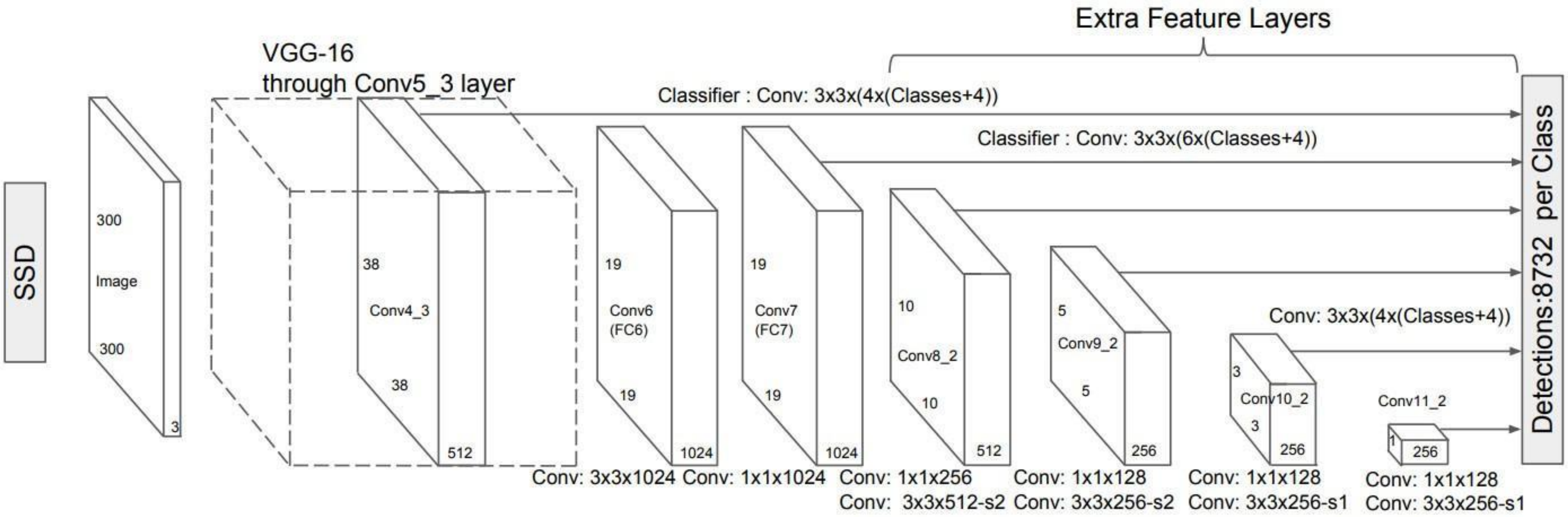
- SSD uses **non-maximum suppression** to avoid duplicate predictions on objects

- Loss function is as follows where N is the number of positive match and α is the weight for the localization loss

$$L(x, c, l, g) = \frac{1}{N}(L_{conf}(x, c) + \alpha L_{loc}(x, l, g))$$

W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, A.C. Berg, Ssd: Single shot multibox detector, 2015

Net Architecture-SSD



W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, A.C. Berg, Ssd: Single shot multibox detector, 2015

Net Architecture - MobileNets

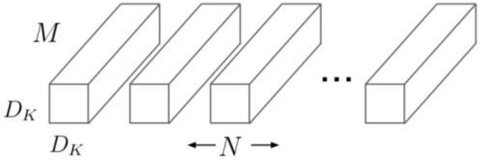
- **MobileNets** are specialized convolutional neural networks for mobile and embedded application

- **MobileNet** model is based on depthwise convolution and 1 × 1 convolution called a pointwise convolution.

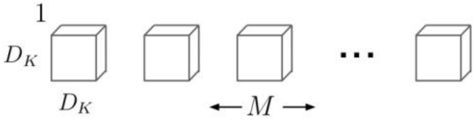
- The depthwise convolution applies a single filter to each input channel, whereas the pointwise convolution then applies a 1 × 1 convolution to combine the outputs the depthwise convolution

A. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

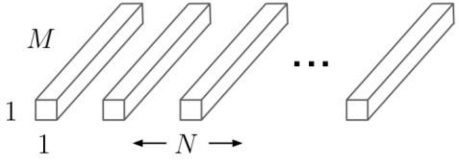
Net Architecture - MobileNets



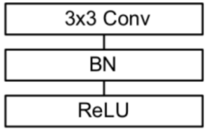
(a) Standard Convolution Filters



(b) Depthwise Convolutional Filters



(c) 1×1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution



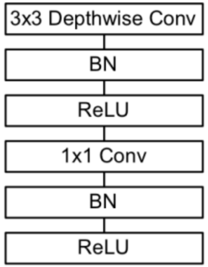


Figure 3. Left: Standard convolutional layer with batchnorm and ReLU. Right: Depthwise Separable convolutions with Depthwise and Pointwise layers followed by batchnorm and ReLU.

M: input channels,
N: output channels,
Dk × Dk: the kernel size

A. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

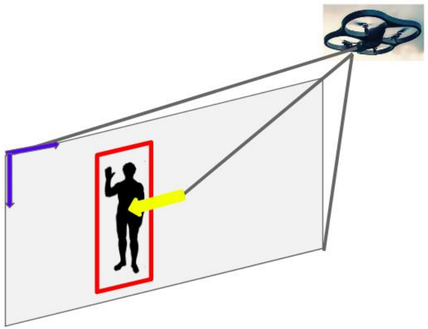
Controller

Control Algorithm

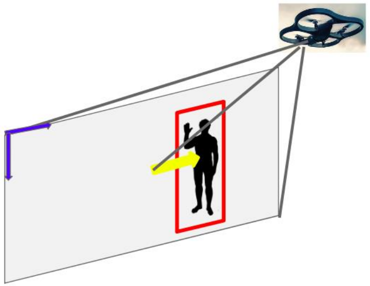
- Proportional controller:
  Output = K*Error
- Define error:
```
C = (x_min + x_max)/2
If C> 60% or C < 40%:
        error = (C - 50%)
Else:
        error = 0
roll = K_r*error
```

- The control law ensures that the person is laterally centered in the image with +- 10% steady error

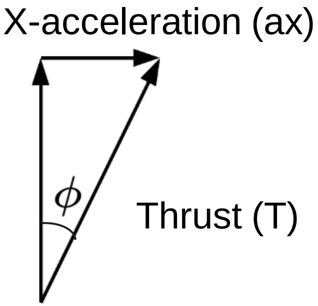
Roll Left



Roll Right



$$e_x = x - x_{ref}$$
$$\ddot{e}_x \approx T\phi$$
$$= Tk_P e_x$$



Challenges - implementation (frequency, TF+ROS, Oscillations)

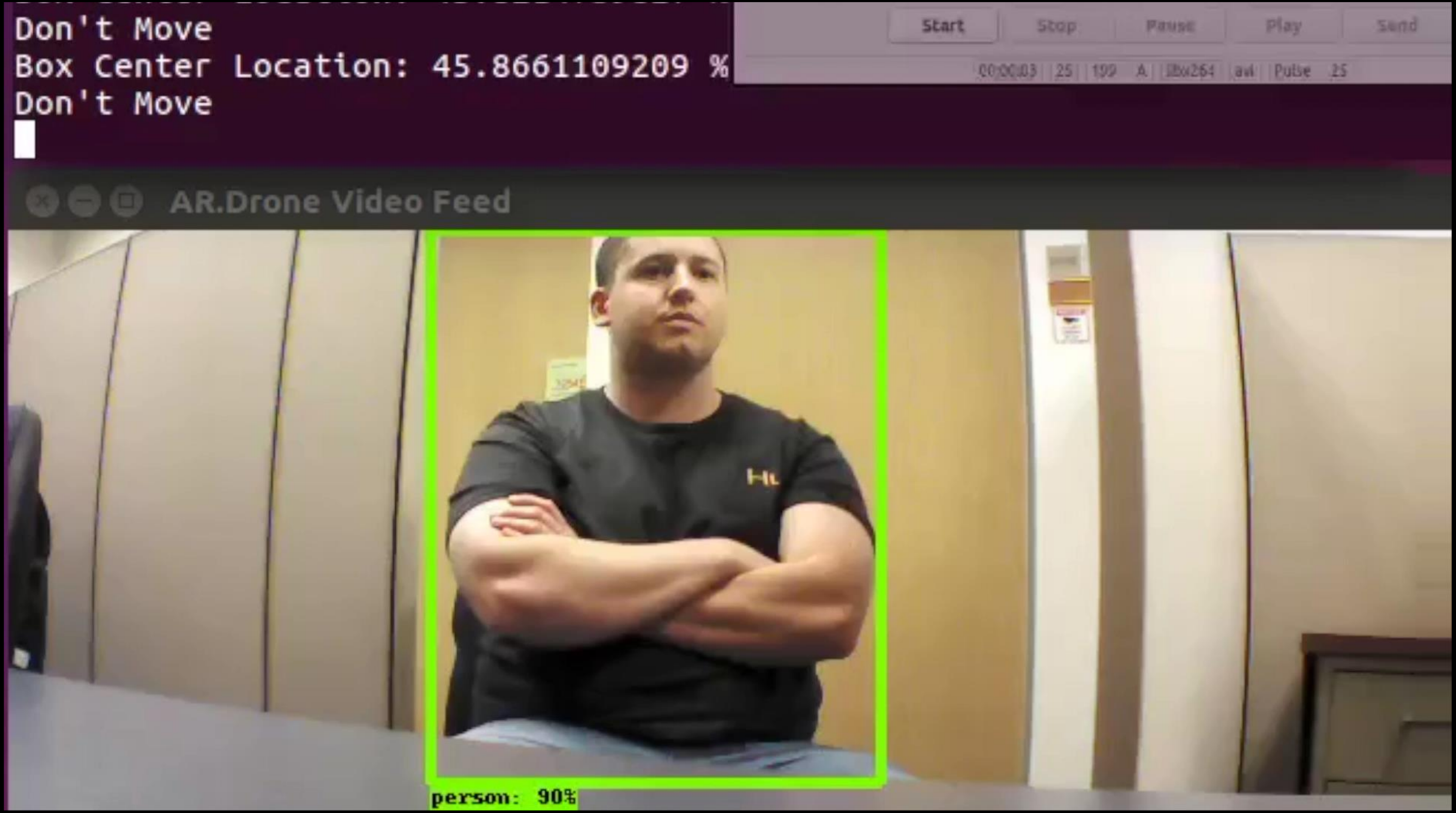
Challenges - implementation

- Frequency of image recording
  - Video over WiFi @ 30 fps --> sampled down to 5 Hz
  - Neural Net detection @ average 0.12 seconds ~ maximum 8.33 Hz frequency
- Tensorflow + ROS binding not perfect
- Roll command oscillations - tracking

Non drone videos – lab testing

+

Real time drone video – outdoor flying


Don't Move
Box Center Location: 45.8661109209 %
Don't Move
Start
Stop
Pause
Play
Send
AR.Drone Video Feed
person: 90%

Limitations & future work

LIMITATIONS & FUTURE WORK

Limitations:

- Drone is very sensitive to wind
- Loss of length reference
  - Using only 1 camera reduces everything to image plane
  - Area in image plan is highly attitude dependent

Future Work:

- Implement pitch controller (move forward & backward)
- Implement yaw controller based on orientation
- Yaw - Roll selection algorithm
- Implementation directly onto quadcopter

References

References

U.S. Patent No. 10,240,930. (2019). Washington, DC: U.S. Patent and Trademark Office.

K. K. Lekkala and V. K. Mittal, "Simultaneous aerial vehicle localization and Human tracking," 2016 IEEE Region 10 Conference (TENCON), Singapore, 2016, pp. 379-383. doi: 10.1109/TENCON.2016.7848025

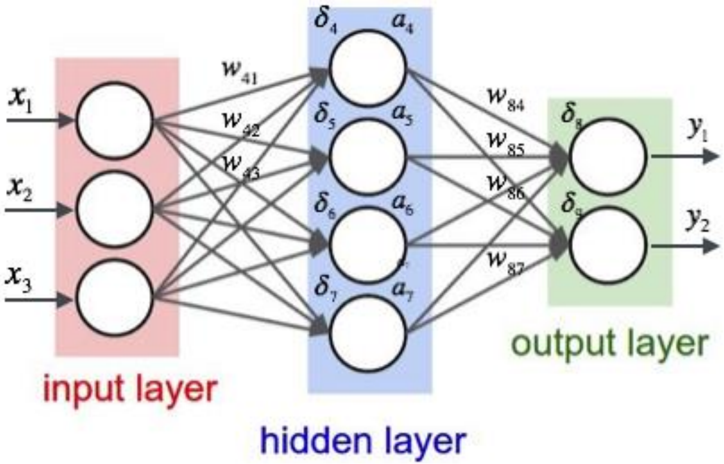
https://medium.com/nanonets/how-i-built-a-self-flying-drone-to-track-people-in-under-50-lines-of-code-7485de7f828e

W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, A.C. Berg, Ssd: Single shot multibox detector, 2015

A. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861, 2017.

DEMONSTRATION

Thank You



Questions

Reward Function?

IT DEPENDS!