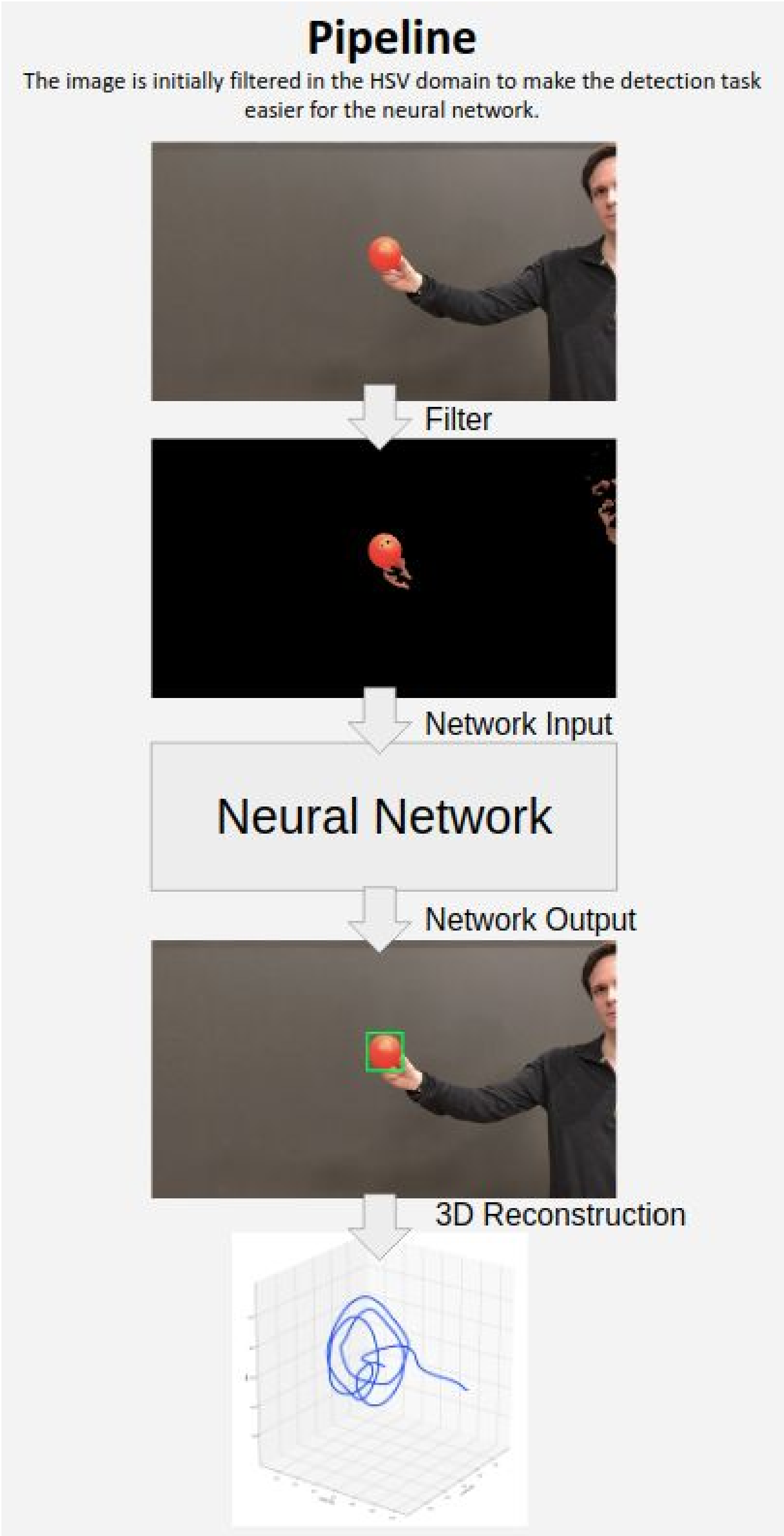
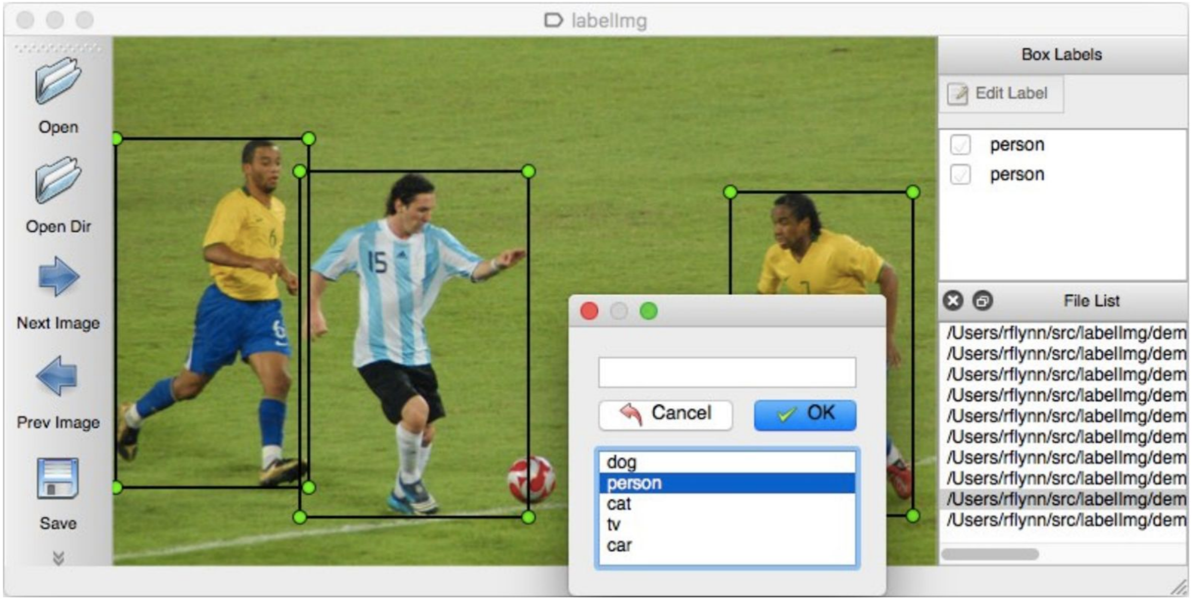
Final Project

Goal: Detect red ball in a video sequence and reconstruct the path of the ball in 3D world coordinates.



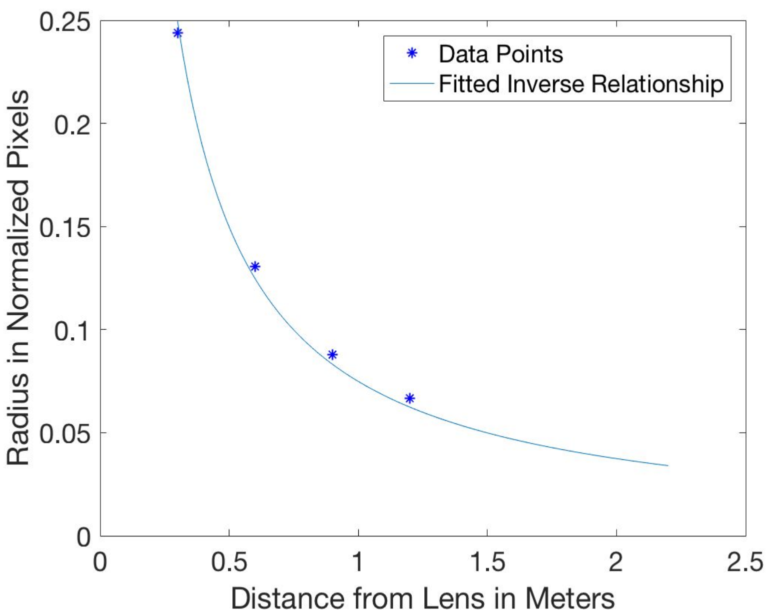
Annotation Procedure:



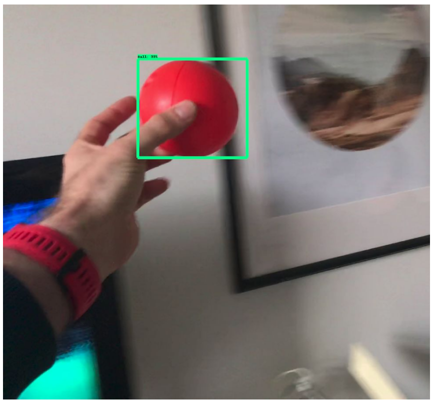
Distance Calculation:
The distance to camera and object size are inversely related to one another. They follow from

$$I_s = \frac{O_s \cdot f}{d}$$

Where I_s is the size of the object in the image, Os is the actual size of the object, f is the focal length and d is the distance between object and camera.
We gathered data of known distance and object image size and fitted to this relationship to determine Os * f. The value was found to be 0.0748.



Data: 6-7 video sequences converted into images and annotated using the labelIMG tool.




Total: 600 images - Network trained on 500 and tested on 100.

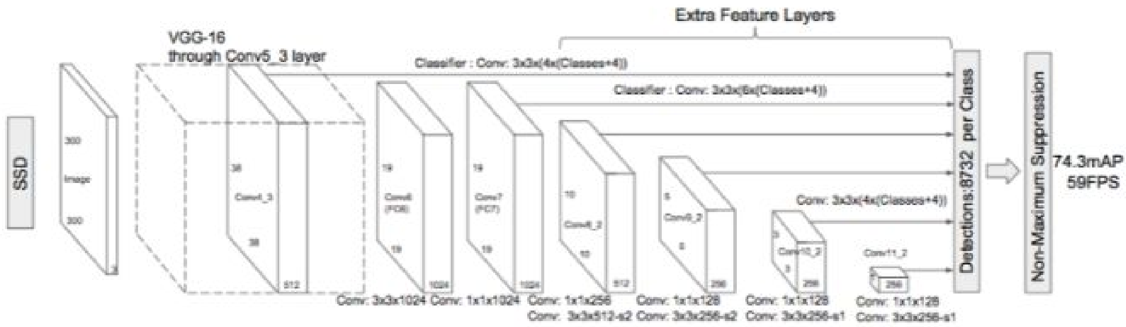
Model:
Ssd_mobilenet_v1_coco - Chosen for high FPS with good accuracy

COCO-trained models {#coco-models}

Model name	Speed (ms)	COCO mAP[^1]	Outputs
ssd_mobilenet_v1_coco	30	21	Boxes
ssd_mobilenet_v2_coco	31	22	Boxes
ssdlite_mobilenet_v2_coco	27	22	Boxes
ssd_inception_v2_coco	42	24	Boxes
faster_rcnn_inception_v2_coco	58	28	Boxes
faster_rcnn_resnet50_coco	89	30	Boxes
faster_rcnn_resnet50_lowproposals_coco	64		Boxes
rfcn_resnet101_coco	92	30	Boxes
faster_rcnn_resnet101_coco	106	32	Boxes
faster_rcnn_resnet101_lowproposals_coco	82		Boxes
faster_rcnn_inception_resnet_v2_atrous_coco	620	37	Boxes
faster_rcnn_inception_resnet_v2_atrous_lowproposals_coco	241		Boxes
faster_rcnn_nas	1833	43	Boxes
faster_rcnn_nas_lowproposals_coco	540		Boxes
mask_rcnn_inception_resnet_v2_atrous_coco	771	36	Masks
mask_rcnn_inception_v2_coco	79	25	Masks
mask_rcnn_resnet101_atrous_coco	470	33	Masks
mask_rcnn_resnet50_atrous_coco	343	29	Masks

Mobilenet is the underlying CNN for extracting features from images. “Mobile” because it can be run on mobile applications. Mobilenet, made by researchers at google, splits the 3x3 convolutions into 3x3 depthwise conv and a 1x1 pointwise conv for efficieny at the cost of some accuracy.

On top of mobilenet lies a SSD (single shot object detection).



Architecture of Single Shot MultiBox detector (input is 300x300x3)

Uses an underlying network for feature extraction (here mobilenet), and has a fixed number of output boxes. Each box has a set of class predictions and a location. This location is altered to try and match the actual underlying bounding boxes.

- **Confidence Loss**: this measures how confident the network is of the *objectness* of the computed bounding box. Categorical cross-entropy is used to compute this loss.
- **Location Loss:** this measures how *far away* the network's predicted bounding boxes are from the ground truth ones from the training set. L2-Norm is used here.

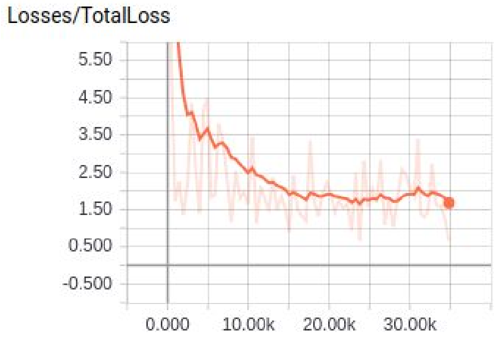
multibox_loss = confidence_loss + alpha * location_loss

Negative mining: Under training, most boxes will be wrong in the image which leads to an overwhelming majority of negative examples. Therefore we only keep enough of these for each image during training so we have a ratio of 3:1 negative:positive.

Non-max suppresion: We apply non-max suppresion by threshold the class loss and the IOU (intersection over union).

Final model is implemented in tensorflow object detection API and through quite a lot of code we made it do transfer learning to our own predefined dataset.

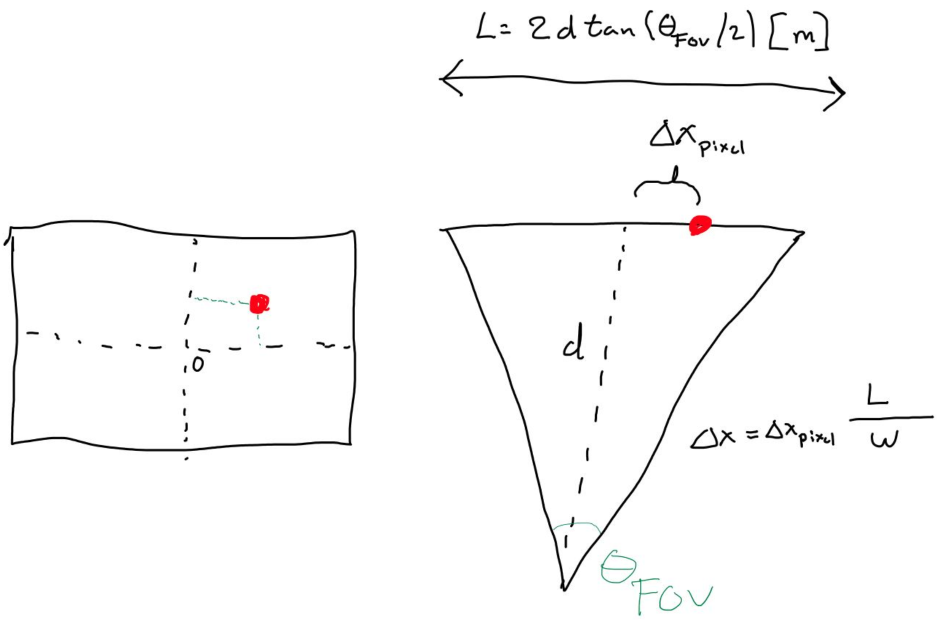
Training took around 6 hours on a big GPU server. We redid it quite a few times.



Tracking the ball:

We have a bounding box around the ball and we know the intrinsic camera parameters (iPhone 7).

Size of bounding box → Distance from ball to camera (approximation through measurements) **[m]**



Results