RESEARCH REPORT

Sanyam Sharma

Topics : Fish Detection and tracking system
experimentation with yolo

Submitted to : Prof. AK Bhateja
Dr. Aditi Bhateja

May 16, 2021

Contents

1 Abstract

We made a multi-class multi-object detection and tracking system for fishes using data gathered from Microsoft Kinect camera. The camera provides depth frames alongside RGB frames which can be used to track the coordinates of fishes in dimensions. We assigned unique id to each fish after labelling the data ourselves and using it as ground truth. Additionally, we experimented with attention layers and architecture of yolov5.

2 Introduction

We aim to distinguish the pattern of different species as well as different fishes of same species each having their own unique ID. Real time trajectory of school of fishes can be used for various purposes. We used RGBD channels to create a real time trajectory and further experimented with the model's architecture with attention layer.

3 Motivation

Yolov5 can be trained on any Object detection Dataset.We trained the model on dataset of fishes gathered by Microsoft Kinect Camera.It identifies the species of Fish and provide unique label to every species.This work will be helpful in multi-class multi-Object Detection as well as Object detection and tracking training on custom dataset. We further experiment with the yolov5 architecture and

add attention layer to improve the precision of the model.

4 Methodology

Methodology of the work has been explained in this section.

4.1 Data gathering and ground truth

The Microsoft Kinect sensor is a peripheral device (designed for Xbox and windows PCs) that functions much like a webcam. However, in addition to providing an RGB image, it also provides a depth map. Labelimg has been used for labelling the data and establishing ground truth.

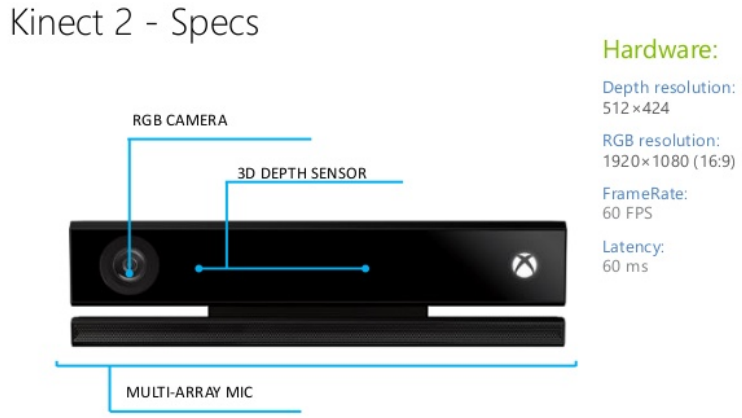


Figure 1: Microsoft Kinect

4.1.1 Data gathering

Meaning for every pixel seen by the sensor, the Kinect measures distance from the sensor. The Kinect V2, is a 3D sensor produced by Microsoft, it is composed by a RGB camera with resolution of 1920×1080 pixels, a depth camera with resolution of 534×451 pixels and an infrared emitter with a time stamp.
With the invention of the low-cost Microsoft Kinect sensor, high-resolution depth and visual (RGB) sensing has become available for widespread use. The complementary nature of the depth and visual information provided by the Kinect sensor opens up new opportunities to solve fundamental problems in computer vision.
The dataset used was gathered using KinectV2 in a control environment. Data was recorded and images on RGB frames, depth frames and infrared frames were extracted from the video for different values of concurrent timestamps.

4.1.2 Data processing

The dataset had to be processed first. The timestamps were needed to be aligned to the frames. This task have been discussed below :-
$-> $ Timestamps Timestamps of depth and color image were highly inconsistent with random gaps and intervals. Since KinectV2 was not originally designed for gathering consistent 3D data, timestamps were missing and were of different intervals for each and every frame of depth and RGB respectively. Upon using various techniques and some hit and trial method, appropriate intervals were

calculated and pattern was analyzed and appropriate images were chosen from the same. We chose the timestamp with minimum relative difference between RGB

and Depth frame carefully considering the extreme conditions. We analyzed timestamp of Depth frame and RGB frame, came upon a pattern, and transposed the RGB on Depth frame timestamp to avoid irregularity. Now we have appropriate frame of Depth and RGB with the closest time stamps. Our next step is to map each and every pixel of RBG frame to depth frame to form RGBD (Red Green Blue Depth) attributes.

4.1.3 Data labelling and ground truth

The data has been labelled with the help of Labelimg software. Each fish has been labelled by its own unique numeric id, so that the model can learn and assign correct classes to the fishes when detecting. Multiple ambiguities were handled while assigning the class and bounding box to the fishes. If only a certain fraction of the fish was visible, then we did not label it as the model can later on confuse the cracks and similar objects to be fish. Also, reflections of the fishes in the glass were not labelled as ground truth as the reflections are translucent in nature and model can learn this provided enough dataset.

4.1.4 Preprocessing

Auto Orient

Stripping of EXIF data (data to show the images as they are stored in disk). If the orientation of images in disk differs our inference can be ruined, thus we

stripped of this data and did auto orientation so that orientations of all the images fed in model are same.

Resize

To train a model on images we want to make sure all our images are of same aspect ratio and resolution, for our model training we resized all our images to 416X416, for best model training.

Auto-Adjust Contrast

Contrast Stretching- rescaled all images to include all intensities and removing any outliers that may silently ruin the model training Adaptive Histogram Equalization- for enhancing the local contrast by calculating histograms of pixel intensity values and spreading them out uniformly in different tile regions of the image.

4.2 Frame Calibration

While the Kinect's color and depth streams are represented as arrays of information, you simply cannot compare the x y coordinates between the sets of data equally: the bytes of the color frame represent color pixels from the color camera; the bits of the depth frame represent Cartesian distance from the depth camera.

To map RGB with depth, we have to calibrate our Kinect V2 using check board configuration to find intrinsic values of Kinect. Each and every Kinect has a slightly different set of intrinsic values. These intrinsic values along with the rotational matrix is used to map the same.

It is basically a standard stereo calibration technique;

the main difficulty comes from the depth image that cannot detect patterns on a flat surface. Thus, the pattern has to be created using depth difference.

Here I used a rectangular piece of carton cut around a chessboard printed on an A3 sheet of paper. The color camera intrinsic can be calibrated using standard chessboard recognition. Depth intrinsic value is found by extracting the corners of the chessboard on the depth image and storing them.

Mapping depth pixels with color pixels The first step is to undistort RGB and depth images using the estimated distortion coefficients. Then, using the depth camera intrinsics, each pixel (x_d, y_d) of the depth camera can be projected to metric 3D space using the following formula:

$$\begin{aligned} P3D.x &= (x_d - cx_d) * depth(x_d, y_d)/fx_d \\ P3D.y &= (y_d - cy_d) * depth(x_d, y_d)/fy_d \\ P3D.z &= depth(x_d, y_d) \end{aligned}$$

with fx_d, fy_d, cx_d, cy_d the intrinsics of the depth camera. We can then reproject each 3D point on the color image and get its color:

$$\begin{aligned} P3D' &= R.P3D + T \\ P2D_{rgb}.x &= (P3D'.x * fx_{rgb}/P3D'.z) + cx_{rgb} \\ P2D_{rgb}.y &= (P3D'.y * fy_{rgb}/P3D'.z) + cy_{rgb} \end{aligned}$$

with R and T the rotation and translation parameters estimated during the stereo calibration.

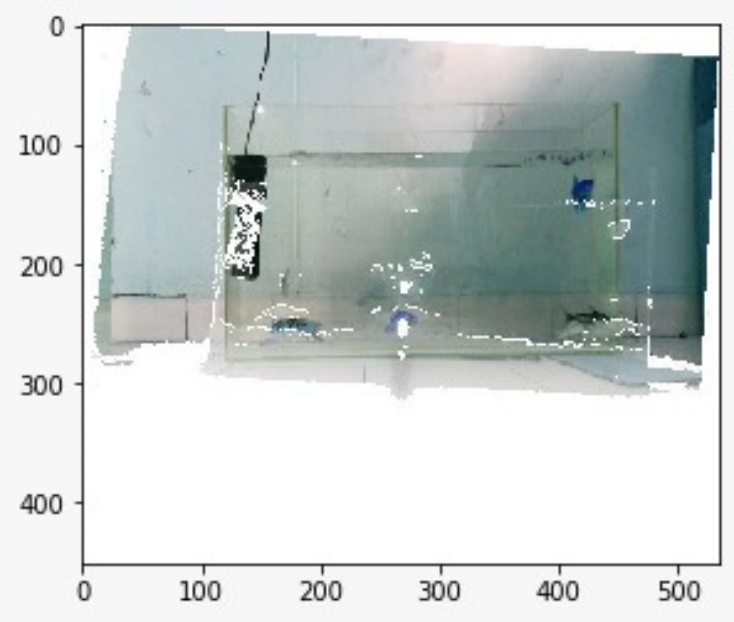
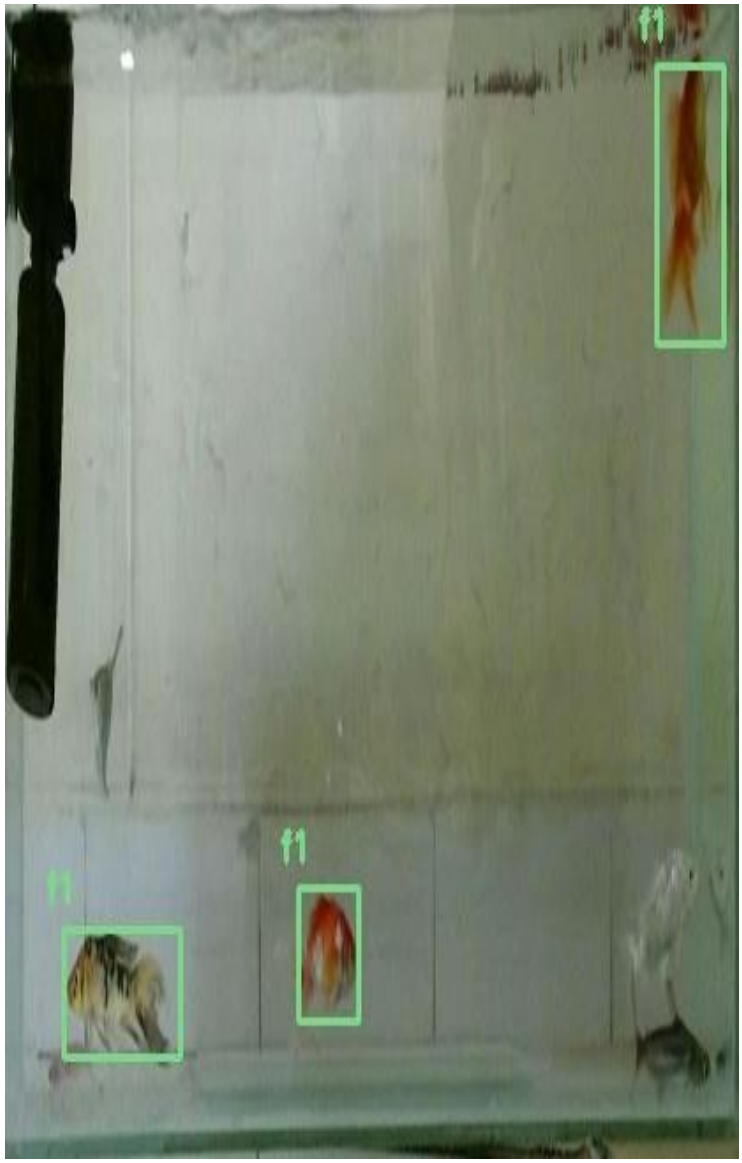


Figure 2: mapping

4.3 Model Preparation

We first used YOLOv3 for detecting fishes. We used pretrained weights for single class classification and used algebraic functions to assign unique IDs to the fishes thus upgrading it to a multi-class multi object classification system from a single-class multi-object system. We took the location the fish as a single point by considering only the center of the bounding boxes around the fishes and them compared it to the previous image and assigned IDs accordingly. But this method was inconsistent and results were not satisfactory. We then tried to accommodate the area of the bounding boxes into the function assigning the IDs to the fishes but since the movement of fishes is highly random in nature involving a lot of turning in close proximity

with each other. This method of upgrading a single-class multi-object classification system to a multi-class multi-object classification system failed. YOLOv3 single class classification upgrade results:-



single class classification experiment

4.3.1 Detection with existing yolo model

We annotated our dataset consisting of 943 images, where each image consisted of 7 fishes (Classes). After getting this tedious process done. We further augmented our dataset by resizing the images and flipping random images in between in order to provide more cases to our model so that it could learn better. We used the small version of YOLOv5 in order to get higher FPS from the system because the problem statement requires a real-time system. We have achieved inference of 0.01 seconds on our test data and therefore we can claim 100 FPS for our system. The results we achieved are as follows.

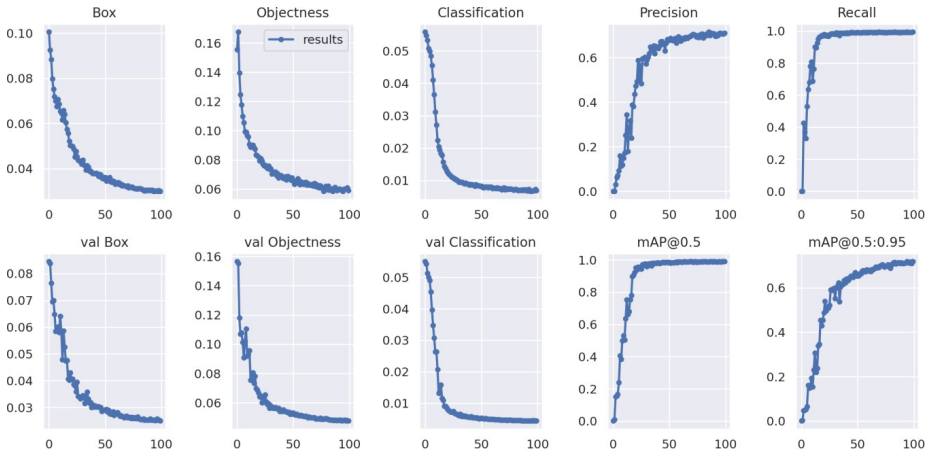


Figure 3 : single class classification experiment

4.3.2 Modified model architecture

We applied attention layer on extended version of yolov5 which has an extended head and uses multiple anchors. We applied attention layer at the end of the backbone

of yolov5s6 and got similar results. The experimentation results are documented on wandb platform.

4.3.3 Detection with modified model

The procedure of detection with modified yolov5 is same the procedure of detection in original yolov5. The results have been compared in tabular form in below section.

5 Results and Analysis

Modified yolo results :-

metric	value
precision	0.98677
recall	0.97718
mAP 0.5	0.0.98661
mAP 0.5:0.95	0.68882

Original Yolo results :-

metric	value
precision	0.9764
recall	0.98926
mAP 0.5	0.98593
mAP 0.5:0.95	0.69697

6 Conclusion

Hence, the initial problem statement given to us has been solved. A multi-class multi-object detection and track-

ing system for fishes using data from microsoft kinect has been made. Also, the additional task assigned to us for adding attention layer in yolov5 and experimenting with it has been completed too.

THANK YOU