Bottle Cap Detection and Classification Network (BCDC-Net)

Jaswanth Bandlamudi
Hochschule Bonn-Rhein sieg
Bonn, Germany
jaswanth.bandlamudi@smail.inf.h-brs.de

Abstract

*This work concentrates on building a pipeline for solving a task of detecting the crown caps in a tray from a video stream. The following tasks are solved to solve the task **i) Extracting a stable frame from the video stream, ii) Detecting the objects of interest in the stable frame, iii) Post-detection processing**. we used absolute difference from previous frame to the current frame and a falling edge in its value is used as a trigger for capturing stable frame. To perform crown caps detection we used a deep learning model and trained it on a dataset that's custom annotated. The post processing is performed to filter the detections made and provide the detections made as csv file. The evaluation is done by measuring the mean F-score using the ground truths from test dataset.*

1. Introduction

Object detection is a computer vision task of detecting instances of semantic object and assigning them with respective class names in digital videos and images. While, the task of object detection in Images is straight forward the object detection task on videos involves a stable frame extraction stage. By solving object detection task we can count objects in a scene and determine and track their precise locations, all while accurately labeling them.

This paper provides a general pipeline to detect crown caps, their condition (deformed or not deformed) or their pose (facing up or facing down) in the Region of Interest (RoI). In this project, the tray into which the caps are thrown into is considered as RoI. We are only interested in the coins that are present inside of the RoI i,e. inside the tray. This work is very helpful in efficient waste management by segregating the caps which can be re-used and cannot be re-used and also avoiding the accidental addition of environment harming materials into our ecosystem.

1.1. Contributions

In this work, we develop an efficient crown cap detection from the videos of the trays used to collect the caps. Our contributions are three-fold:

- Stable frame extraction based on the absolute differences in the consecutive frames.

- Use a Single-shot-detector with an InceptionNet backbone (SSD-InceptionV2) for crown cap, their pose and state detection.
- Filter the detections if they are present in the Tray or not.

2. Related Works

In this section we summarize about different methods for object detection methods from the images.

2.1. Object detection methods

Object detection has been one of the more sought after problems that is being solved by computer vision [2]. The main objective of object detection is to develop computational models and techniques that provide one of the most basic pieces of information needed by computer vision applications: What objects are and where they are?.

The evolution of object detection has mainly gone through two different phases. Traditional Object detection and Deep-learning based object detection.

2.1.1 Traditional object detectors

The initial day object detectors are mainly dependent on hand-crafted features. The traditional methods include

1. Viola Jones detector, it uses sliding window method for detection, the main idea is to go through all possible locations and scales in an image to search if any window embedes a human face.
2. Histogram of Gradients detector (HoG), In this method authors rescales the input image for multiple times also the size of a detection window is unchanged. This method has been a bench-mark and a foundation method for many computer vision application
3. Deformable Part-based Model (DPM), this method uses "divide and conquer" philosophy, where the network training is simply considered as learning a network to decompose the object parts. while, inference is assembling all the parts together to detect an object from the image.

2.1.2 Deep-learning based object detection

As the performance of hand-crafted features became saturated, object detection problem was solved using deep learning methods. In deep learning era, object detection can be grouped into two genres: "two-stage detection" and "one-stage detection", where the former frames the detection as a "coarse-to-fine" process while the later frames it as to "complete in one step".

1. In two-stage detectors, sparse region proposals are generated in the first stage and then are further processed in the second stage. RCNN [6] utilized low-level computer vision algorithms to generate proposals, then it adopted a Convolutional Neural Network to extract features for training a support vector module classifier and bounding box regressor. As an update to previous method Fast R-CNN [6] is proposed to extract features for each proposal on a shared feature map by pooling the spatial features. Faster R-CNN integrated the region proposal process into the deep convnet and makes the entire detector an end-to-end trainable model.

2. One-stage detectors perform classification and regression on dense anchor boxes without generating a sparse RoI set. YOLO [5] is an early exploration that directly detects objects on dense feature map. SSD [4] proposed to use multi-scale features for detecting variant scale objects. MobileNet-SSD V2 [3] proposed focal loss to address the extreme class imbalance problem in dense object detection. SSD-Inception [1] introduced anchor refinement module and the object detection module to imitate the cascade regression on dense anchor boxes.

3. Methodology

In this section we will summarize the method to extract a stable frame from from the video and detect objects of interest from the stable frame extracted. The methodology is as shown in Figure 1

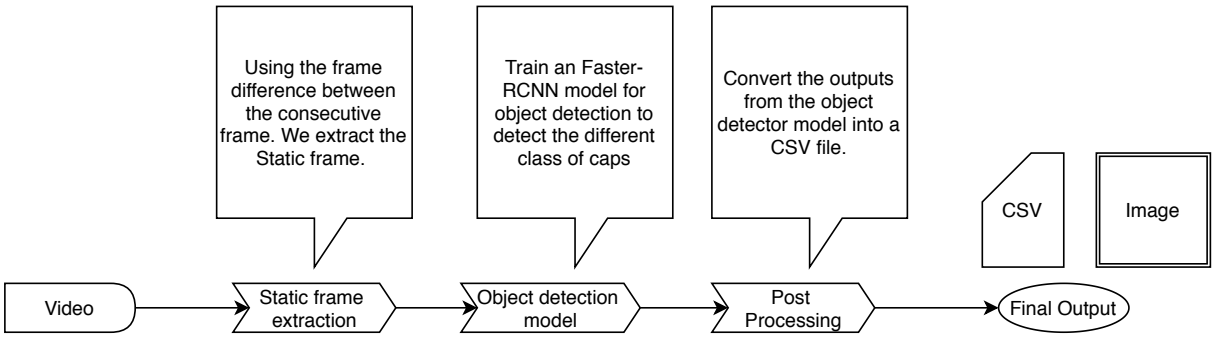

Figure 1. Pipeline for detecting the crown caps from a video stream.

3.1. Dataset

The dataset(D) consists of videos in which the task of throwing the caps along with detractors into a tray and picking them from the tray is performed. The videos should be a RGB video and have 1900 X 1080 resolution with a frame rate of 25 fps and a bit rate of 24 Bit. The static image extracted should also be a RGB and have 1900 X 1080 resolution with same bitrate. The dataset also has object labels of the caps and their state and pose labels.

$$Dataset(D) \ni (Videos(.mp4), labels(.json))$$

The labels of objects of interest O are

$$O \in \textit{(Cap faceup, Cap facedown, Cap deformed)}$$

along with the object of interests there are also distractors which include coins, wrappers and other objects. Inorder to make the filtering process easy we also included $Tray$ as object of interest. If the detected bounding box is in the bounding box of the tray then the object is considered to be in Region of Interest (RoI).

The statistics and the features of the dataset are shown in Table **??**

3.2. Stable-frame extraction

For the purpose of exact object detection from the provided videos it is necessary to find a frame in all of the objects remain stationary in the camera view for a defined time. In order to accomplish this we used the absolute difference between the consecutive frames as a signal to extract stable frame.

Total images	270
Training dataset	230
Validation dataset	23
Test dataset	17
Annotation type	Bounding box
Annotation format	VOC
Object categories	4
Image resolution	1024 * 1024

Table 1: Dataset exploratory analysis and statistics

The absolute difference of two images is defined as sum of the absolute difference at each pixel. For consecutive image frames the absolute difference $D(t)$ is defined as

$$D(t) = \Sigma_{i=0}^{N} |I_{t-1}(i) - I_t(i)| \tag{1}$$

N is the total number of pixels, i is the pixel count, t is the current frame number

We observe for the rising edge in the absolute frame difference value which indicates that the action of throwing the objects into the frame had started and then we check for a falling edge for three consecutive frames which indicates the action of throwing the caps is completed. Hence, using a frame after completing the object would result in a frame with all stationary objects. The algorithm 1 shows various steps involved in stable frame extraction. The depiction of our approach can be seen in Figure 2.

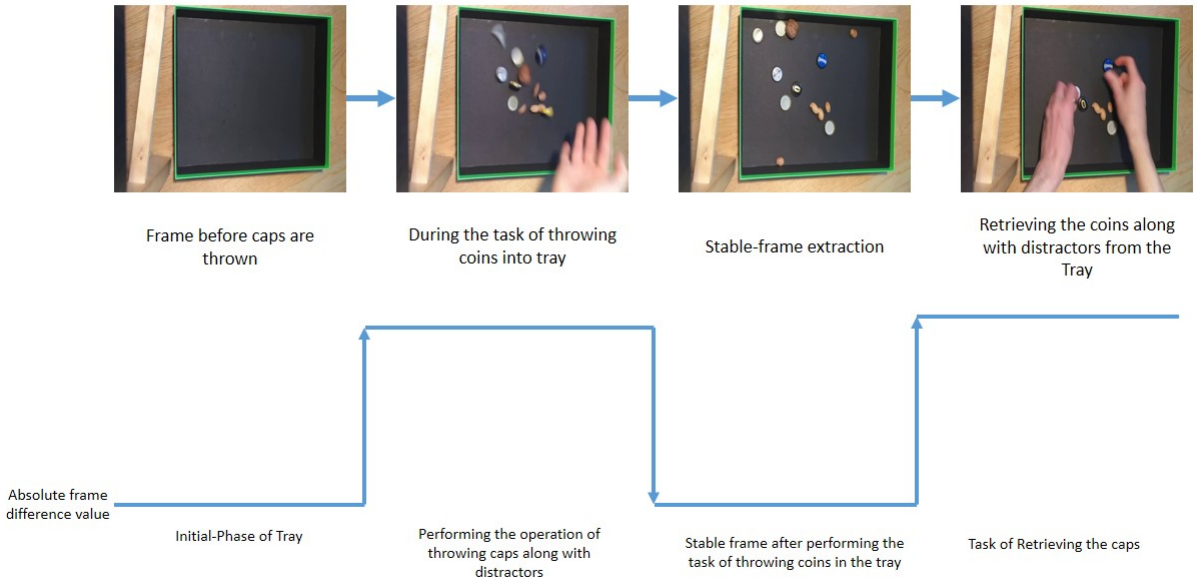

Figure 2. Stable frame extraction pipeline

The Algorithm for stable frame extraction from a video stream is mentioned here

Algorithm 1: Extract stable frame

```
Result: A stable frame with no moving objects
initialization;
while Video stream is ON do
    abs diff = modulus(current frame - prev frame);
    if raising edge on abs diff then
        Cap throwing started;
        if falling edge for three consecutive frame then
            stable frame detected
        else
            continue
        end
    else
        continue
    end
    if stable frame not detected then
        stable frame = middle frame from video;
    else
    end
end
```

3.3. Crown cap detection and classification

In order to detect objects in the extracted stable frame in the previous step, a deep learning based method is being used. To solve this task we chose to make use of Single Shot Detector(SSD) [4] with Inception V2 [7] network as feature extractor.

The reason for chosing SSD over the multi-stage object detectors like Faster-RCNN [6] or Mask-RCNN is that the SSD provides better detection speed with best possible accuracy. As the application is supposed to be deployed on devices while consuming low power. Hence, the design choice of using SSD is made. The architecture of SSD is shown in Figure 8.

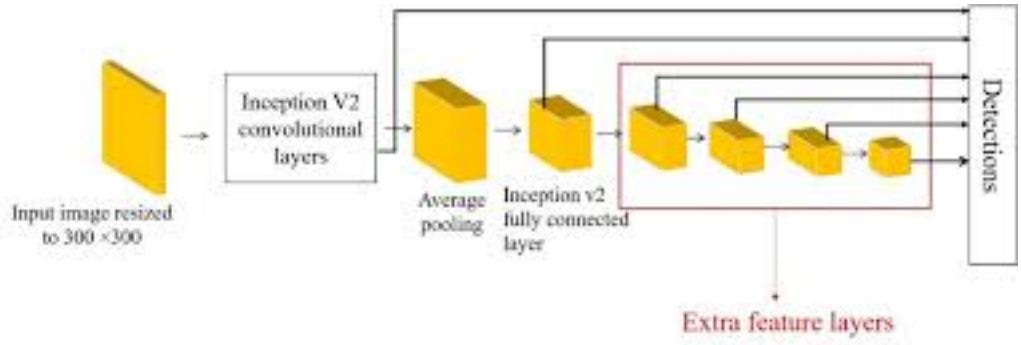

Figure 3. Single Shot Detector with InceptionNet feature extractor.

The detection pipeline is implemented by taking the advantage of pre-trained detection models from the Tensorflow Object Detection API and by exploiting the concept of Transfer learning.

4. Experimental Results

In this sections we explain the experimental setup and the performance of the model in detecting the objects.

4.1. Experimental setup

The experiments are performed using the High perfromance computing cluster provided by the Platform for scientific computing at Hochschule Bonn-Rhein Sieg. The setup is as follows

Tools used	Numpy, Tensorflow, OpenCV
Processor	NVIDIA TITAN RTX 24GB
Training Paradigm	Transfer learning

Table 2: Experimental Setup

4.2. Stable frame extraction

The stable frame extraction is implemented using opencv. To calculate the absolute difference, we converted the images from RGB color model to a Gray color model which results in the images being in a same coordinate frame. Absolute difference of images is calculated using $absdiff$ function from opencv.

As mentioned in Algorithm 1 this $absdiff$ value is observed in order to extract the stable frame.

4.3. Cap detection and Classification

For the purpose of cap detection and classification, we used a SSD with Inception-V2 backbone. These deep learning models needs a lot of data to get trained but the available data is very limited with only 230 images available in the training dataset. In order to adddress the lack of data we chose to employ data augmentation techniques. The augmentation used techniques are

1. Random Horizontal Flip
2. Random Adjust Brightness
3. Random Adjust Contrast
4. Random Jitter Boxes
5. Random Black Patches
6. Random Pad Image

The model is trained using Tensorflow object detection API and we used an early stoppage criterion by monitoring the loss on the validation dataset.

The model is trained for 76 epochs and is early stopped due to the validation dataset loss not reducing any more. We used the same test dataset to evaluate the model on the 40 images present. The results are as follows

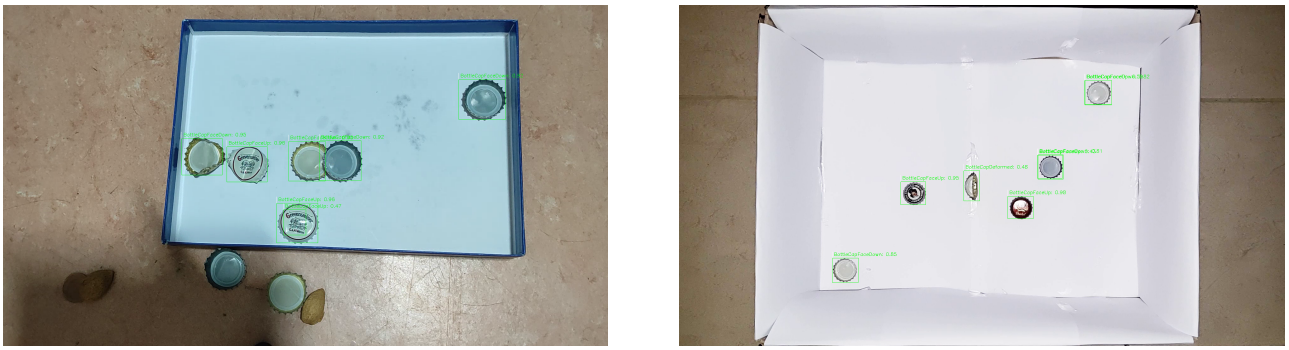

Figure 4. Detections done using SSD object detection model

Detection class	Cap Faceup	Cap Facedown	Cap Face deformed	Tray
Cap Faceup	71	2	0	0
Cap Facedown	6	48	1	0
Cap Face deformed	3	6	18	0
Tray	0	0	0	14

Table 3: Results from detection model

5. Conclusion and observations

The algorithm worked well in case of detecting Cap with Daceup or Face down. But the model struggled in detecting the deformed caps. We think the reason for the lack of performance in case of deformed caps is majorly the less amount of deformed cap samples in the training dataset and the amount of common features between the normal caps and the deformed caps.

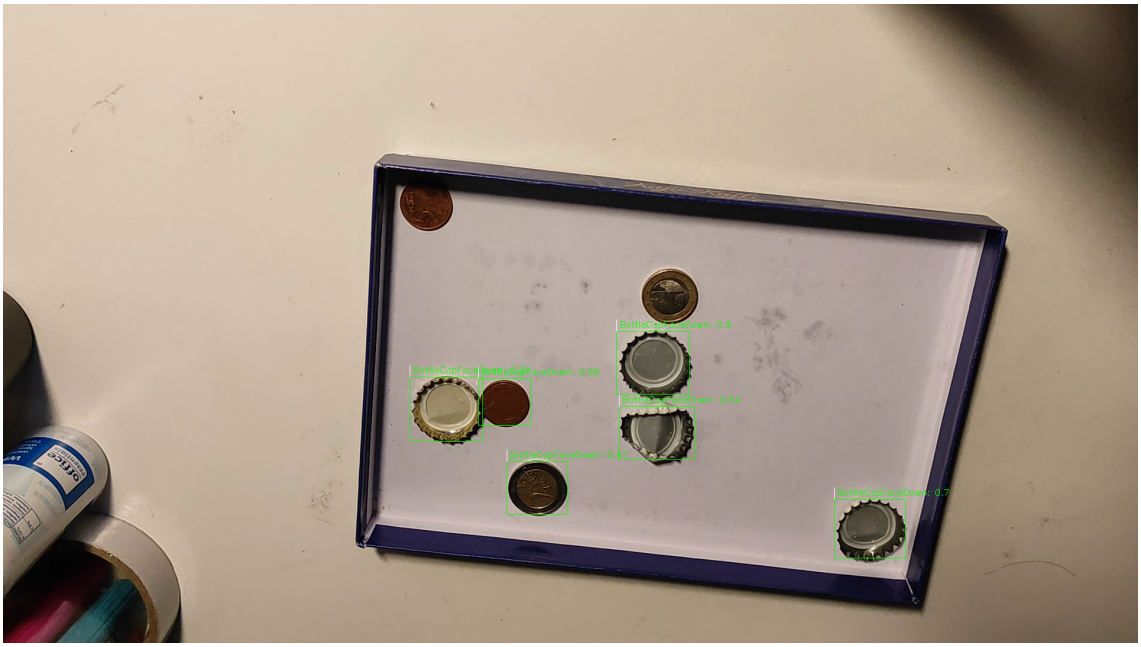

Figure 5. Model classifying the deformed bottle cap as Normal cap

We also observed that the model struggled in detecting the tray. Especially in cases where the general tray doesn't exist but a tray like setting was made. Inorder to make the model work in these cases we implemented a largest rectangle shape extraction from the stable frame which possibly is a nearest possible tray detection and decided to use it to filter the detections.

Figure 6. Tray not detected by the SSD detector but detected using largest rectangle shape finding method

The largest rectangle like object method in a condition where the tray like object doesn't exists. We decided to not consider it as a valid dataset and ignored it from testing different conditions

Figure 7. Ignored dataset becasue of absence of no Tray like object present

The performance of model is satisfactory in poor light condition which we believe is attributed to the augmented techniques used. Though it missed detecting some caps we observed that missed caps were hard to detect even with a human eye.

Figure 8. Performance in very poor lighting conditions

References

[1] S. R. Bose and V. S. Kumar. Efficient inception v2 based deep convolutional neural network for real-time hand action recognition. *IET Image Processing*, 14(4):688–696, 2020.

[2] Dario Cazzato, Claudio Cimarelli, Jose Luis Sanchez-Lopez, Holger Voos, and Marco Leo. A survey of computer vision methods for 2d object detection from unmanned aerial vehicles. *Journal of Imaging*, 6(8), 2020.

[3] Y. C. Chiu, C. Y. Tsai, M. D. Ruan, G. Y. Shen, and T. T. Lee. Mobilenet-ssdv2: An improved object detection model for embedded systems. In *2020 International Conference on System Science and Engineering (ICSSE)*, pages 1–5, 2020.

[4] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 21–37, Cham, 2016. Springer International Publishing.

[5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, June 2016.

[6] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2017.

[7] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015.