Edible nut detection in videos using deep learning

Anirudh Narasimamurthy Jayasimha
Department of Computer Science
Hochschule Bonn-Rhein-Sieg
Bonn, Germany
anijaya9@gmail.com

***Abstract*—This paper provides a pipeline for detecting the total number of edible nuts present in a video that lies within a homogeneous region of interest. The steps in performing the task are logical divided into three sub-tasks: 1) Extracting the most stable frame in the video, 2) Finding the 3 classes of edible nuts in the image frame, and lastly 3) Post-processing the result. Extracting the most stable frame involves selecting the video frame where all the objects are stable. Our approach for solving this task involves taking the difference between consecutive frames of the video and taking the first frame where the total number of unique pixels lie below a given threshold. For the edible nut detection task a deep learning model was trained on a custom annotated dataset. Finally the results extracted from the deep learning model is extracted and processed and written to a csv file along with a result image. The mean F-scores of the results are used to evaluate the obtained results with the ground truth.**

***Index Terms*—Object detection, Stable frame detection**

I. Introduction

The task of finding objects in an image and localizing it is a fundamental problem in computer vision. The main goal of object detection is to locate the instances of an object that are present in the camera frame and draw a bounding-box. In the case of a video there are other preprocessing steps that are required to extract the frame and run detection on them.

This paper gives a pipeline for extracting various types of edible nuts that are present in a given region of interest of a frame. The tray that is present in the camera frame is considered to be the ROI in our case. Counting objects in the computer frame has many important application in both industries and academia. Industries make use of this nut detection application for classification of nuts and segregating them. The ideas given in this paper can also be extended or modified for many such similar tasks in industrial environments like object quality estimation or segregation.

II. Previous Work

The task of object detection has undergone a tectonic shift in the recent years due to the practical advent of the learning based methods. The study of object detection can be broadly be divided into two time periods : *"traditional object detection period (before 2014)"* and *"deep learning based detection period (after 2014)"* [1].

Classical or traditional computer vision is referred to as methods where hand-crafted features or patterns are involved. These patterns are based on the geometric and color properties of the objects and other insights gained by the humans about the environment. Machine learning and deep learning methods on the other hand are data driven methods where the features and patterns are automatically extracted from the data. To extract good features, data driven methods must be trained on a large and diverse dataset which should give a good estimate of the whole population.

The representation of the various features are one of the most important task for object detection. Some of the common image features that can be observed are the edges, corners, region of high contrasts in the image, ridges etc. The major issue with such simple features are that they are highly dependent on the object pose and the lighting conditions and cannot be generalizable. Methods like Scale Invariant Feature Transform (SIFT) [2] and Speeded-Up Robust Features SURF [3] build on the simpler features and use many techniques like Gaussian pyramids to make them scale and rotation invariant. These methods give good repeatably and robust features but cannot be used for cases where there are less overlapping features between the test and the training images. This reduces the generalizablity to object detection tasks.

Unlike the classical methods of feature extraction, deep learning methods are able to extract the various complex features without any manual intervention. The deep learning feature extractors can learn shape, size, color, texture and other inherent object features that are very difficult to manually encode. Given enough data, the deep learning model like SSD [4], YOLO [5] and F-RCNN [6] can give state of the art object detection and precise bounding box and are highly generalizable to different lighting conditions, object pose and perspectives.

III. Methodology

The task of finding nuts in a video frame can logically be divided into three separate sub-tasks.

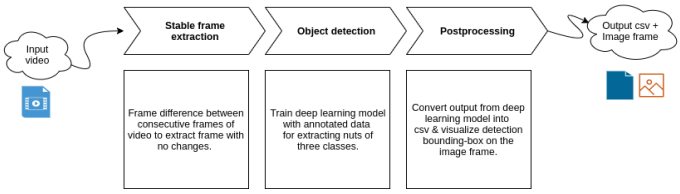

Fig. 1. The general pipeline to find the most stable frame of the video, find the nuts present in the ROI and writing the results to a CSV file.

The first of the three subtask is *1) Stable frame detection*: This step deals with finding the frame in the video when all the objects lying the ROI are stationary. *2) Object detection* : This method deals with finding the three classes of edible nuts that are found in the frame and finding the coordinates of the bounding boxes. The last step *3) Postprocessing* Involves using the results obtained from the object detection stage and removing all the results that don't lie inside the region of interest. Each of the subtask of the pipeline will be dependent on the results obtained from the previous steps to successfully process. So it is important to make sure that each of the step gets successfully completed to complete the whole task correctly.

A. Dataset

The dataset consists of two groups($\mathcal{G}$) of objects ie. the edible nuts($\mathcal{N}$) and the distractors($\mathcal{D}$).

$$\mathcal{G} \ni \{edible\ nuts(\mathcal{N}), distractors(\mathcal{D})\} \tag{1}$$

The edible nuts has three fixed number of classes: hazelnut, peanut and walnut. These objects are the main focus for the object detection task and must be detected by us ie. True positive classes. The distractors may consists of any small object like pens, bottle caps, coins etc. The distractors should not be detected by the object detection model and the main task of these objects is to confuse the model and increase the false positive results.

$$\mathcal{N} \ni \{peanut, hazelnut, walnut\} \tag{2}$$

$$\mathcal{D} \ni \{pens, caps, dice, ...\} \tag{3}$$

The images with the nuts and distractors were annotated using Labelme [1] to get polygon annotations of the objects. These polygon annotations were then converted into VOC bounding-box annotations to make them compatible with tensorflow data loader.

Total images	380
Training dataset	300
Test dataset	80
Annotation type	Bounding box
Annotation format	VOC
Object categories	4
Image resolution	1936 * 1216

TABLE I
DATASET STATISTICS.

The annotated images mentioned above were extracted from the corresponding 380 videos which were later used to test the entire pipeline. Few of the assumptions made during the collection of the dataset is that the the complete tray will always be visible within the camera frame. The lighting is sufficient enough to observe the objects in the frame and the objects not visible directly to human eyes are not considered as valid ground truths. We also assume the camera properties are homogeneous ie. no change in resolution or other properties throughout the data collection.

[1]https://github.com/wkentaro/labelme

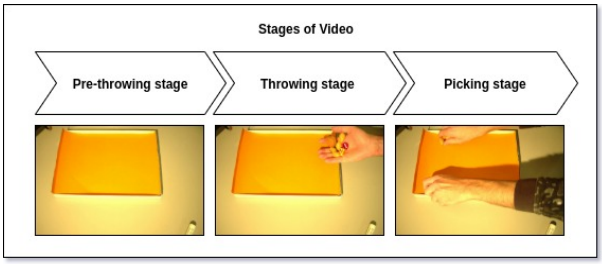

Fig. 2. Stages of the video based on the action being performed by the user.

The videos in the dataset can be roughly divided into three separate stages based on the action being performed in the frame. The first stage is *1) Pre-throwing stage*: This stage includes the first few frames in the video where the ROI(tray) without any objects and the table is visible. This stage may lead to false positive results in the stable frame extraction stage as there is no movement in these frames. The second stage *2) Throwing stage* : This stage include the user throwing the nuts inside the ROI and waiting till all the nuts and distractors are stationary. The stable frame lies within this stage of the video. The last stage in the video *3) Picking stage* : This stage involves picking up all the nuts and the distractors from the tray and the stationary frames till the end of the video.

B. Stable frame extraction

Extraction of stable frame involves selecting the frame in the video after the objects are completely stationary. The heuristic to consider a stable frame is considered as : Nuts and the distractors must not be moving for minimum of 2 consecutive frames.

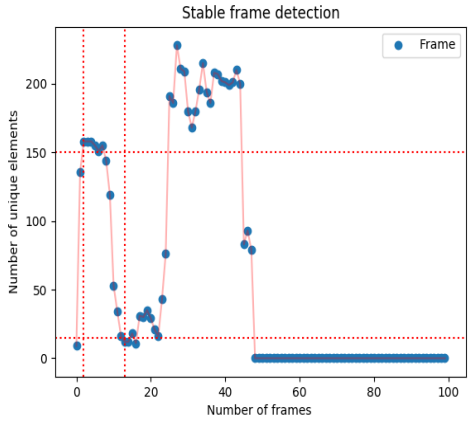

Fig. 3. Graph showing the total number of unique elements present in the difference frame across the video.

The algorithm used to find the stable frame is an extension of the consecutive frame subtraction. For each extracted frame ($\mathcal{F}_t$) of the video the difference between that and the previous frame ($\mathcal{F}_{t-1}$) is found. Using the obtained subtracted frame ($\mathcal{S}_t$), the total number of unique elements in are calculated.

Framework	Tensorflow 1.14 , OpenCV DNN
Model	faster_rcnn_inception_v2_coco
Base architecture	Faster-RCNN
Input size	640 * 480
Feature extractor	faster_rcnn_inception_v2
Batch size	16
Pretrained dataset	COCO
Augmentation	random_horizontal_flip , random_vertical_flip , random_adjust_brightness
Classes	4

TABLE II
IMPLEMENTATION OF CNN MODEL FOR TRAINING AND INFERENCE.

$$\mathcal{S}_t = \mathcal{F}_t - \mathcal{F}_{t-1} \tag{4}$$

The above algorithm is performed to the list of all consecutive frames of the video and a graph of the number of unique elements vs the frame is generated. Performing this on all the 300 training videos it was observed that we can use the results to find the starting and the ending of each stage of the video. Flags were set in the code to start to extract the stable frame after the first stage is complete and the total number of unique elements between the two consecutive frames is less than 10.

C. Object detection

For the task of extracting the nuts in the extracted stable frame, a learning based method has be considered. The large annotated dataset that is collected for the task can be leveraged by the Convolution Neural Network model to extract highly generalizable features and work under diverse object poses and environment states.

Compared to single-stage detectors like YOLO [5] and SSD [4] which are designed to obtain fast inference times by compromising on the high detection accuracy, FR-CNN is a two stage detector that is designed for a very high accuracy.

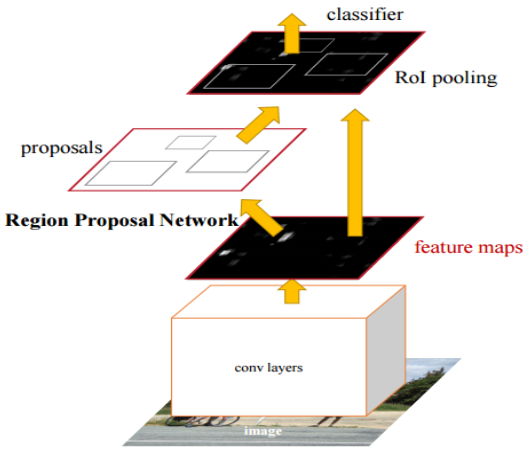

Fig. 4. Faster-RCNN uses two stages of feature extraction and classification to improve the object detection accuracy and give a more precise bounding box. [6]

D. Postprocessing

This step involves removing the detected bounding boxes that lie outside the region of interest according to the problem statement. When capturing the video of the tray with objects ($\mathcal{G}$), there are many nuts ($\mathcal{N}$) that lie within the image frame but not necessarily inside the ROI.

For extracting only the true positive detection that lie inside our ROI, we use the bounding-box of the detected tray object to remove all the object detection bounding-box centres that lie outside it. This is done by iterating through all the detected bounding box except the tray, and find their centre and perform a simple point in polygon test.

IV. RESULTS

Using the annotated ground truth and the results obtained from the pipeline, the true positives (TP), false positives (FP), true negatives (TN), false negatives (FN) were calculated and visualized in the confusion matrix given below. The object detection was run on 80 images of the training dataset.

Actual \Predicted	Tray	Walnut	Peanut	Hazelnut	Other
Tray	**80**				
Walnut		**174**			
Peanut			**157**		
Hazelnut				**177**	
Other	**1**				**0**

TABLE III
CLASS RESULTS CONFUSION MATRIX SHOWING TRUE POSITIVE, FALSE POSITIVE, FALSE NEGATIVE AND FALSE POSITIVE. THE 'OTHER' CLASS INDICATES ALL THE DISTRACTORS AND OTHER ITEMS IN THE IMAGE FRAME THAT LEADS TO FALSE POSITIVES.

It was observed that the model was able to detected a total of 177 Hazelnut out of total 192 ground truth values to give a detection accuracy of **92.18%**. The failures were attributed mainly to hazelnuts which were being occluded by the tray or other objects and were only partially visible, and due to bad lighting conditions. In the case of Walnuts the detection accuracy was **100%** ie. all the 174 ground truth walnuts were detected correctly in our dataset. Out of the 159 peanuts, 157 were detected in our test dataset giving a detection accuracy of **98.7%** class accuracy. The tray in the frame were also detected **100%** of the time but there were a few cases of false positives as the table was detected as a tray due to similar texture and lighting conditions between the two.

Class	Total images	Detected images	Percentage accuracy
Walnuts	174	174	100%
Peanuts	159	157	98.7%
Hazelnuts	192	177	92.18
Tray	80	80	100%

TABLE IV
INDIVIDUAL CLASS ACCURACY OF THE DETECTED CLASSES.

It was observed that when performing object detection, most of the errors were false negatives ie. the nuts or the tray were not detected correctly in the image. In a few images due to the similar texture of the table and the tray, the tray was detected more than once. There were no cases of the nuts

to be classified to the wrong class once it was detected. Such high detection were attributed to the very similar conditions in which the test dataset and the training dataset were taken in.

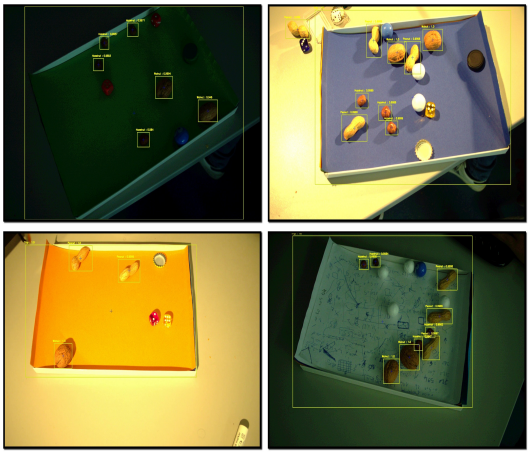

Fig. 5. Results of the extracted stable frame and the object detection performed to find the ROI and the various true positive results.

V. Expected errors

There are three specific errors that were observed during testing. These errors were still not corrected at the time of writing this report. The main error in each of the sub-task is given below.

A. Error stable frame detection

Since manual threshold values are used for extracting the most stable frame from the video, there are many videos in our dataset that cannot work well with it. This leads to errors in extracting the most stable frame from the video.

B. Errors in the object detection

This report is considering only two classification errors *false positive* and *false negatives*. False positive refers to an object detected as a positive sample even though it is not. False negative refers to the object detected as a negative sample even though it is not.

It was observed from the detection results that the hazelnuts were a major cause of false negatives. In case of the tray the similar homogeneous texture and the color of the table caused the tray to be detected twice in the image frame. This problems can be corrected by training models with a more diverse dataset and better augmentation. The model can be trained to less epochs to reduce overfitting.

C. Errors in postprocessing the output

The major problems faced with postprocessing involves finding the colored tray which is the region of interest for our task. Since the bounding box we get is always going to be a largest bounding box rectangle that covers the tray regardless of the orientation. In come cases the nuts may not not lie inside the region of interest but inside the bounding-box. This type of error leads to increase in False positives.

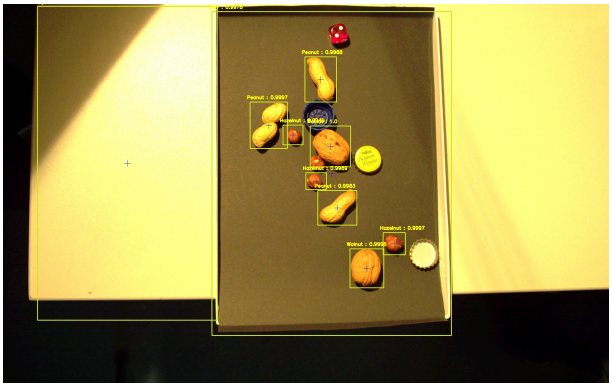

Fig. 6. Two trays are detected by the CNN detector even though there exists only one tray in the frame. The extra frame is a false positive.

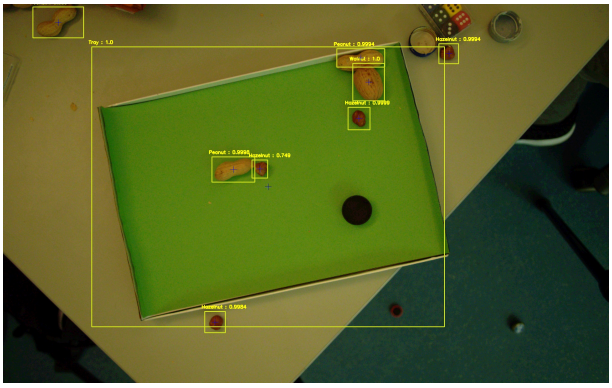

Fig. 7. The hazelnut is considered as a true positive even though it doesn't lie inside our region of interest.

The problem shown in the figure can be corrected by using skewed bounding box on the tray which will give a more precise region of interest around the tray.

VI. Conclusion

The task of detecting the True Positive objects from a video is quite complex considering that the failure of any one of the subtask can lead to the failure of the entire task. Extracting the stable frame is the main sub-task that leads to the failure under diverse lighting and camera positions. This is due to the manual threshold values that is used in the algorithm which are non-generalizable to all videos.

As for the task of object detection, both classical and learning based methods were tested for the task. From the various experiments conducted it can be observed that deep learning methods are able to obtain state of the art detection accuracy and generalization if there are enough annotated data. In the case of our task the object detector was able to give a mAP of over *80%*. Although deep learning methods are able to obtain excellent results, these methods are black-box and it is difficult to understand the main results of the false positive results or unexpected outputs.

The task of extracting only the objects that lie inside the region of interest are solved using the bounding box obtained by the deep learning model. Although this method fails when the objects lie very close to the tray but not inside the it. There are many methods, like the ones discussed in the previous sections that can be used to improve the algorithms and that can be considered as a future enhancement.

ACKNOWLEDGMENT

I would like to thank Prof. Dr. Rainer Herpers and Christoph Pomrehn for providing an opportunity to use the Lab for collecting the dataset. I would thank our university for providing the GPU cluster and the computational resources for training the models.

REFERENCES

[1] Z. Zou, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: A survey," *ArXiv*, vol. abs/1905.05055, 2019.
[2] D. G. Lowe, "Object recognition from local scale-invariant features," *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2, pp. 1150–1157 vol.2, 1999.
[3] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, "Speeded-up robust features (surf)," *Computer Vision and Image Understanding*, vol. 110, pp. 346–359, 2008.
[4] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *ECCV*, 2016.
[5] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788, 2015.
[6] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, pp. 1137–1149, 2015.
[7] "Comparison of hash functions in opencv."

VII. APPENDIX

There were a few unsuccessful methods that were tried. These methods showed unsuccessful results and were not generalizable to the large lighting and contrast variations in the images present in the dataset.

A. Stable frame extractor - Unsuccessful methods

The main algorithm required for this sub-task is that it must be able to differentiate between two slightly different camera frames. One of the common methods that is used for obtaining image similarity is the use of multimedia hashing functions. The main aim of these multimedia hashing function is to map an image to to a hash value such that similar hash values denote similarity between images and if the distance between the hash values are large it means the images are very difference.

One such hash function that is used for our task is called as perpetual hashing function or pHash. pHash function for images convert the input image into the frequency domain using Discrete Cosine Transform. This makes it easier to remove unwanted data from the image and compress it. Performing a few down-sampling and processing gives a 64 bit hash which can be used for comparison. Simple distance function like hamming distance can be used for comparing two different pHash obtained.

For this task we will be using pHash [7] due to its robustness against noise, brightness and contrast differences. For finding the most stable frame we start finding the hash values of all the frames starting from the 2nd Stage of the video ie. start of the throwing stage. This is to prevent the stationary frames in the pre-throwing stage 2 from being extracted, which will return wrong results.

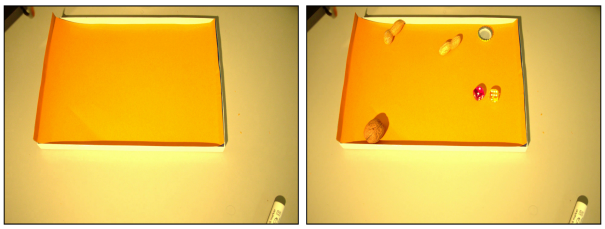

Fig. 8. Difference between the hashes is a positive number as the images are very different.

When the images are very different like shown in the Figure. 8, the hashes of the images will be different so the difference between them are a positive number which can be used to differentiate between them.

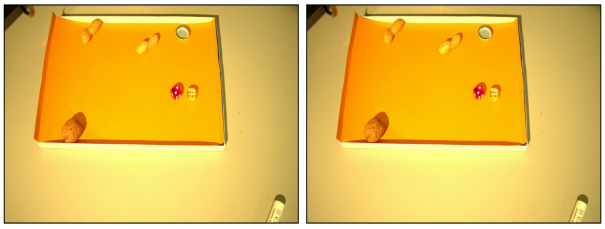

Fig. 9. Difference between the hashes is 0 although the images have a slight difference.

But the major drawback of using hash functions is that when the images are very similar like that observed in Figure.9, the hash functions are the same. This leads to selection of a stable frame based on the very small or null hash function distance.

B. Object detection - Unsuccessful methods

The classical methods that were tried for extracting the nuts in the frame were observed to fail due to the large variance in the lighting conditions of the dataset given.

To extract the nuts, the color based thresholding methods were used as an initial choice. This was because the nuts are all similar in color and in most cases can be The major problem with these methods were observed to be the difficulty in setting a correct color threshold range over a wide range of environment lighting conditions.

Even in the case of performing a lot of preprocessing steps to improve the thresholding, the contours that were detected needed to be classified based on the shape, color or other physical metrices or features. This would lead to lot of extra problems like finding the best ratio between the various areas of the contour or additional color based classification for various nuts. The distractors in the video frame would also lead

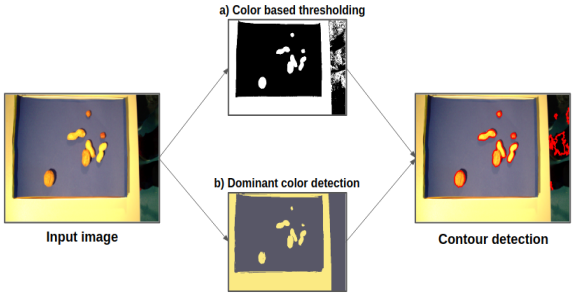

Fig. 10. Color based contour detection for finding objects in the required ROI.

to more classification problems due to the extreme variance in the size, shape and color.

C. *Postprocessing - Unsuccessful methods*

The extraction of the largest homogeneous region of interest was initially formulated as a color based RoI extraction method. These methods failed mainly due to the diverse lighting conditions and difficulty in extracting colors of different types using thresholding.

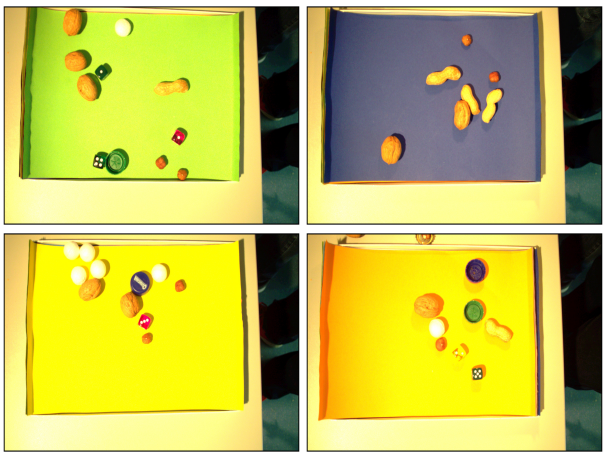

Fig. 11. Dataset showing tray with different objects, colored backgrounds and under different lighting conditions.