Computer Vision and Image Processing for Commercial Use Applications

Maggie Barclay, Theo Guidroz
Embedded Computing and Signal Processing (ECASP) Research Laboratory
Department of Electrical and Computer Engineering
Illinois Institute of Technology, Chicago, IL, U.S.A.

Abstract—Utilizing computer vision for creating new smart technologies can significantly improve the practical applications of a system. This project explores several concepts in image processing, including object detection and object classification. A NVIDIA Jetson Nano Device is utilized to demonstrate the basics of Computer Vision. Python, TensorFlow, Keras, and OpenCV are used to work with trained networks. The networks for recognizing specific objects are trained to utilize YOLO, a network with Deep Learning algorithms. This project aims to construct an experience for a consumer, optimizing these concepts into a smart technology 'product'.

I. INTRODUCTION

I.I Necessity in Context

Computer Vision is increasingly being utilized for surveillance purposes, for creating driverless vehicles, and into individuals' everyday lives. In order to contribute to the movement of bringing smart technologies to the consumer, a focus is placed on providing automation or simplification on portions of daily life. The effort discussed within this report has this emphasis as it's purpose.

I.II Project Overview

In order to incorporate computer vision into everyday life, groceries and food supply regulation will be the topic of focus. Within a typical household refrigerator, smart technology can be used to monitor and report on food items currently in stock. This system can allow remote access to this information for consumer benefit. In order to create this system, there are several distinct elements allowing for the full intended functionality.

<u>Project Statement</u>: *This project makes considerable progress towards a refrigerator monitoring system with the capabilities to monitor and report on contents in real-time, utilizing computer vision technology.*

Within *Figure 1* below, the flowchart of the project's components is displayed. Within the chart, a single image is first captured from the refrigerator and processed by the Jetson Nano. The image first runs through a QR code scanner since QR codes are the most accurate way of detecting an object. If any object is detected, the name of the product is written in a text file. The Jetson Nano then runs the second step, which is an image classification algorithm implemented using Yolo V3. The detected objects are logged on the text file. A second image classification is run to detect any bottles in the refrigerator. If a bottle is found, a cropped picture of the bottle is saved. A text detection algorithm is then implemented on the copped image to read the label and saving the content of the bottle on the text file.

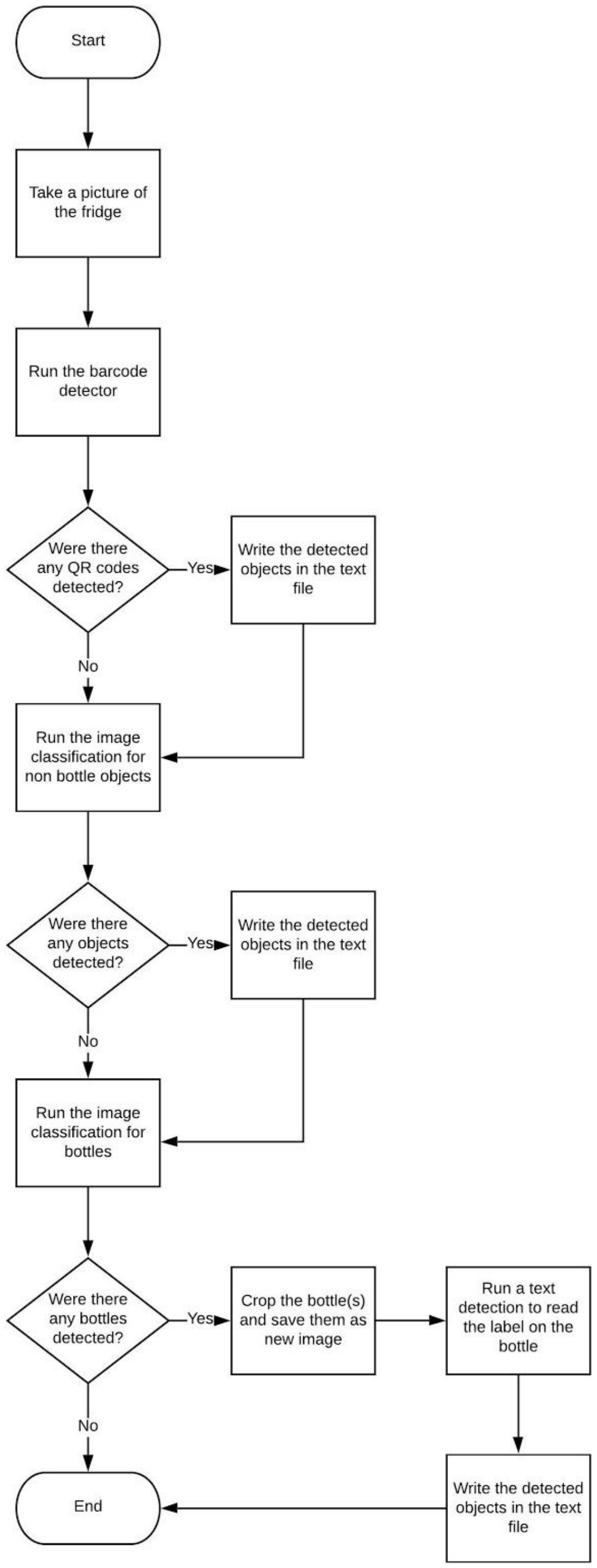

Start
Take a picture of the fridge
Run the barcode detector
Were there any QR codes detected?
Yes
Write the detected objects in the text file
No
Run the image classification for non bottle objects
Were there any objects detected?
Yes
Write the detected objects in the text file
No
Run the image classification for bottles
Were there any bottles detected?
Yes
Crop the bottle(s) and save them as new image
Run a text detection to read the label on the bottle
Write the detected objects in the text file
No
End

Figure 1: Current system flowchart

Figure 1 conveys the current state of the project. There are numerous elements left to be implemented to reach full intended functionality. Implemented elements will be described in depth, and unfinished elements will also be discussed as they are critical to understanding the intention of the system.

II. TECHNICAL COMPONENTS

II.I Hardware and Software

A. *Jetson Nano*

NVIDIA Jetson Nano is an embedded system-on-module and developer kit from the NVIDIA Jetson family, including an integrated 128-core Maxwell GPU, quad-core ARM A57 64-bit CPU, 4GB LPDDR4 memory, along with support for MIPI CSI-2 and PCIe Gen2 high-speed I/O. It is a small, powerful computer that can run multiple neural networks in parallel for applications like image classification, object detection, segmentation, and speech processing. This developer kit was a perfect fit to execute the algorithms once the data was trained. A USB webcam was used to capture images on the Jetson Nano. However, to train the data, a computer with an NVIDIA GeForce GTX 1050 Ti graphic card was used since this power GPU trained data faster than the Jetson Nano.

B. *Matlab*

MATLAB (an abbreviation of "matrix laboratory") is a multi-paradigm numerical computing environment and proprietary programming language developed by MathWorks. MATLAB allows matrix manipulations, plotting of functions and data, implementing algorithms, creating user interfaces, and interfacing with programs written in other languages. This programming language offers built toolboxes like Image Processing Toolbox™ which provides a comprehensive set of reference-standard algorithms and workflow apps for image processing, analysis, visualization, and algorithm development. This toolbox can be used to perform image segmentation, image enhancement, noise reduction, geometric transformations, image registration, and 3D image processing. MATLAB was used to learn basic image processing skills, such as edge detection using different algorithms and shape detection.

C. *OpenCV*

OpenCV (Open Source Computer Vision Library) is a library of programming functions mainly aimed at real-time computer vision. It primarily focuses on image processing, video capture, and analysis including features like face detection and object detection. OpenCV was widely used in the development of the project to read and write images and perform image processing tasks such as background extraction.

D. *TensorFlow*

TensorFlow is an end-to-end open-source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries, and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML-powered applications. Tensorflow was used to train the image classification datasets implemented using YOLO V3.

E. *Keras*

Keras is an open-source neural-network library capable of running on top of TensorFlow. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. This library was useful when implementing the image classification algorithm.

II.II Background Subtraction

Background subtraction is a technique which allows an image's foreground to be extracted for further processing. Generally, it is used for detecting moving objects in videos from static cameras. However, this method can be used to detect foreground objects from images. An initial image of the background is required to perform this technique. In the figure below, the subtract method of Open CV was used to perform background subtraction.

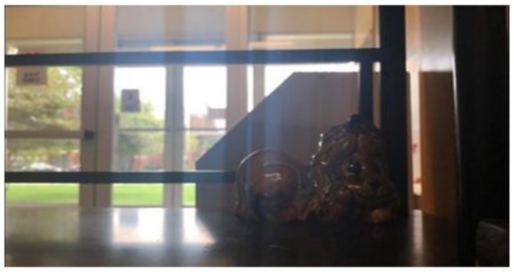


Figure 2: Initial background

Figure 3 – Figure 2

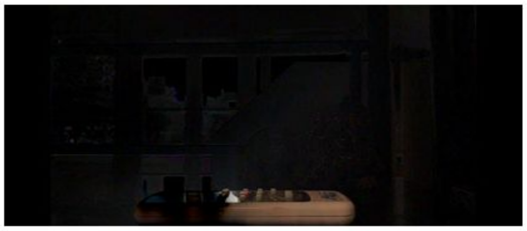


Figure 4: Foreground left after background extraction

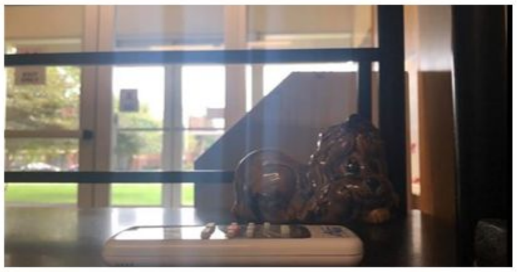


Figure 3: Image with foreground

Background subtraction could be useful in this project to detect the latest items added to the fridge. However, due to time constraints, the students were unable to implement this technique for the project.

II.III Edge Detection

Edge detection is a technique used in the image process for finding the boundaries of objects in an image. This technique was studied and practiced, as it was considered to be a useful method for this project. Though the final project did not directly include the code that was utilized for edge detection in an image, this concept was helpful in understanding how object detecting systems function in general. Edge detection algorithms detect discontinuities in brightness. There are numerous variations of these algorithms, including Prewitt and Canny. Prewit is a simpler method, but it is more sensitive to noise in an image. Canny uses a smoothing effect to remove noise, and it is immune to a noisy environment. Canny takes longer to implement to reach a real time response. Figure 5 below shows the difference in these two algorithms' output when running on an image of coins on a table. The coins are somewhat outlined in the images, yet the previously discussed differences are notable.

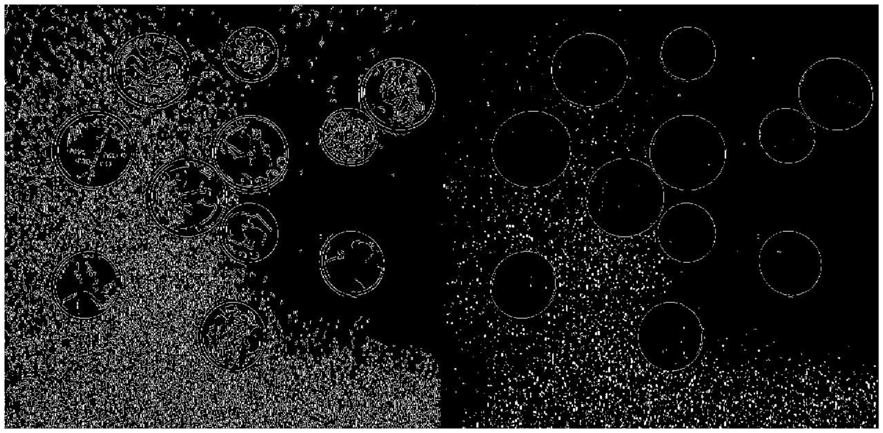


Figure 5: Canny algorithm vs Pewitt algorithm

II.IV Image Classification

Image classification is a technique that is used to classify or predict the class of a specific object in an image. The main goal of this technique is to identify the features in an image accurately. Extensive data training is required to classify objects using computer vision. For accurate classification, thousands of labeled images are needed per object. The task of building a dataset can be tedious and time-consuming. To remediate this problem, students used an open-source data set to train the model. Google's Open Image Dataset V6 contains over 600GB of labeled images. Students used YOLO V3 to implement the Image Classification.

III. UTILIZING YOLO FOR OBJECT RECOGNITION

III.I What is YOLO

Yolo is a network that is able to perform real-time object detection. Using deep learning algorithms combined with a dataset of images, YOLO is able to output bounding boxes around objects for which it is trained to detect. The YOLO network includes three central portions: the predictions vector, the network, and the loss function. The algorithm works on an image split into a grid, checking each spot individually for an object of the trained classes or a piece of the object. The prediction confidence (the probability that the object is genuinely belonging to the class) is calculated for each section, and uses non-max suppression to remove low probability boxes. A YOLO network contains convolution and max-pooling layers, along with two fully connected convolutional neural network layers. The loss function selects the bounding box with the highest intersection over union with the ground truth. This ensures that one bounding box will be utilized for a single identified object.

III.II Implementation of YOLO

This project's implementation of image classification was completed by training image sets specific to the systems requirements. Because the system is monitoring a refrigerator for the contents, the network is trained to recognize bottles of varying sizes and shapes. This is indicated by a bounding box within an analyzed image. Any detected bottles are scanned for text (Section IV.II) to ensure greater accuracy in the item's classification. In addition to the network being able to recognize bottles, it can detect various common food items within an image. The datasets in which these classes are trained are pulled from Google's Open Image Dataset V6. In Figure 6, an image is analyzed with a trained network to detect cheese.

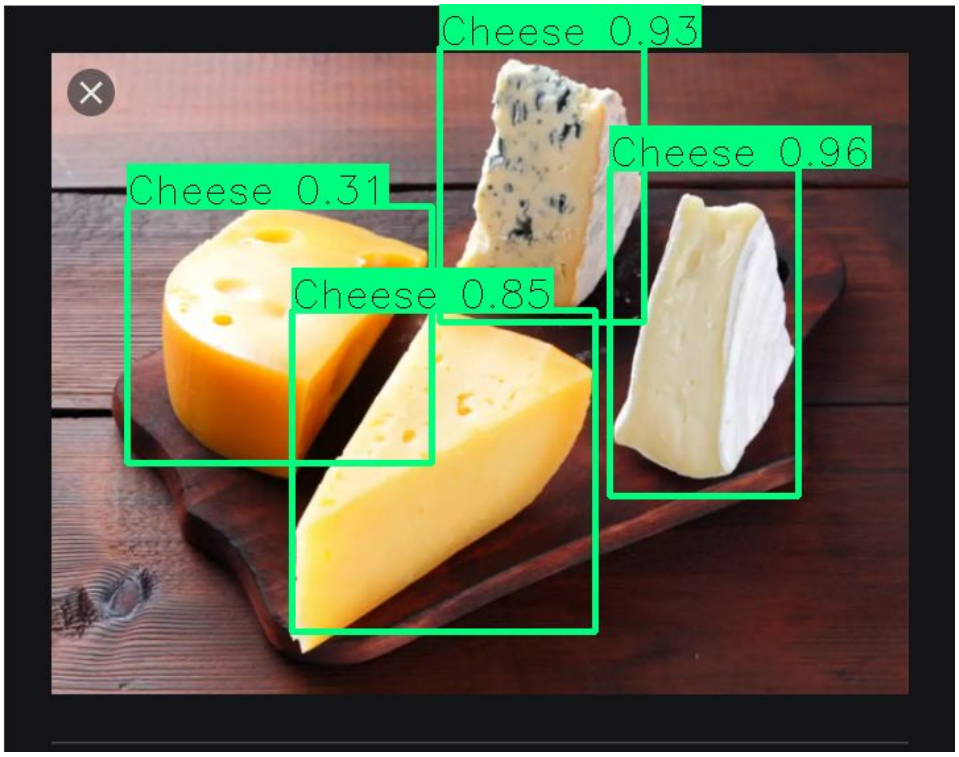
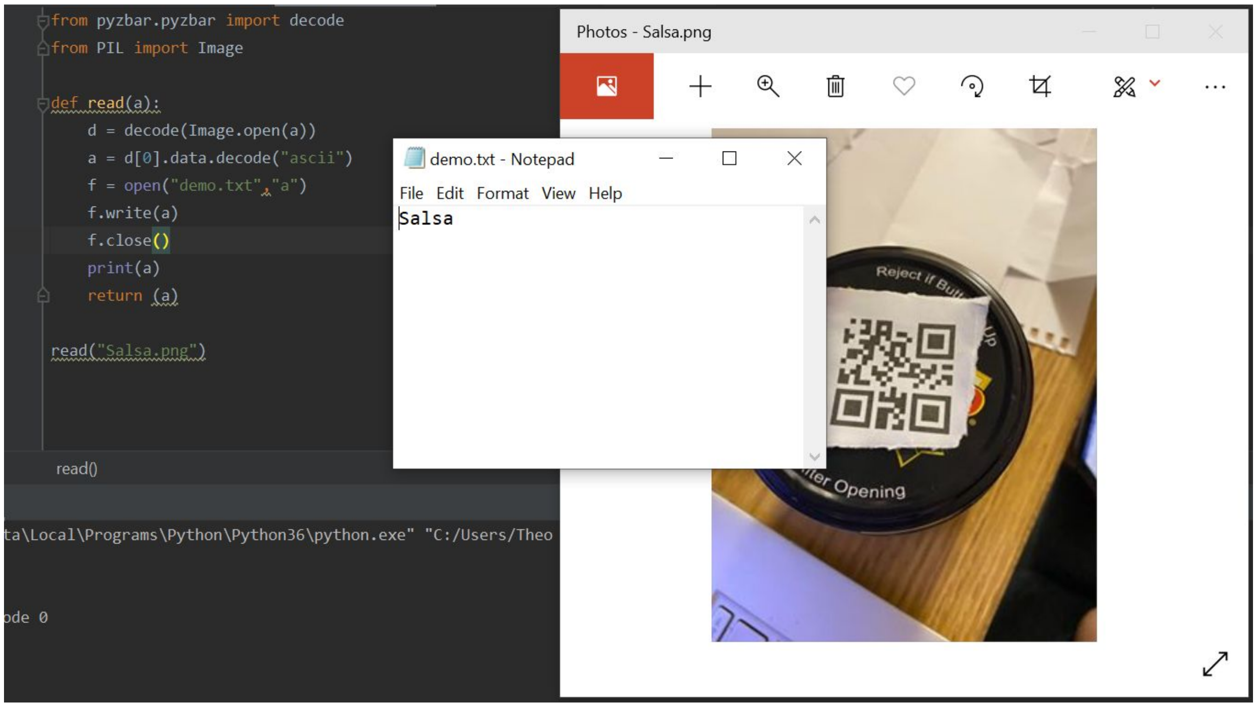


Figure 6: The students' trained network detects food items within images

IV. QR CODE AND TEXT RECOGNITION

IV.I QR code Recognition

The system begins an analysis of a specified image within a refrigerator by scanning for QR codes on objects. The item linked with the QR code is output to a text file to save the information. The QR code reader utilizes QR codes from an online generator and is not yet compatible with retail QR codes. This was implemented in order to eliminate inaccuracies from object classification and provide an alternative method for recognizing objects. The script that detects the QR code utilized the Python library of openCV to read the image and pyqrcode to scan the QR code. A barcode reader was also created and tested but was found to be less accurate than the QR code detector.



```
from pyzbar.pyzbar import decode
from PIL import Image

def read(a):
    d = decode(Image.open(a))
    a = d[0].data.decode("ascii")
    f = open("demo.txt","a")
    f.write(a)
    f.close()
    print(a)
    return (a)

read("Salsa.png")
```

```
read()
```

```
ta\Local\Programs\Python\Python36\python.exe" "C:/Users/Theo

ode 0
```

```
Salsa
```

Figure 7: The code to detect a QR code is run on a jar of salsa. This is output to a text file.

IV.II Text Recognition

There are numerous different bottles that may be present in a refrigerator, and the system works to classify what is inside each bottle through text analysis more accurately. First, if a bottle is detected in an image, a cropped image of the bottle is saved and run through a text analyzer to read the label on the bottle. This script is created in python, and it uses the pytesseract library. In order to accurately read the text, several threshold values need to be moved in order to adjust for lighting and variance within the images. These values are a work in progress, as the text analyzer does not have a high enough success rate in order to be considered complete at this time.

V. PROJECT COMPONENTS FOR FUTURE COMPLETION

In order to complete the system in a way that would be useful to a consumer, there are several additional aspects that need to be considered for future implementation. Each of these components was not completed due to time constraints, however, are fully possible to implement.

V.I Real-Time Detection

Monitoring a consumer's refrigerator will be most effective when real-time detection can be implemented. Allowing for automatic updates to the user's current system will be the most effective way to give users remote access to their inventory. Rather than utilizing single images and running the systems code once, a real-time detection system will be able to notice changes using a webcam. The webcam will take pictures of the inventory, updating the user in a specific way (discussed within the subsequent sections *V.II* and *V.III*). Every few seconds, the webcam will capture the image that will be used in the same way as the test images in the current system.

V.II Database Implementation

The product would be most effective if the data gathered from images of the inventory were output into an organized system rather than a text file. The database would hold items, quantities, and can expand the project scope to incorporate amounts of items and expiration dates. Allowing this information to be kept within a database would open opportunities for recipe recommendations, further helping the user in everyday activities. Based upon the foods within the refrigerator, a set of possible recipes could be generated and displayed to the user in some way (see section V.III).

V.III Mobile Application

A user interface for this system would be most effective in a mobile application. A mobile application interface could build upon the database design, allowing users to access their inventory and their recipe possibilities remotely. Real-time monitoring interfaced with this type of user interface would enable users to make decisions on what to purchase at a grocery store easily from a mobile device.

VI. CONCLUSION

To conclude, the system that the students have created has been successful in the goal of making considerable progress towards a refrigerator monitoring system. Trained neural networks were utilized in order to recognize everyday objects and provide an output based upon the classifications. QR code detector and the beginning of a text detection script were created to improve the accuracy for the user. There are many elements that were not implemented due to time constraints, but these elements embody the commercial aspect of the improved user experience. Real-time detection and a mobile user interface would enable the consumer to utilize the product effectively. Given the progress that has been made on the system, a considerable amount of technical work has been done to enable accurate monitoring of the inventory. Computer vision concepts were explored and effectively led to a workable design solution that allows smart technology to aid a consumer in daily life.