A Look into the Application of Computer Vision

To Merchandising Resets

Alitquan Mallick

St. John’s University

Queens, New York  
alitquan@gmail.com

Abstract

In the domain of retail, millions of dollars are spent in allocating slotting allowances for brand-new products. However, there is a large overhead in effectiveness due to the fact that substantial differences exist between plan-o-grams, UPC price tags, and the merchandise on the shelf. In this paper, I attempt to remediate some of these problems by applying convolutional neural networks to merchandise recognition and also, to a pre-existing algorithm for barcode detection.

Introduction*[[1]](#footnote-1)*

While the past decade has been momentous in terms of the incorporation of technology in a multitude of domain spaces, especially in pertinence to automation, there still lacks an integration of computer vision into the domain of retail businesses.

A valid application of computer vision, more specifically, could be in the niche of merchandising resets, a cyclical process that occurs in stores to replenish purchased stocks of merchandise, introduce new merchandise, and also to cut out non-lucrative/ discontinued merchandise. It is a standard procedure in any retail business. In fact, in a 1997 study, Grocery Manufacturers of America, Inc. reported that annually approximately 1,100 to 1,200 newly introduced products were added on average per retail store. Two years prior, the USDA’s Economic Research Service recorded a peak of over 16,000 new product introductions. (FTC 2003) And this product introduction is costly, with a surveyed retailer estimating that it spent $3.5 to $4 million annually in evaluating new items for introduction, and also estimating that the failure rate for new products is approximately 70%.

A recurrent issue in the reset process is regarding merchandise and tags. Personnel are often hired to inspect UPC tags, observe the display of the respective merchandise, and determine whether to replace this product, or whether there is space to cut in a new product. It is a time-consuming task that has a high rate of error, as tags are easily misplaced/damaged while objects may be misplaced. There is also always the impending possibility of human blunder.

Merchandising resets are not procedures that are substantially depend on human creativity; it is a process that solely deals with facts, namely the capacity of merchandisers to comprehend all that is within their vision. With such a mechanical nature, the potentiality of applying computer vision to it is evident.

Devising an algorithm that can streamline this process while minimizing the rate of error can save retail stores expenses that are paid to personnel. It may also prove beneficial to manufacturers trying to push new products. Consumers may also be benefited due to the resulting lower chances that desirable products not be in rotation. Lastly, application of this algorithm on a much larger scale can have broader implications, such as an automated inventory system based on image-processing.

Already, many researchers have proposed and even attempted to implement computer vision in this particular domain. For example, already, two extremely effective algorithms for barcode detection have been proposed, such as Tunistra’s Algorithm, a breakthrough algorithm that uses the intensity difference between bars to ensure that a gradient highlights the bars. (Tunistra 2006) Another method is that of Telkin & Coughlan, which similarly focuses on the gradient, but scans each edge pixel for opposite polarity pixels, and then calculating the entropy value for opposite polarity pixels. (Tekin & Coughlan 2009).

In this paper, I propose an efficient approach that recombines many of the aforementioned ideas. My approach involves utilizing a category-wise product detection model and a CNN-augmented barcode scanning algorithm. Although my approach was not conducted at the scale of a store belonging to a colossal supermarket chain, these are techniques that are meant to scale.

Approach

My approach will include two main phases: product recognition and barcode processing.

For product recognition, my approach can be said to be a hybrid one. While many approaches nowadays emphasize a classification approach, it is objectively inefficient and illogical given the nature of merchandising resets. Using only a classification approach would imply constant retraining of the model, especially with day-to-day changes such as product introductions or product-redesigns.

The ideal approach is a class-agnostic detector. This is a system able to a system able to detect any object regardless of class, and draw a bounding box around the detected object. However, this type of model is computationally-expensive to train.

Ideally, I envision a hybrid approach, using category-wise product detection followed by computing descriptors for these products and cross-referencing them in a library to find the closest match.

For the purposes of my research I opted to follow through with the category-wise detection, contrasting its performance with that of product-wise detection. As computing descriptors diverges from the rest of my methodology.

Barcode processing itself can be said to consist of two processes: barcode detection and barcode decoding. Barcode detection refers to the process of detecting the patterns of bars that a barcode consists of while barcode decoding is merely deriving the code from the detected pattern.

For detecting barcodes, I decided to build upon a novel algorithm, which I have already mentioned --- Tunistra’s Algorithm. The premise of this algorithm is using the properties of the bars in UPC tags to create a gradient for analysis. However, several defects arise due to factors such as properties related to the images taken, namely distance from which the image was taken, and also the environment. By applying my own transformation to images processed according to this algorithm, and then training a CNN to classify the resulting patterns, I hope to see some progress in dismissing these problems.

As for decoding the barcodes, I opted to use a pre-existing library. Decoding barcodes is simply just following the standards (in this case, UPC), and is thus not something that can be intuitively improved upon. While delicate image processing can lead to better accuracy with decoding barcodes, it is something that is outside the scope of my research.

Essentially, first I create two datasets of vanilla images – one for products and one for UPC codes. Then, specifically for the UPC codes, I modify the images using Tunistra’s algorithm. This would be the control dataset. Then I duplicate this set of images, and apply my own transformation on top of it. This would be the experimental dataset.

Next I annotate both data sets – the product dataset and the control UPC dataset. I make sure to annotate the product dataset two times – one annotating based on individual product, and the other annotating based on the category of product (can, bottle, shampoo, etc.). Then I apply an image augmentation pipeline to enrich and expand the datasets.

At the end, there were 3 datasets to be trained: the control UPC dataset, the individual product dataset, and the category-wise product dataset. I separate the datasets into training and validation sets. I train the models and made comparisons. I use the model trained on the control UPC dataset to classify UPC tags in the experimental dataset. Then I compared the accuracies of product-wise detection with that of category-wise detection through a variety of tests. I also observed the resulting differences amongst the different types of bibs in the UPC dataset. Below is a more in-depth look into more specific aspects of this methodology.

**Tools and Datasets**

Implementation will be done using Python, which has several libraries are useful for my purposes. Two such libraries are OpenCV2 and the ZBar Library.

I will be using datasets I have created. This includes a dataset consisting of different retail items. I will be observing this dataset through two lenses: the products being labeled according to shape, and the products being labeled according to individual product names. Overall 338 photos are present. There are 24 different individual products featured, with each one of these belonging to one of 3 classes. These photos were taken stressing variation in lighting, background, and orientation to acquire more generalizable results. Annotations per image range from 1 to 23.

For applying Tunistra’s algorithm, I similarly have made my own dataset consisting of images taken of retail shelves. These images have been processed according to Tunistra’s algorithm, resulting in a binary image with contours of the UPC codes (see *Tunistra’s Algorithm* for more details).While initially, I had taken 213 photos, after processing, only 173 of these photos had generated contours substantial enough that they may be of use. Annotations per image range from 3 to 32.

I divided the types of resulting UPC contours into 4 categories: normal, partial, long, and bibs. *Normal* is the default contour (a solid gray rectangle) I encountered when the images in-question are taken at a closer range. Partial contours are those that, due to environmental factors, display an unconventional contour. Long contours are how the resulting contours are displayed when Tunistra’s Algorithm is applied upon images taken at longer ranges. Lastly, bibs refer to contours resulting from elongated tags, which have a distinct pattern from the prior 3 categories.

For both image sets, I decided to use an image augmentation pipeline to expand the dataset by applying combinations of various transformations onto the already-taken images. This includes translations, rotations, dilations, changes to saturation, etc. This was a great way to diversify my dataset and also provide further annotations for training.

Upon testing my methods, I found that I had 10,016 annotations for the augmented barcode database, and 8,878 annotations for the augmented product database.

Barcode Detection

As mentioned before, I will be using Tunistra’s Algorithm, although modified to reflect my proposed method.

**Tunistra’s Algorithm**

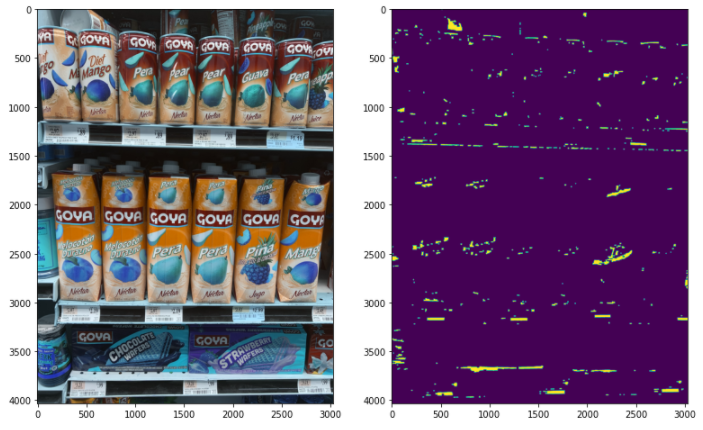
This algorithm was one of the first created that effectively detected barcodes. In fact, most methods today build upon Tunistra’s novel idea to use the gradient resulting from the bars.

The first step is to use Sobel kernel to estimate the gradient in the X and Y directions.

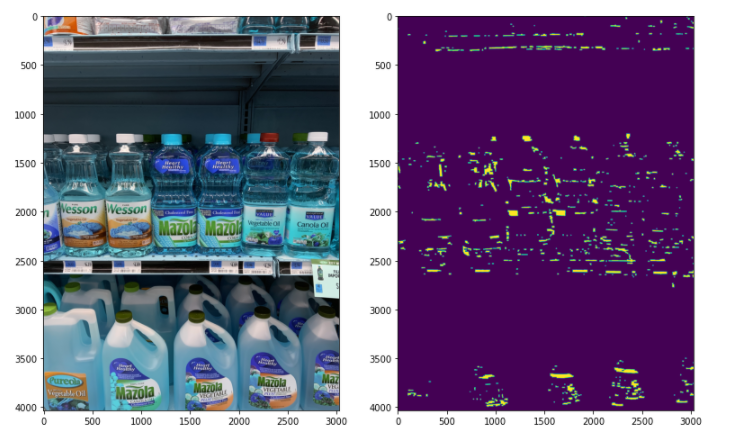
Next, the gradient image is thresholded and the pixels having a higher gradient value are singled out.

Morphological erosion is incurred to remove thin, irrelevant segments that were fused into the region via the dilation.

Lastly, a morphological dilation merges nearby non-combined components to produce a region.

**However, there are several weaknesses with Tunistra’s Algorithm. The angles at which the images are taken affect the contours detected by the algorithm. Slight wear-and-tear can impact the purity (or thickness) of the contours. The top row of UPC tags can be obscured by a combination of wearing on the tags, and also by the fact that due to the camera angle, the protruding edge of the top shelf masks the lower portion of their barcodes.

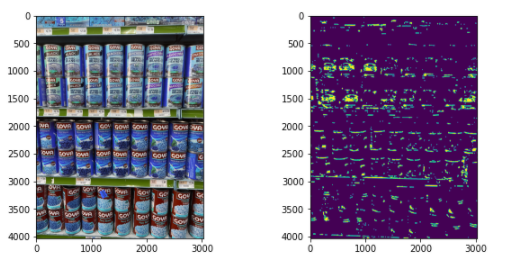
*Figure 1 – Wear-and-Tear of UPC Tags*

This is even more problematic due to the fact that at a distance, the Sobel derivative can pick up reflections of light or dense text as contours similar to those of barcodes.

*Figure 2 – Interference Caused by Reflection*

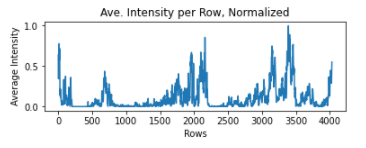
There is also the issue of distance --- as distance increases, sharper lines result from the increase in density of detail. As a result, objects such as letters result in the same contours as

barcodes would have generated.



*Figure 3 – Problem of Density of Detail*

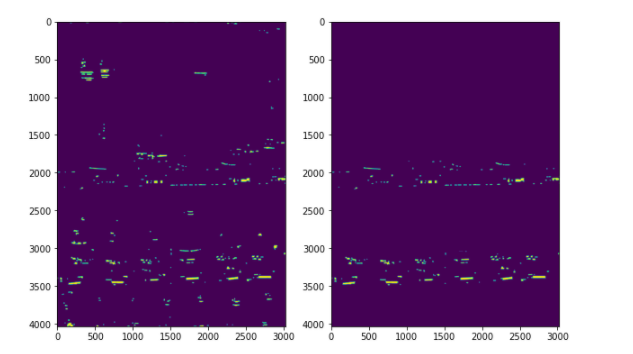
**Modification**

 As previously mentioned, the effectiveness of Tunistra’s Algorithm fails as a function to the distance of the images taken. This is due to the increasing amount of noise generated since more and more detail is compressed into a tighter space, leading to the bi-directional Sobel kernels misclassifying these details as bars.

*Figure 4 – Average Intensity per Row of Left-Image in Figure 5*

A simple way to reduce all of the noise caused by merchandise is to focus on the space between the merchandise, which is where, coincidentally, the UPC barcodes are found. The best way to do this is to take the image after the dilation phase and calculate the mean intensity per row of the image. Since all the barcodes are vertically within a few centimeters of each other, the areas where the barcodes are located will have a significantly higher average intensity, especially if rows that have an intensity lower than an arbitrary threshold are left out (see Figures 4 & 5). Hence, these regions can be extracted with the intention of creating a more accurate model.

Even with the reduction in noise, some of the UPC codes do not have defined contours. But with the reduction in noise, there is a less muddled space for classifying contours. This presents itself as a perfect opportunity to utilize a Convolutional Neural Network. Training a CNN to appropriately classify these contours should show some improvement over the original algorithm.



*Figure 5 – Result of my Denoising Modification*

**Barcode Decoding**

After getting the regions of interests, some pre-processing, such as inversion and further thresholding will take place.

Before passing on the newly acquired regions of interest to ZBar Library, a Python library, that will be used to scan these areas for barcodes.

Results

Overall some positive results were achieved with respect to all of the modifications put in place.

**Observations**

First, some observations should be mentioned. For example, the quality of the photo and environmental factors is imperative in enhancing results. Photos taken at an angle perpendicular to the ceiling and floor seem to yield more contours with higher density per contour, meaning that there is a much higher chance of the UPC code being successfully decoded.

Additionally, shelves with lights nearly perpendicular to the UPC tags yield the poorest results. Glare from the lights prevents the gradient of the bars from accurately manifesting themselves during a photo.

**Detection by Category vs. Detection by Product**

Instead of choosing to detect by the exact product, I decided to differentiate by category instead. However, I decided to train a separate model to classify merchandise according to individual product rather than by name control. Detection in this manner proved to be inefficient in every regard.

First of all, there was the training of the model itself. The category-wise detection model took an average of 3,359.2 seconds per epoch to train over the course of 7 epochs, whereas the product-wise detection model took an average of 3,523.8 seconds per epoch.

Additionally, additional time was taken proofreading the annotations for errors. With over 25 different labels (types of annotations) present in product-wise dataset, the probable margin of error is high, especially taking into account the fact that a difference of a mere character would require intervention to get the model to work. With this in consideration, it took an average of 4 seconds per annotation for the category-wise detection model, and an average of 11 seconds per annotation for the product-wise model.

Then there is the issue of accuracy. While the product-wise detection model performed accurately for its intended purpose, it was still less accurate in comparison to its competitor. The biggest problem seemed to be errors in distinguishing between products in the same category, and more so, errors in distinguishing between products of the same brand. Running this model on a set of 12 images totaling 134 annotations, all containing Goya cans, the model only accurately labeled 98/134 (73.1%) of these annotations.

When fed a new image with products that were neither in the testing or training test, the category-wise model was able to accurately categorize the product into one of the three pre-defined classes. Of a set of 7 images totaling 38 annotations, the category-wise model was able to accurately label 36/38 (94.7%).The product-wise model, on the other hand, simply just miscategorized objects in the model with the closest match, seemingly. On a set of 4 images totaling 23 annotations of the same class, the product-wise model was only able to accurately label a *Red Bull* can of a slightly different color. This, in summary, presents the biggest advantage of category-wise classification: flexibility.

With new products constantly being added into inventory, combined with product re-designs, a model that classifies based on the specific product is impractical due to the fact that it cannot generalize. It is also impractical to spend so much time reviewing the annotations during the training process. Product-wise categorization, in some manners, seems to incur the same problems that traditional relational databases are prone to, but on an exponential scale.

**Barcode Detection on Pre-Processed Images**

In an effort to compensate for a lack of accuracy with respect to environmental factors, I decided to imbue Tunistra’s Algorithm with the integration of a Convolutional Neural Network. The basic premise was to overcome the problem of false-positive contours by training a neural network to recognize the true-positive contours. This was attempted in a variety of ways.

For example, as mentioned in the *Tools and Datasets* section, I decided to use 4 different types of classes of labels to describe the contours of the barcodes. This was to more specifically gauge the improvements made by my method.

Without training the model (Model A) on de-noised images, training the model and testing it on 11 new images yielded some interesting results. These images consisted of 18 normal labels, 13 partial labels, 20 long labels, and 23 bibs. The model accurately detected 13/18 (72.2%) of the normal labels, 8/13 (61.5%) of the partial labels, 4/20 (20%) of the long labels, and 18/23 (72.2%) of the bibs. However, 97 bounding boxes were drawn for the 74 annotations, with only 43 of these bounding boxes being precise enough to be considered a match. While there was some improvement in terms of accurately locating UPC codes, my model was still inefficient. However, this is still a remarkable improvement over the CNN-less implementation[[2]](#footnote-2) of Tunistra’s Algorithm, which was only able to obtain 7 tags, all of which were of the UPC variety.

Training the model on denoised pre-processed images (Model B) did not yield much improvement. On the same set of data that I used for the previous trial, Model B detected 14/18 (77.8%) of the normal labels, 10/13 (76.9%) of the partial labels, 1/20 (5%) of the long labels, and 14/23 (60.8%) of the bibs. There was a slight increase in accuracy for the *normal* and *partial* labels, but *long* and *bib* labels saw a tremendous decrease in accuracy. While Model A had an overall accuracy of 58.1%, Model B had one of 48.6%. However, the model was efficient in that it only drew 83 bounding boxes for the same 74 annotations, and also saw improvements in two categories.

However, testing both models on de-noised datasets saw minimal improvement in accuracy for both. Model A saw a jump in overall accuracy to 63.2% in only 83 bounding boxes while Model B saw a jump to 55.1% in only 72 bounding boxes. Hence, while training the model on de-noised data may not necessarily improve accuracy, applying de-noising to the images being classified may actually do so.

Overall, mixed results were observed. While applying my denoising method did increase accuracy somewhat, this was in a very controlled setting. I chose a test set that denoised perfectly, without any significant regions being truncated, as they might have been otherwise.

The object detection model performed amazingly when it came to *normal* and *bib* UPC contours. While accuracy was mediocre for *partial* and downright abysmal for *long* tags, both were cases that were rarely detected by the original Tunistra algorithm.

Discussion

For the most part, this paper suggests that the best way to combat environmental interference (noise) generated by Tunistra’s algorithm is to use patterned tags, such as those of bibs. The distinct contour (caused by the distinct pattern of the bib) was the reason why these types of tags were so sensitive to detection by both models.

Miscellaneous things such as the material and curvature of the shelves is also something that should be noted. Metallic shelves are more likely to reflect light such that the photos are blurry, and the barcodes, as a result, obscured. Increased curvature of the shelves increases this effect.

*Figure 6 – Resulting Contours of UPC bibs*

As for taking pictures, Tunistra’s algorithm seems to be more effective with higher quality images, as the bars of the UPC barcodes are more apparent. While my implementations did see some success with recognizing the UPC barcodes at a distance, the results were not impressive enough to warrant further introspection, especially given the fact that detecting codes from long-range would have more to do with hardware. However, the research that I have done indicates that these implemented methods to do show promise with images taken at close range. As such, to implement any sort of automation in the retail domain with this method, I would suggest some sort of machinery that takes these images at close range.

References

FTC. 2003. Slotting Allowances in the Retail Grocery Industry: Selected Case Studies in Five Product Categories. Federal Trade Commission. Washington, D.C.

Mahee, T.M; Chowdhury, A.I; Rahaman, M.S; and Ahamad, R. 2019. A Study on Image Processing to Facilitate Business System by Multiple Barcode Detection. Department of Computer Science and Engineering, Ahsanulah University, Dhaka, Bangladesh.

Katona, M. and Nyul L. Efficient 1D and 2D Barcode Detection Using Mathematical Morpholoy. Department of Image Processing and Computer Graphyics. University of Szege

Rusakoff, D.B; Tomasi, C.; Rohlfing, T; and Maurer, C.R. Image Similarity Using Mutual Information of Regions. Department of omputer Science, Stanford University, Stanford, CA.

Tekin, E., Coughlan, J.M. 2009. An algorithm enabling blind users to find and read barcodes. In: Workshop on Applications of Computer Vision (WACV)

Tonioni,A; Serra, E.; Stefano, L.D. 2019. A deep learning pipeline for product recognition on store shelves. University of Bologna.

Tuinstra, T.R. 2006. Reading Barcodes from Digital Imagery. Cedarville University

Varol, Gul & Kuzu, R. 2015. Towards retail product recognition on grocery shelves. Department of Computer Engineering and Electronical and Electronics Engineering.

1. Copyright © 2020, by the authors. All rights reserved. [↑](#footnote-ref-1)
2. The CNN-less implementation simply returns the largest *N* contours from the binary images, which may not necessarily be accurate [↑](#footnote-ref-2)