The Pennsylvania State University

The Graduate School

College of Engineering

EMPLOYING TEXT FEATURES FOR VISUAL ASSISTANCE IN NAVIGATION AND CLASSIFICATION

A Thesis in

Computer Science and Engineering

by

Jake T. Eden

Submitted in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science

May 2018

The thesis of Jake T. Eden was reviewed and approved* by the following:

Vijaykrishnan Narayanan
Destinguished Professor of Computer Science and Engineering
Thesis Advisor

John M. Sampson
Assistant Professor of Computer Science and Engineering

Bhuvan Urgaonkar
Associate Professor of Computer Science and Engineering
Graduate Program Chair

*Signatures are on file in the Graduate School.

ABSTRACT

This thesis explores the potential of harnessing text extraction within a constrained environment aimed at guiding and augmenting a visually impaired end-user's shopping experience. Most current visual assistance pipelines attempt to track user location for indoor navigation; some attempt to classify relevant products of user interest; and others provide relevant contextual information by first extracting local environmental text. However, no prior art explores how text might be used as an input to other parts of the visual assistance pipeline. For instance, can text extraction improve localization and product classification in addition to providing relevant contextual information? Thus, this thesis explores the viability of introducing text into such seemingly disjoint problem spaces and ultimately concludes that environmental text extraction can enhance both indoor localization and last mile product classification. Additionally, this thesis answers the general problem of where in the pipeline multidimensional inputs should be combined.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

Thanks to my thesis advisor, my co-advisor, and my honors advisor: Vijay Narayanan, Jack Sampson, and John Hannan, respectively. They provided valuable guidance and feedback throughout the process. Also special thanks to Jinhang Choi, Dafang He, and Xiao Yang for allowing me to use their neural nets for the purposes of this thesis. This work was supported in part by the NSF Expeditions in Computing Program: Visual Cortex on Silicon CCF 1317560. The findings and conclusions expressed in this thesis are my opinions and are not reflective of those of any funding body.

Chapter 1

Introduction

Much of modern life relies upon human sight, which puts those with handicapped or entirely absent vision at a distinct disadvantage. According to the national federation for the blind, about 1.3 million US citizens are legally blind and about 10 million Americans are at least visually impaired [6]. While the zeitgeist of today values the needs of such underprivileged individuals, the technology to meet those needs is not quite there. Thus, much research has taken place to visually assist the blind in common situations present in their daily lives [3,5,8]. The enablement of blind shoppers has been of particular interest as solving such a problem relies upon a combination of many research areas: indoor navigation, product classification, low power embedded systems, and so on.

Indoor navigation has been a particularly active area of research as there’s currently no highly reliable, highly accurate method to get location without resorting to complex solutions like LIDAR [25], which is impractical in the context of a grocery store environment. Instead, computer vision pipelines are currently being explored to perform simultaneous localization and mapping (SLAM) that often rely upon key point detection and potentially a combination of other sensors like the gyroscopic movement of a phone [25]. However, a significant challenge is re-calibrating location estimates to avoid drift, the gradually increasing error over time, present in many SLAM systems. This thesis assesses the feasibility of limiting drift by determining location based purely on nearby text in the wild.

However, before expanding on why text specifically might be used for this purpose, consider also the goal of identifying products for the blind (i.e., classifying grocery store items). The current trend in computer vision is the exploration of convolutional neural networks (CNNs) for classification as they can be extremely accurate with a large enough network and sufficient number of training images, but a typical grocery store environment averages around 45,000 different products [3]. No literature has appeared in which a convolutional neural network has been shown to accurately classify such a large number of classes. In the face of such a problem, this thesis also assesses how text can be applied as well as assessing the general problem of where in the pipeline multi-dimensional inputs should be combined.

The use of text to augment such situations is a good choice for several reasons. Firstly, consider that a grocery store is a constrained environment rich in text. In such constrained environments and with the aim of visual assistance, locating and recognizing nearby text is naturally desirable. For example, text extraction can augment a blind end-user’s shopping experience by helping to identify the exact product SKU, informing the user of the current sale price, providing the basis for queries about nutritional information, and/or querying for relevant recipes involving said product. Obviously, text extraction can be beneficial to the overall shopping experience. Thus, pipelines involving visual assistance to the blind have a strong incentive to include text detection and recognition as key processing stages. Therefore, if text extraction is to be used in such an environment anyway, it leads to the natural question of it can help serve as an additional input dimension to other concurrent processing systems, which is the crux of this exploration.

Thus, this thesis employs state-of-the-art deep neural networks for English text detection and recognition. These nets were used to gather a corpus of textual data with which to represent aisles and subdivisions within those aisles. Different text filtering methods were employed, and ultimately this thesis contributes answers to the questions: can current text extraction provide enough granularity to perform meaningful localization, and how should we handle the high degree of noisy recognition present in the state-of-the-art?

Furthermore, text extraction was used to refine the results of last mile classification. The output of a convolutional neural network trained on ten classes of grocery store products was used to train a simple SVM classifier in which processed text input features provided the other input dimension. One might envision the necessity of such an approach in a grocery store environment where small pieces of text sometimes serve as the only differentiation between two products. However, demonstrating that this multidimensional approach to classification improves last mile accuracy is to be expected (more data should intuitively lead to better results), and thus, this thesis goes further to explore more generally what stage in the pipeline said features should be combined for the best performance. Should one neural net output be used as a pre-filter for the other neural net, or should both outputs be combined independently? In other words, the general problem of feature combination was explored, and it was found that the best performance occurs when these input features are combined at the end of their independent evaluations instead of linearly combining the output of one classifier to serve as a pre-filter towards the other.

Chapter 2

Background and Related Work

In this section, we explore existing visual assistance systems, the technologies they use, the technologies employed in this thesis, and how this thesis naturally builds upon prior work.

Conventional Assistance Systems

Over the years, there’s been a number of different approaches and systems devised to assist visually impaired individuals in varying contexts. Often times these methods go beyond just images and use other sensors like infrared cameras [8], accelerometers [23], or a combination of many sensors from magnetometers to light sensors [24]. In 2001, [15] developed a system to assist the visually impaired to navigate around their local campus. Like many classical approaches to this problem, it combined sensor technology and smart wearables. Specifically, it relied upon outdoor GPS positioning, which interfaced with a backend Geographic Information System (GIS) over Wi-Fi to generate user paths. The end-user could then interact with the system via voice to text commands. The use GPS or the more accurate DGPS (Differential GPS) for outdoor navigation was very common around that time, appearing in different research implementations with the same GPS fundamentals [17-19]. Of course, it’s well known that GPS cannot be used indoors, and so in 2004, the researchers extended their system to include ultrasound beacons that estimate position by calculating flight time differences [16].

This approach whereby researchers place down environmental transceivers or some such pre-existing infrastructure is also quite common. For instance, the use of RFID tags was explored in the PERCEPT system [5], and within the past few years, the potential for indoor navigation via low-energy Bluetooth beacons has been explored with Google recently filing a patent on their own implementation [21], and preliminary results indicate a sparse distribution can support localization within about 5 m (or potentially within 3 m if there’s a high number of beacons) [22]. In contrast, pure computer vision approaches aren’t quite as common, but they too often rely on environmental markers [7]. Using such environmental infrastructure is time-consuming to set up, doesn’t scale well, and depending on the technology is susceptible to interference.

Even when a pure computer vision approach is used without such a constraint, it's often in the context of performing simultaneous localization and mapping (SLAM), which attempts to construct a map of an unknown environment while tracking current user location within it. One approach in 2010, uses stereo cameras and the well-known fastSLAM to navigate through a small environment while avoiding collisions [20]. Its general feasibility is up for question as evaluations were performed within a small environment (thus mitigating most drift) and done over a short time scale that does not demonstrate loop closing capability. A 2015 survey on SLAM research concludes that pure computer vision approaches have yet to demonstrate reliably accurate long-term localization outside of small environments [25].

Thus far, all aforementioned approaches have not relied upon text in any way as an additional input dimension. The few prior works that do use text go only so far as to use it for landmarks similarly to the aforementioned marker approach. For instance, [4] suggests using text landmarks for indoor robot navigation, but their experiment is purely focused on how well they can extract text from the wild, leaving the navigation for future work. Likewise, more recent work in 2015 such as [9] uses certain signs as landmarks with which to inform navigation but again leaves the actual navigation and related experiments to future work.

It is believed that this thesis is the first to show concrete localization results based purely on textual features and is the first to explore the applicability of text towards localization without using predefined, fiduciary markers.

Additionally, prior work tends to focus purely on navigation or classification. Combining the two by exploiting the same input features is not generally done and there's little literature on using text for refining last mile classification. Prior work like [9] has used the text in the tags of online annotated images to help classify objects. However, this naturally isn't viable in real world images that are not annotated. And while [12] uses the same first part of a CNN to feed discrete output layers (i.e., it feeds multiple domains through shared layers), this approach involves internally training a net with respect to these different domains. Furthermore, [13] trains an SVM classifier on the output features of a CNN to achieve high accuracy classification results, but this is still only using a single neural net and doesn't answer the question of how to combine the two nets.

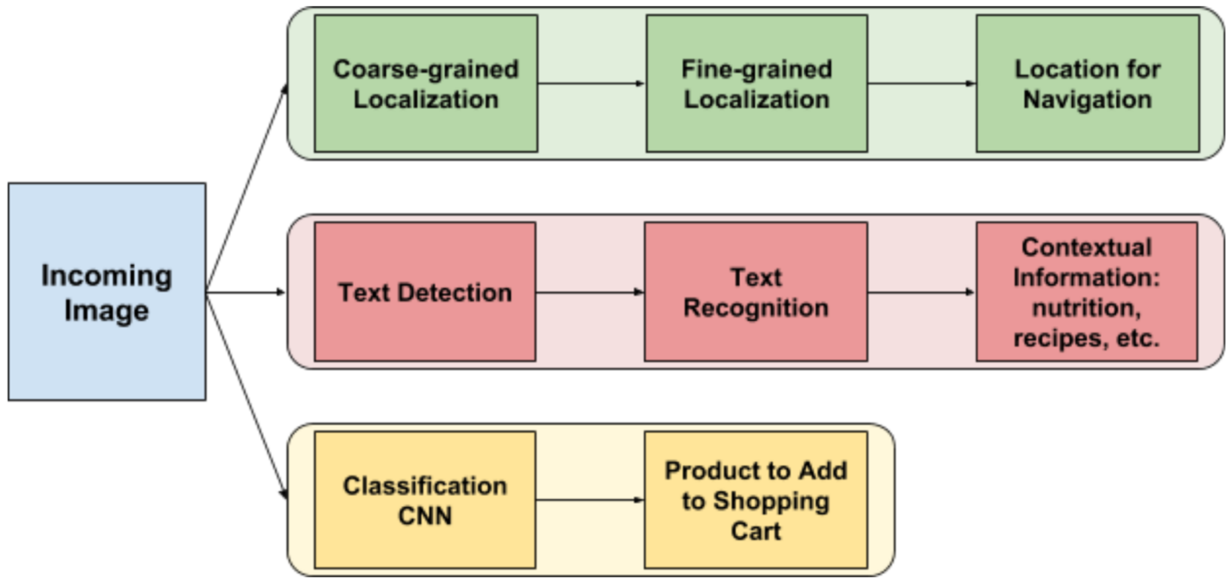

Incoming Image
Coarse-grained Localization
Fine-grained Localization
Location for Navigation
Text Detection
Text Recognition
Contextual Information: nutrition, recipes, etc.
Classification CNN
Product to Add to Shopping Cart

Figure 1. Typical visual assistance system with multiple, discrete pipelines.

Much of the aforementioned research discusses one processing pipeline to assist the blind, and when multiple systems are used, they're not in tandem as shown in Figure 1 in which the top, middle, and bottom paths respectively represent the localization and navigation pipeline, the text extraction pipeline, and the product classification pipeline. However, this thesis proposes modifying the processing pipelines as seen in Figure 2. Thus, they're no longer independent from one another. The text pipeline can be used to augment course-grained localization, and it can also be used to augment last mile classification. As can can be seen in Figure 2, there are several possibilities this thesis explores with regards to combining the text extraction pipeline with the product classification pipeline. One net might be used as a pre-filter for the other, or it's possible to use both nets independently and then combine their outputs to achieve to the final classification. The different dashed arrows represent such different flows.

It's likely that one of the reasons such literature is yet to exist is because text extraction in the wild has only recently become decently accurate [29]. The reemergence of neural networks for deep learning has produced both better text detectors and text recognizers that can operate in real time. Given the small memory footprint of text and its usefulness in providing relevant contextual information, exploring how it can augment other facets of the assistance pipeline is a natural question. Additionally, the grocery store environment is ripe with textual information with the exception of the produce section and aisles with fogged glass like the frozen section. In fact, when excluding those two sections, a qualitative human examination of over 130 products indicates that over 80% of them are useful for text extraction (i.e., they have easily extracted text of sufficient size). This means it's likely nearly every image of the grocery store will include products with extractable text, and this fact provided some of the initial motivation for this thesis.

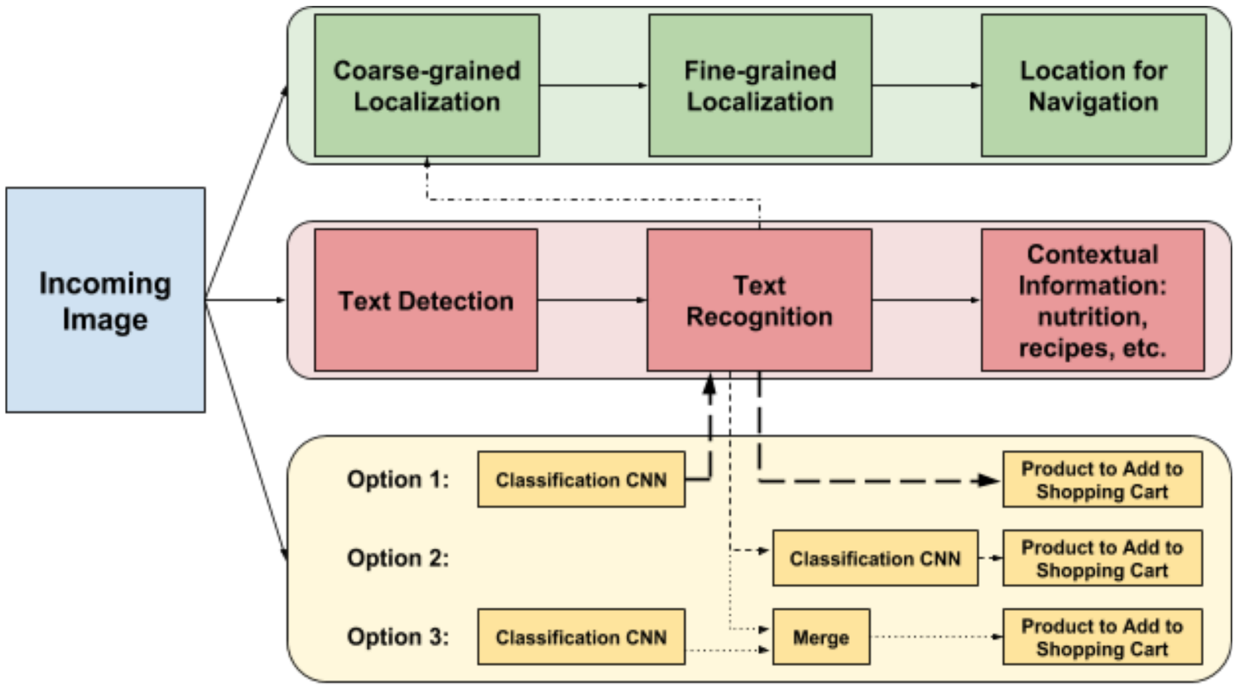

Incoming Image
Coarse-grained Localization
Fine-grained Localization
Location for Navigation
Text Detection
Text Recognition
Contextual Information: nutrition, recipes, etc.
Option 1:
Classification CNN
Product to Add to Shopping Cart
Option 2:
Classification CNN
Product to Add to Shopping Cart
Option 3:
Classification CNN
Merge
Product to Add to Shopping Cart

Figure 2. Modified visual assistance system pipelines explored in this thesis.

Neural Networks

This thesis makes use of three state-of-the-art neural networks—one for text detection, one for text recognition, and one for product classification—, and thus a basic understanding of their workings is necessary, especially since the recent rapid adoption of neural networks in the computer vision community is what has allowed text features to become accurate enough for this thesis to exist [29].

At the heart of every neural network is the perceptron. A single perceptron is relatively simple. It takes as input a vector of numbers and a vector of corresponding weights. The dot-product is performed on the two vectors, and the resulting number is used as input into an activation function. In most modern implementations, this is a Rectified Linear Unit (ReLU) activation function [26]:

$$f(x) = \max(\mathrm{x}, 0)$$

Thus, the output can either be 0 or greater than 0 (i.e., not fired or fired). In a certain sense, this resembles the binary nature of the neurons within the brain. Note that a single perceptron can classify linearly separable data. To classifier non-linear data (like objects within an image), multiple “layers” of perceptrons must be connected together.

Fully connected neural networks, ones which interconnect each perceptron between adjacent layers quickly consume large amounts of memory, require a long training time, and consume more energy/time during the inference stage [1]. In order to reduce the sheer number of neurons and weights in vision tasks, convolutional neural networks are used. They take an image as input but do not apply a perceptron per pixel and do not retain the dimensionality of the image throughout the layers due to a technique called max pooling. Each “convolution layer” uses filters composed of a single perceptron where the weights of the filter (or kernel) are the input weights of perceptron. How many filters to use is up to the designer of the network. After one or two convolutional layers, there’s often a max pooling layer, which usually reduces the dimensionality of the prior layer by 4x as max pooling simply retains the maximum value of four neighboring values.

After so many layers and reductions in dimensionality, it becomes computationally feasible to have one or two fully connected layers (or even more in recent iterations). To convert an *m*-dimensional feature vector into a *n*-dimensional vector where *n* represents the number of classes, we simply have a fully connected layer with *n* nodes. This marks the end of the generated feature vector. A classifier (like

SVM or softmax) is now needed at the end. Typically, softmax is used and is what the nets used in this thesis employ. It’s a somewhat simple function:

$$\mathrm{f}(x_i) = \frac{e^{x_i}}{\sum_{k=1}^{n} e^{x_k}}$$

It guarantees a couple properties. One, that no negative values are produced due to the exponentials. The second (stronger claim) is that everything is scaled between 0 and 1. On an intuitive level, this can be interpreted as producing likelihood probabilities for each respective class.

Neural networks can be trained in either a supervised or unsupervised fashion. However, for the purposes of our discussion, the focus will be on supervised learning—all the neural networks used in this thesis were trained in a supervised fashion. To achieve good performance, it’s recommended to use more images (sometimes in the thousands), if there’s a large number of classes or particularly complex ones. It’s also possible to decreases the number of needed images if a pre-trained model is used. This is because the pre-trained model can share higher order features that earlier convolutions would extract, like edges or certain color patterns, and the later stages can be re-trained independently.

To train convolutional neural nets, a loss function is used to determine how much each weight contributes to the current loss, and with a form of gradient descent, we can try to minimize loss. To calculate the amount of loss per neuron, we use a form of the chain rule. This propagation of error backwards through the net is known as back propagation and is fundamental to the training of all neural networks. However, the neural nets used in this thesis were already trained, and thus only their inferences were needed.

Text Detection and Recognition Networks

The text detection network was graciously supplied by a fellow researcher, Dafang He, who based his architecture off of the one in [27], and the text recognition was supplied by his co-researcher, Xiao Yang, who details the architecture in [28].

The text detection architecture is composed of a series of fully convolutional layers stacked upon one another such that the feature map sizes at each layer continually shrink by half while the number of output channels increases. Because text can exist at many sizes, a detector would ideally use features from both the shallow and deep layers. Thus, [27] feeds and combines the outputs of the fully convolutional layers to the feature merging branch of computation whose gradual concatenation of layer outputs is finally fed into the bounding box descriptor layer.

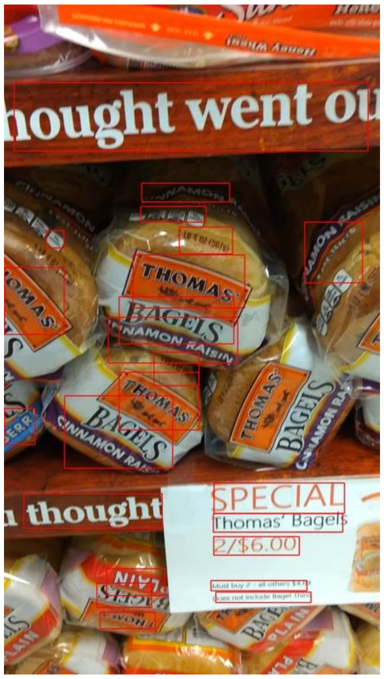


Figure 3. Example output from the text detection neural network

Ultimately, the net produces many bounding box guesses as to where text may be located. Non-Maximum Suppression is then applied such that overlapping bounding boxes are discarded, and the bounding box in that location with the highest score is kept. Note that unlike in the corresponding paper, the detection net used in this thesis does not produce rotated bounding boxes. However, clearly it's possible, and if rotated bounding boxes were produced, it would likely improve our results.

After detecting text location with the detection net, the image and associated bounding boxes are then sent to the text recognition network. When the text from the image in Figure 4 is fed to the recognition network, it produces the following output words: SPICY, FLAVORED, NACHO, I0O, Pal, Dontes, and BRAND. As can be seen from that example output, it usually produces a nontrivial amount of noise, but it still works well enough in the sense that it also extracts the desired text that is exactly or close to the actual word.

Many similar nets attempt to break down recognition to a per character level. However, this tends to be less accurate due to kerning (the spacing between words) and possible distortions. The context information of whether or not surrounding characters exist provides an extra layer of input towards the ultimate recognition task. Additionally, the network used combines both a CNN and a recurrent neural network (RNN) [28].

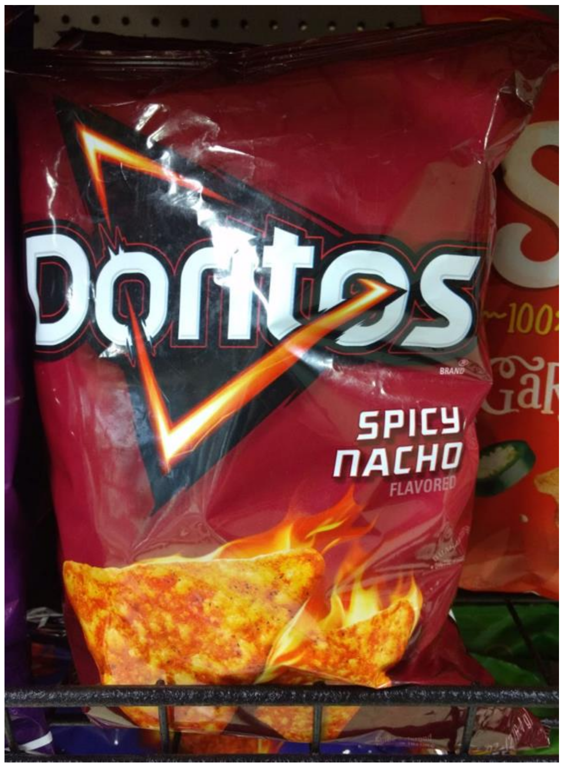


Figure 4. Example input image to recognition net

An RNN is built upon the same perceptrons but now there's feedback loops among some of them. An RNN is useful because its feedback loop is time sensitive. Thus, perceptrons that have responded to characters identified earlier in the image have an impact on the classification of later ones. RNNs are typically used in text/speech analysis and thus make sense as a later stage within the text recognition pipeline.

The first part of the used network is typical, composed of a series of convolutional layers. This CNN portion creates a sequence of features. For instance, when the input image is a word, the CNN traverses it left to right and generate the sequence of features, which will later be interpreted to mean certain characters. This sequence is then fed into the RNN with GRU, and the resulting output is another sequence where each output in the sequence can be thought of as a character classification. The output can indicate “no character” well. Note that over many time sequences, it’s typical that “no character” is predicted. Additionally, when there is a predicted character, it’s typical that it persists for multiple time units. Thus, the “no character” prediction delineates the guessed characters [28].

Note that even after the prediction of a label, the text may incorrect. However, the difference between the predicted word an actual word tends to be small—in other words there’s a small edit distance. The only other major source of error is when the space between adjacent words is truncated. Thus, sometimes words are incorrectly concatenated, but this can and was worked around.

Of course, if the text detection mistakenly places a bounding box where no text exists, then recognition will necessarily fail. This is a fairly common occurrence, but it’s preferable to have more false positives than miss any true text instances as the recognition network can distinguish between the two. In cases where no text exists, the recognition network typically outputs a series of small nonsensical characters. Thus, this can be easily filtered by excluding low letter count words. Even when noise is longer, that may not pose a problem as repeatable nonsense words can still be used for aisle identification purposes. Overall, these networks were more than sufficient for environmental text extraction in this thesis.

Grocery Store Classification Network

The classification CNN used is the same one as presented in [30], and it was trained on 10 classes of synthetic grocery store products modeled within the Unreal Engine. The architecture is the same as

Google LeNet, which was chosen because of its accuracy and reduced computational cost. Typically, more accuracy can be achieved with deeper CNNs, but more convolution layers means more parameters to train. However, there's significant downsides towards blindly building deeper and deeper architectures. Besides requiring much more computations, there's also likely to be significantly more wasted computation. In other words, there will be more neurons that end up with weights resulting in them never firing. Additionally, it requires longer training and more input images. It's not always feasible to increase image sets alongside depth.

To skirt this dilemma, [2] showed that it's possible to have smaller networks with less "useless" neurons that can emulate larger networks. Thus, Google LeNet is partial realization of that goal. Instead of adding multiple convolutional layers, they decided to increase the "width" of the network. In other words, the output of layer would sometimes be a vector that's the concatenation of multiple convolutional kernels along with max pooling. In this case, the kernel widths are 1x1, 3x3, and 5x5. However, recall that the number of parameters per layer depends both on the kernel widths, the number of them, and the depth of the previous layer. If all three kernels are used in conjunction with one another, then we lead to both parameter and computational explosion.

To avoid this, before using the 3x3 and 5x5 kernels, the network first performs a 1x1 convolution. This significantly reduces the dimensionality of the problem. For example, take the 5x5 case and assume there's 3 layers. Then there needs to be 5x5x3=75 weights. However, if we first convolve with a 1x1 filter, then there only needs to be 1x1x3x5=15 weights. This subsequently reduces training time, inference time, and the number of images needed. Ultimately, despite being 22 layers, Google LeNet has only 4 million parameters as compared to the 60 million found in AlexNet, the now famous CNN that sparked the recent CNN research in the field. Additionally, Google LeNet placed first in the ILSVRC 2014 Detection Challenge [1]. Thus, the network structure was a good fit for our classification purposes.

Chapter 3

Implementation

Indoor Navigation

To test the potential for indoor navigation, videos were taken along the aisles of local grocery store on a handheld phone. The videos are taken vertically and traverse the vertical items of the current aisle location before slowly moving horizontally and repeating the process. The aisle data collection videos are approximately ten minutes long to ensure that each section of the aisle will have ample data associated with it. Additionally, the camera was held slightly in front of a reasonable shoppers distance to mimic where it would likely be when attached to an end-user. In more precise terms, the distance varies from about a third of a meter to a full meter. Additionally note, that the full lengths of all the aisles are approximately 27.13 meters.

After gathering the data for each aisle, frames were later extracted from the videos for further processing. The frame extraction happened once per second, which equates to 600 frames for a 10-minute video. Note that it’s possible more frames could be extracted per second, and thus, likely to yield in even better results.

After frame extraction, each image was used as input for both the text detection CNN and the text recognition CNN. The text detection one outputs an array of bounding boxes that is likely to hold text, and it’s within those bounding boxes that the text recognition CNN output its guesses as to what the text is in alphanumerical English. A mapping between the frame number and the associated text is saved for later processing.

All of the above steps are then repeated for another video, dubbed the “verification” video. This verification video is meant to serve as a visually impaired individual walking down an aisle. Thus, the video is taken at a constant elevation and speed. This is done in an effort to eliminate any bias that may favor misleading results. For instance, it would be inappropriate to significantly vary camera height with arbitrary discretion because it may lead to the camera capturing products that better lend themselves to text extraction. By keeping the camera mostly unvarying, this researcher bias is eliminated. Note that these videos are significantly shorter (around 3-4 minutes) and could have been made even shorter without sacrificing quality if frame extraction was performed more than once per second on images with little to no motion blur.

With the data collected, it was then processed via Python 2.7 scripts (more specifically an iPython Notebook). Persistent dictionaries were created that associated frames to words and words to frames. Before understanding how these dictionaries were used, it must first be said that each of the 5 aisles were initially partitioned into quadrants. This was done because assessing the accuracy of this coarse-grained approach first ensures that the easiest localization test can pass before delving deeper into more fine-grained localization.

This initial quadrant partition wasn't performed by simply dividing the aisle into four equally sized quadrants. Instead, the aisles were divided into quadrants based upon the items within those quadrants. For instance, in the first aisle, there was a large chips section. That entire section became one of the quadrants. On an intuitive level, it makes sense to first determine whether or not localization is possible by segmenting the aisle according to its products as sections with overlapping products would consequently have overlapping text. If this yielded poor results, an arbitrary segmentation based solely on length would likely fair far worse.

The algorithm used to determine what quadrant a current frame in the verification video belongs to works as follows. A score is assigned to each quadrant. The quadrant with the highest score is the one to be guessed. Each frame of the verification video, text is extracted from it and the extracted text is used to update the quadrant scores. The quadrant scores are updated according to the number of occurrences that word has within that quadrant. Note that the text updating each quadrant's score was not initially filtered. However, in an effort to improve accuracy, words under certain sizes were ignored and case was ignored. If value is the number of occurrences the word has within quadrant 1, then quadrant 1's score is updated as follows:

$$quadScore_1 = value \times alpha \times transProb + (1 - alpha) \times quadScore_1$$

The *alpha* variable is simply a parameter between 0 to 1 that serves to weight the new score and the old. Those familiar with how TCP uses an exponential weighted moving average to calculate round trip time will note the resemblance as that served as the inspiration. The only other distinction is the inclusion of the *transProb* variable. This helps mitigate any erroneous guesses by taking on a value of 1.0 when the current quadrant and the quadrant whose score is increasing are adjacent. However, when a non-adjacent quadrant is increasing in score, *transProb* is less than 1.0 to squash the score increase as this scenario is unlikely to occur.

Note that the above equation cannot blindly be applied in isolation because otherwise quadrant scores would never decrease. Thus, whenever one quadrant score is increasing, the others are decreased as given in the following equation:

$$quadScore_i = \ decay \times quadScore_i$$

The *decay* value is a parameter that is typically set just beneath 1.0 (such as 0.977) for best results. Also note that the exact code used is supplied in Appendix A. The results of this approach a given the Results section.

Thus far, the estimation of aisle location was coarse-grained (only into quadrants) and relied upon human annotation to determine where one section began and the other ended. However, this thesis went further to explore how well arbitrary segmentation would work along with how fine-grained the localization could reasonably be. Generating an aisle prediction for arbitrary segmentation works the same way except that frame time is assumed to correspond to location, which is reasonable as speed was constant at video time.

Last Mile Classification

To test improvements to last mile classification, the aforementioned product classification CNN, the one trained on virtual grocery store products, was used as baseline that the text extraction pipeline could augment. Note that in this thesis, last mile classification does not involve object segmentation. It is assumed that some other pipeline is able to segment images and produces ROIs for where objects are likely to be. This thesis is only concerned with the classification of products once a crop for a product already exists.

Three of the classes the product classification CNN was trained on were different brands of chips: Lays, Doritos, and Kettle. Classifying these three brands of chips was the specific task the text extraction was to assist in. Those classes present a good test opportunity for further refinement for a couple reasons. Firstly, the training images didn't involve just one chip flavor from each brand; no, instead multiple chip flavors (and thus multiple types of chip packages) were used per brand, which means that there's sometimes overlap in color. An overlap in color would potentially cause confusion for a product classification CNN but is irrelevant for text extraction. This leads into the second point that these classes are sometimes confused with one another by the product classification CNN. It has an accuracy 76.67%, and its imperfectness provides the chance for text to provide further disambiguation.

However, there are numerous ways to combine the results of the two pipelines. This thesis explored three methods but started with an approach that runs each pipeline independently before combining the results of the individual pipelines.

To be more specific, the class guesses of the product classification CNN and the text extraction pipelines were first gathered, but recall that the text extraction pipeline does not naturally output a class. Instead, it outputs a series of words. Thus a conversion was needed to transform the words to a class guess. How that was done will soon be detailed, but first assume that we have the output class guess of the text extraction pipeline. These two outputs were combined into a single 2-diminsional feature vector, which was used to train a support vector machine (SVM) classifier. Once trained, this classifier could receive the two outputs from the product classification CNN and the text extraction pipeline to determine the class a product image belonged to.

Now, the method of outputting a single class from a collection of words will be detailed. Several algorithms were considered, but ultimately, the final algorithm used works as follows. Each of the three classes (Lays, Doritos, and Kettle) have unique text on their packaging. For example, all the chip packages of the Lays and Doritos brands tend to result in words similar to or exactly matching the corresponding brand name. Additionally, flavor differences between the brands means certain words would only ever appear for certain brands. A few examples of this include: the word "classic" associated with Lays, "brand" primarily associated with Kettle, and "nacho" associated with Doritos. Thus, in order to determine which brand best matches all the words, each extracted word was compared against a collection of words for each class. To be more precise, the Levenshtein distance (better known as the edit distance) was computed between each extracted word and all the words within every brand's word set. The likelihood that the image was of a Lays, Kettle, or Doritos chips bag was dictated by whichever brand had the lowest Levenshtein distance across all its associated words. Thus, this allows for the text extraction pipeline to output a single class number. The word sets for each brand are given in Table 1 below:

Class	**Word Set**
Lays	lays, classic, ketchup, dill
Doritos	doritos, cheese, nacho
Kettle	kettle, brand

Table 1. Classes and the associated word sets edit distance was computed over.

In order to train the SVM classifier, an image base was needed for each brand. Thus, an image base of 300 training images was gathered from both self-shot and freely available online resources in which each brand's training set consisted of 100 corresponding images. Note that while it's true neural networks and even classical training methods like SVM often benefit from thousands of training images, in this case, such a large number is not necessary. That's because, as previously mentioned, the feature vector only consists of two data points, and thus a small number of images is likely to hit every possible feature vector permutation that will naturally occur.

The actual SVM code was written in Python 2.7 and used the standard sciKit SVM library. Initially, a linear SVM kernel was used—though, the default non-linear one was quickly adopted for accuracy and linear non-separability reasons.

The described combinational method thus far used the two outputs of independently running the text extraction and product classification CNN pipelines by combining them via SVM. However, this thesis also explored if other combination approaches might work better in order to answer the question of how to best combine multi-modal data. Both of the other methods involved using one pipeline as a pre-filter for the other.

The first pre-filter method used the text extraction pipeline before the product classification CNN. The information from the text extraction pipeline that influenced the product classification CNN was just the output class guess of the text extraction pipeline using the same Levenshtein method detailed above. It then used that in conjunction with the top two guesses of the product classification CNN to determine the final class guess. Overall, this method worked according to the following rules:

1. Pick the text class' 1st guess as the final output if the product classification CNN outputs the same class as either its 1st or 2nd guess.
2. Pick the text class' 1st guess as the final output if the product classification CNN doesn't guess any chips class as its 1st or 2nd guess (recall the that the product classification CNN was trained on more than just chip types).
3. Pick the 1st guess of the product class as the final output if neither of its first two outputs contains the same guess as the text extraction's output. Thus, this rule weights the outputs of the product classification CNN more so than those of the text extraction CNNs when they differ.

The second method used the product classification pipeline as a pre-filter for the text extraction pipeline. The rules of operation are similar to when the text extraction is used as a pre-filter. The exact rules are given below:

1. Pick the product class' 1st guess as the final output if the text extraction CNN outputs the same class as either its 1st or 2nd guess.
2. Pick the text class' 1st guess as the final output if the product classification CNN doesn't guess any chips class as its 1st or 2nd guess.
3. Pick the 1st guess of the text class as the final output if neither of its first two outputs contains the same guess as the product classification CNN's output. Thus, this rule weights the outputs of the text extraction pipeline more so than the product classification pipeline when they differ.

In summary, there were three methods discussed here: the SVM approach, the text extraction pipeline as a pre-filter for the product classification pipeline approach, and the product classification pipeline as a pre-filter for the text classification pipeline approach. The comparative results of all three methods are discussed in the second half of the Results section, but note now that the SVM approach faired the best.

Chapter 4

Results

Indoor Navigation

When dividing the aisles into quadrants that correspond to sections without overlapping products, the accuracy was initially poor because of unfiltered extraction noise. The text extraction pipeline had a habit of extracting several small collections of letters when no such text existed. These words tended to consist of only two to three characters, and this initially made it impossible to determine which image belonged to which quadrant because an image from say quadrant 1, would always match most closely with the longest quadrant (that being quadrant 3) due to the fact that quadrant three included a large number of small noise words that occurred in almost every frame. Without filtering such noise, accuracy was incredibly poor at 42.65% when determining quadrant location. However, upon filtering out such noise, accuracy shot up and became a notable 95.96% when determining quadrant location.

However, hand selecting quadrants without overlapping products is both tedious and requires very coarse-grained quadrants, depending on the products within the aisle. Fortunately, when ignoring products and segmenting into quadrants evenly across the aisles, the following accuracies were obtained:

	Aisle 1	**Aisle 2**	**Aisle 3**	**Aisle 4**	**Aisle 5**
Accuracy	82.23%	94.51%	89.94%	89.62%	89.92%

Table 2. Accuracy of localization with 4 evenly divided sectors.

It is also possible to increase the number of sectors while still maintaining a high accuracy. For the tested aisles, decent accuracy could be maintained even when splitting the aisles into 8 sectors without regard for the products within those sectors. The following Table 3 highlights this fact, and note that while the minimum accuracy of the five aisles was 75.13%, the maximum was aisle 2 at a surprisingly high 89.38%. The 8 sector division represents the largest number of sectors each aisle could be split before the accuracy plummeted for a couple of the aisles.

	Aisle 1	**Aisle 2**	**Aisle 3**	**Aisle 4**	**Aisle 5**
Accuracy	75.13%	89.38%	81.56%	82.31%	76.34%

Table 3. Accuracy of localization with 8 evenly divided sectors.

The incorrectly predicted frames tended to only occur when transitioning between quadrants after first determining the correct location. Additionally, transitions could be relatively quick with only 3 or 4 incorrectly guessed frames, but as the number of sectors increased, that number could rise to around 8 incorrect, consecutive frames. The average of number of frames needed to transition across all aisles when divided into 4 and 8 sectors was 4.8 frames.

The aisles measured in this thesis were all 27.13 m in length, and given that an average of 80.94% accuracy could be achieved by subdividing the aisles into 8 sections, then this thesis shows that it's possible to reliably localize within segments of 3.39 meters in length within an aisle. The aisle widths are quite small within this store, tending to be just under 1.50 meters. Thus the effective localization area is 3.39x1.50m rectangles.

While identifying where an individual is within an aisle is important, so too is maintaining accuracy when transitioning between aisles. It's possible that considering multiple aisles might confuse the localization guesses as some aisles might have similar text sets. However, when allowing for this possibility, this thesis found that such an event does not occur when five simultaneous aisles are considered if the algorithm can generate an initial correct guess. This is because the transition function prevents far off aisles from interfering. Additionally, such a localization system would naturally not need to compare against all aisles at once if aisles near the entrance have a strong initial weighting, which also makes it likely to generate the first correct guess.

These results indicate that text features can be used for first-level localization in certain localization algorithms. Typically, something like GIST might be used for clustering, which essentially limits the search space when trying to identify matching images to then determine location. While an algorithm like SURF can compare key points and then determine location, it's far too costly to perform such an operation across all database images. If text extraction and processing can first provide a small area within which the user is likely to be, then the more expensive, fine-tuned algorithms can be used on a limited dataset.

Last Mile Classification

The results of running the classification and text extraction CNNs independently before combining results will be presented first as this combination proved to be the most accurate. However, before delving into the results of the combined SVM pipeline, it's important to establish a baseline by gathering the results of how both the classification CNN performs in isolation and how the text extraction CNNs perform in isolation. The former has an accuracy of 76.67% and the latter has an accuracy of 96.67% on a test image set of 60 images. Now onto the the combined accuracy.

When using a linear kernel for training the SVM classifier, it would simply default to always using the text extraction pipeline to guess the class. Obviously, this is undesirable, and it was expected that the results would not be linearly separable but this is mentioned here for completeness.

When using the default non-linear kernel, the SVM classifier managed to obtain an accuracy of 100%. Of course, with more testing images, it's not likely to have a real-world accuracy of 100% but the key idea is that both CNN pipelines were able to contribute to improving the accuracy of classification that neither one could achieve alone. As previously mentioned, grocery stores have an average of 45,000 unique products. The results here indicate that employing a series of hierarchical classifiers might be feasible for such a situation. For instance, classifiers could first distinguish on a high level what type of product is being viewed, which could then inform which classifiers to use on the next level of classification, and finally the lowest levels of classification could be used in conjunction with a text extraction pipeline to gain the final class inference.

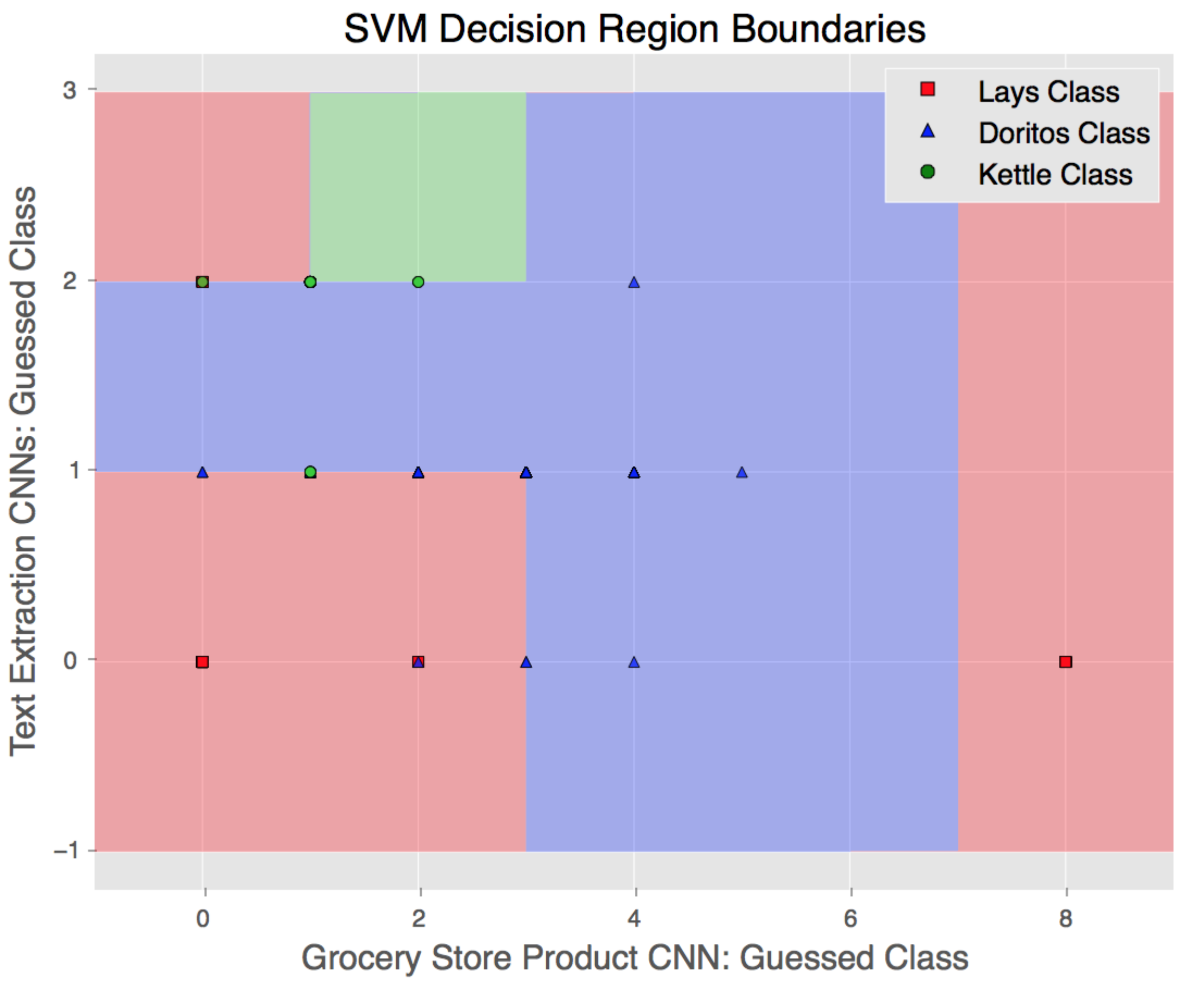


Figure 5. SVM boundaries after training on text extraction and product classification nets.

Of course, how to best combine such multimodal classifications was also explored within this thesis. By applying the CNN pipelines in a linear fashion, it's possible to use one as pre-filter towards the other as explained in the previous chapter. However, neither version of this more linear approach performed as accurately as the combined SVM approach. When using the text extraction pipeline as a pre-filter for the product classification CNN, the accuracy was 95.0%. When the product classification CNN was used a pre-filter for the text extraction pipeline, the accuracy was 98.3%.

The reason why the text as a pre-filter performed worse than the SVM approach appears to be because the classification CNN did not tend to guess the correct class as a close second whenever it could not infer it directly. Why exactly a neural net sometimes isn't even close is not a question that can be easily or satisfactorily answered as their inner workings are still somewhat mysterious, but the practical measurements provide quantitative justification towards not combining the nets in such a fashion.

Note that Levenshtein distance naturally favors words that are close in length even if they have no other correlation. This can be problematic because the Lays brand is only 4 letters long while Doritos is 7 and Kettle is 6. Since noise from the text extraction stages tends to be short in length, Lays is unnaturally and unfairly given precedence is such situations. However, this could be addressed if each brand's set of words contained dummy words like "zzzz" such that noise does not favor Lays unless it is lexicographically close. However, in this thesis, it was found that just by giving each brand's set of words a few somewhat small but relevant words, the inherent bias in Levenshtein distance was not significant enough to matter.

Chapter 5

Discussion and Future Work

The results of this thesis indicate that combining the text extraction pipeline with other concurrent processing pipelines of a visual assistance system is feasible and desirable. Overtime, text extraction is only bound to improve. While this thesis employed state-of-the-art text detection and recognition CNNs, the recognition is still imprecise and under active improvement. More precise recognition, would allow for more unique text at each location due to the text on each product's respective label. That in turn might allow for more precise localization than shown here.

This method of localization has the distinct advantage of requiring simplistic end-user setup. In order to employ such a system, all that is really required is for an individual to walk linearly down the aisle and take a video. No specific markers need to be placed down, and even if a few products change location, as long as the majority of the aisle is static, then re-gathering aisle data would not need to happen often. Additionally, this corpus of data is naturally small. All text data associated with an aisle was at most several hundred kilobytes. This data is uncompressed and redundant. Thus, it could likely be made even smaller. This means such data can easily be stored on an embedded device's RAM. However, future work might explore if it's possible to remove all prior data collection, and instead, perform SLAM using textual features as an additional input dimension. Additionally, when text extraction becomes even better, future work should explore if it's possible to retrieve enough textual data within the produce and frozen sections to perform similar localization and classification refinement like was performed in the other sections.

In this thesis, words were not processed when determining location except for slight filtering based on size and basic capitalization filtering. This Laissez-faire approach was done in part because the text extraction, as mentioned in the Background section, has its own method of gathering the closet valid word. However, the text extraction would often produce words like “nachocheese”. Clearly, this is the concatenation of the two words: “nacho” and “cheese”, and Levenshtein distance is insufficient to capture these substring similarities. Thus, it’s worthwhile to explore whether or not replacing Levenshtein distance with largest common substring would result in similar accuracies especially since Levenshtein distance is expensive to compute on words with greater than 15 characters or so.

The fact that localization was performed only within the length dimension of the aisle should be brought to attention here. While in the context of grocery store aisles, this is not a major limitation, to see the potential for text localization in other environments, future explorations should examine the possibilities of determining location in two dimensions. It may be possible to make use of the relative sizes of text bounding boxes to establish distance away from the aisle, or a stereo camera system could be employed to generate depth information, which could then obviously be used to compute aisle distance from the cameras.

While this thesis demonstrates that localization within several meters is possible, recall that only one frame was extracted per second. This was initially done for the sake of time during the data processing; however, the greater the number of frames extracted per second, the larger the data corpus for each individual section. It’s expected that a higher accuracy could have been achieved if this was done.

As for future work involving last mile classification, it would be worthwhile to explore whether or not text extraction can be used in tandem with a classification CNN to guess classes that the CNN was not trained on. Given the results of the SVM combination in this thesis, it is highly likely that such a system would work, but what’s unclear is how many classes could be distinguished with text refinement. Would the text extraction be sufficient to help classify the tens of thousands of classes within a grocery store? It’s unclear at this time.

What’s particularly interesting is the fact that classification using only text extraction and Levenshtein distance was able to achieve a baseline accuracy of 96.67% while the product classification CNN had a baseline accuracy of 76.67%. This indicates that text alone might provide sufficient classification accuracy for items with enough overlaid text, and in such a situation, a traditional classification CNN might be eschewed entirely. Of course, how text extraction for classification fairs when many products share similar text remains to be seen. In the future, it would be worthwhile to explore just how robust text extraction classification is to both classes with similar text and classes with a lot of textual noise. However, based on the results in this thesis, text can at the very least be used in conjunction with traditional classification pipelines.

Note that while the entire visual assistance system, both the location and classifications systems, were not ran in real time, there’s no reason that they could not be given the addition of networking code to transfer images from a user camera to a backend processing unit over Wi-Fi. This can be confidently asserted because the neural nets used in this thesis are all able to process multiple images per second once their corresponding frameworks are loaded onto a GPU. Thus, the expected slowest part of this pipeline is likely just the transfer of data between the end-user and the remote backend processing server. However, embedded systems are beginning to take advantage of neural nets more and more. The latest flagship phones at the time of writing are Google's Pixel 2 and Apple's iPhone X. Both of these phones contain dedicated hardware for efficient neural net computation [14]. Thus, it's fair to expect this trend to continue and the possibility of the pipelines existing entirely on one or several small devices is likely to be a soon reality.

Chapter 6

Conclusion

State-of-the-art deep neural networks for English text detection and extraction were employed for the purposes of localization and last mile classification. By using only text within a given frame, it was possible to achieve localization accuracy of 80-94%, depending on the granularity of the localization. Furthermore, this was achievable with minimal text processing, an extraction rate of only 1 frame per second, and held up when travelling across multiple aisles.

This text extraction pipeline was also used to refine the results of classification. A CNN trained to classify grocery store products was augmented with the results of text extraction using SVM to classify the combination of the two. It was found to have higher accuracy than either in isolation, and it was found that combining multimodal inputs at the end of a classification leads to the best results.

Overall the contributions of this works can be summarized as follows:

- Demonstration and evaluation of pure text pipeline for localization within text rich environments that does not make use of special text markers
- Strong indication that text extraction alone can provide sufficient information for image classification when classes have unique text
- Demonstration and evaluation of classification refinement using text extraction for visually similar classes
- Demonstration of the applicability of Levenshtein method as a sufficient filtering method against extraction noise
- Determination that combining multimodal features is most effective after gathering said features independently rather than using one as a pre-filter for the other

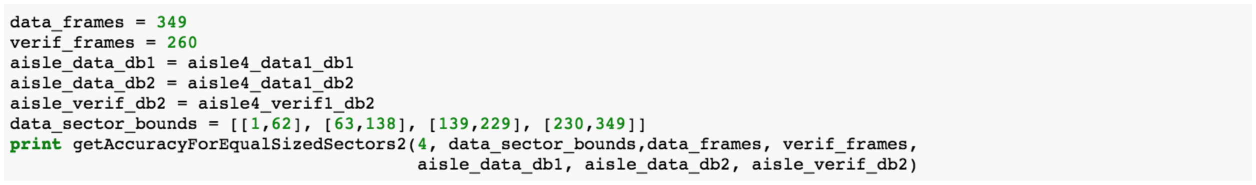
Appendix A

Aisle Localization Python Code

```python
alpha = 0.0004
decay = .977

def get_trans_prob2(first_bin_num, second_bin_num):
    if abs(first_bin_num-second_bin_num) > 1:
        return 0.7
    else:
        return 1.0


def update_sectors2(data_frame, value,
                    sectors, num_of_sectors, data_sec_bounds, frames_in_data_vid, cur_guess):
    global alpha
    trans_prob = 1.0
    data_sector_size = float(frames_in_data_vid)/num_of_sectors

    for sec_index in range(num_of_sectors):
        if cur_guess != 'tied':                                                            # Get the transition prob
            trans_prob = get_trans_prob2(int(cur_guess), sec_index)

        # Update sector bins
        if data_sec_bounds[sec_index][0] <= data_frame <= data_sec_bounds[sec_index][1]:
            sectors[sec_index] = value*alpha*trans_prob + (1-alpha)*sectors[sec_index]
        else:
            sectors[sec_index] = decay * sectors[sec_index]
```

```python
def getAccuracyForEqualSizedSectors2(num_of_sectors, data_sec_bounds, frames_in_data_vid, frames_in_verif_vid,
                                    aisle_data_db1, aisle_data_b2, aisle_verif_db2):
    cur_guess = 'tied'
    cur_ground_truth_sector = '0'
    sectors = [0] * num_of_sectors
    match = 0                                                        # Num of frames we correctly guessed
    total_frames = 0                                                 # Num of total frames seen so far
    verif_sector_size = float(frames_in_verif_vid)/num_of_sectors

    # Loop through frames in verif and determine matching sectors
    for i in range (1,frames_in_verif_vid+1):                      # Go through all verif frames
        if str(i) in aisle_verif_db2:                              # If we have words for in this verif frame
            for word in aisle_verif_db2[str(i)].words:             # Loop through the words
                if word in aisle_data_db1 and len(word) > 2:       # If the word is in the data set and not too small
                    for data_frame,value in aisle_data_db1[word].sec_to_occurances.iteritems():
                        update_sectors2(data_frame, value,
                                        sectors, num_of_sectors, data_sec_bounds, frames_in_data_vid, cur_guess)


            # IN OTHER DB
            for word in aisle1_verif_db2[str(i)].words:             # Loop through the words
                if word in aisle_data_db1 and len(word) > 2:       # If the word is in the data set and not too small
                    for data_frame,value in aisle_data_db1[word].sec_to_occurances.iteritems():
                        update_sectors2(data_frame, value,
                                        sectors, num_of_sectors, data_sec_bounds, frames_in_data_vid, cur_guess)


        # Get our current bin guess
        if max(sectors) == min(sectors):
            cur_guess = 'tied'
        else:
            cur_guess = str(sectors.index(max(sectors)))

        # Update our "match" used in accuracy calculation
        for sec_index in range(num_of_sectors):
            if verif_sector_size*(sec_index) <= i <= verif_sector_size*(sec_index+1):
                cur_ground_truth_sector = str(sec_index)
                if cur_guess == cur_ground_truth_sector:
                    match += 1
        total_frames += 1.0

    return (match/total_frames)*100
```

The below image shows how to call the localization code in the case of dividing aisle 4 into equal sized sectors.

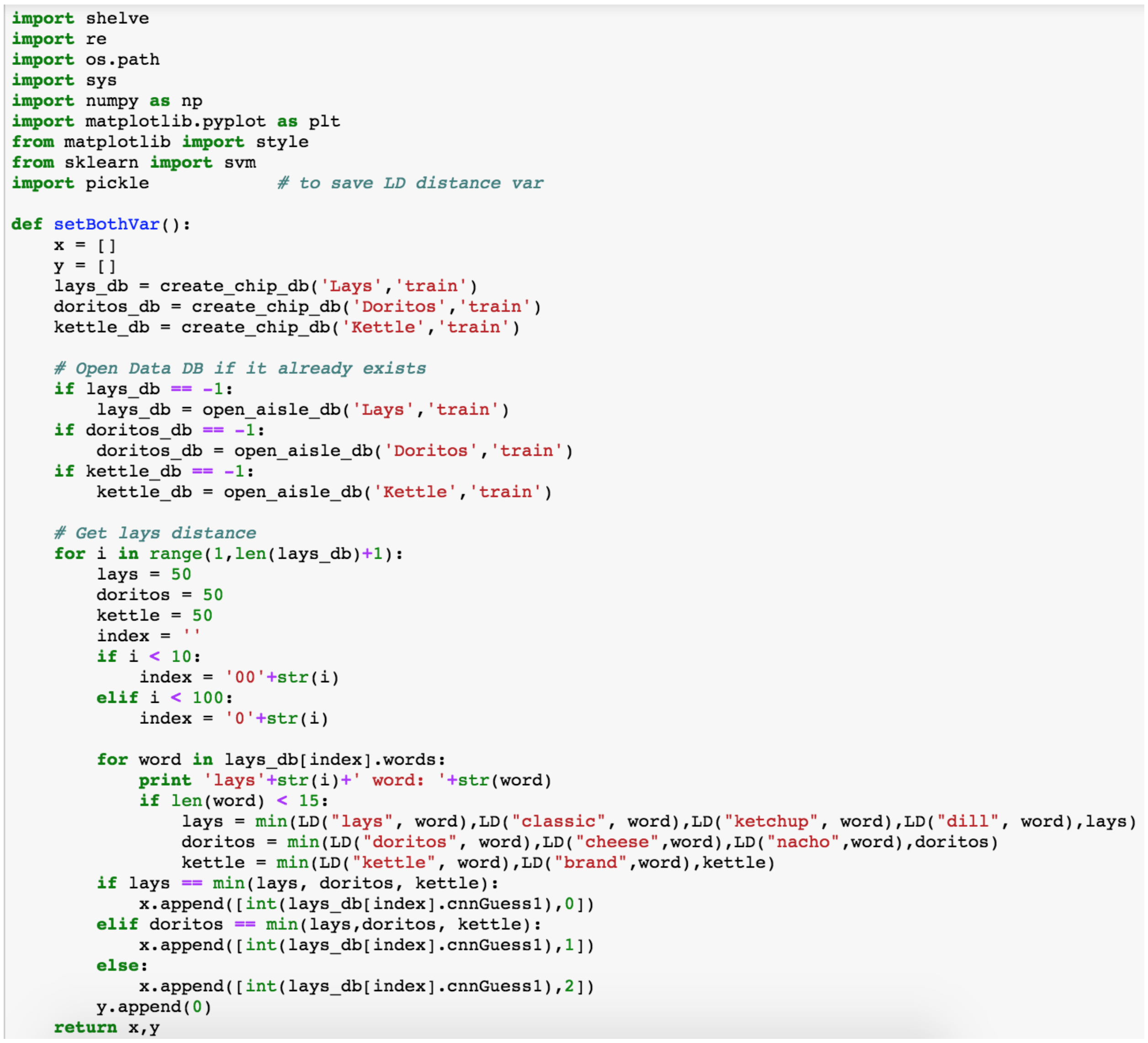


```
data_frames = 349
verif_frames = 260
aisle_data_db1 = aisle4_data1_db1
aisle_data_db2 = aisle4_data1_db2
aisle_verif_db2 = aisle4_verif1_db2
data_sector_bounds = [[1,62], [63,138], [139,229], [230,349]]
print getAccuracyForEqualSizedSectors2(4, data_sector_bounds,data_frames, verif_frames,
                                       aisle_data_db1, aisle_data_db2, aisle_verif_db2)
```

Appendix B

SVM Python Code

The *setBothVar* function below returns two variables. The first is an array of two dimensional features of the form [CNN product class guess, text extraction product class guess], and the second is a parallel array of the same sizing giving the ground truth for each element’s class. Note that the below code gets just the Lays feature vectors (getting the Doritos and Kettle feature vectors works similarly).



```python
import shelve
import re
import os.path
import sys
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import style
from sklearn import svm
import pickle            # to save LD distance var

def setBothVar():
    x = []
    y = []
    lays_db = create_chip_db('Lays','train')
    doritos_db = create_chip_db('Doritos','train')
    kettle_db = create_chip_db('Kettle','train')

    # Open Data DB if it already exists
    if lays_db == -1:
        lays_db = open_aisle_db('Lays','train')
    if doritos_db == -1:
        doritos_db = open_aisle_db('Doritos','train')
    if kettle_db == -1:
        kettle_db = open_aisle_db('Kettle','train')

    # Get lays distance
    for i in range(1,len(lays_db)+1):
        lays = 50
        doritos = 50
        kettle = 50
        index = ''
        if i < 10:
            index = '00'+str(i)
        elif i < 100:
            index = '0'+str(i)

        for word in lays_db[index].words:
            print 'lays'+str(i)+' word: '+str(word)
            if len(word) < 15:
                lays = min(LD("lays", word),LD("classic", word),LD("ketchup", word),LD("dill", word),lays)
                doritos = min(LD("doritos", word),LD("cheese",word),LD("nacho",word),doritos)
                kettle = min(LD("kettle", word),LD("brand",word),kettle)
        if lays == min(lays, doritos, kettle):
            x.append([int(lays_db[index].cnnGuess1),0])
        elif doritos == min(lays,doritos, kettle):
            x.append([int(lays_db[index].cnnGuess1),1])
        else:
            x.append([int(lays_db[index].cnnGuess1),2])
        y.append(0)
    return x,y
```

The below code shows how to call the prior code, train via SVM, and use the newly trained SVM object.

```
# Inputs to the SVM training
x,y = setBothVar()
X = np.array(x)

# Train with default SVM parameters
clf.fit(X,y)

# Print the predicted class using the trained SVM
print(clf.predict([1,2]))
```

BIBLIOGRAPHY

[1] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, et al. Going deeper with convolutions. Cvpr, 2015.

[2] Sanjeev Arora, Aditya Bhaskara, Rong Ge, and Tengyu Ma. Provable bounds for learning some deep representations. In *International Conference on Machine Learning*, pages 584– 592, 2014.

[3] John Nicholson, Vladimir Kulyukin, and Daniel Coster. Shoptalk: independent blind shopping through verbal route directions and barcode scans. *The Open Rehabilitation Journal*, 2(1):11–23, 2009.

[4] Xiaoqing Liu and Jagath Samarabandu. An edge-based text region extraction algorithm for indoor mobile robot navigation. In *Mechatronics and Automation, 2005 IEEE International Conference*, volume 2, pages 701–706. IEEE, 2005.

[5] Aura Ganz, Siddhesh Rajan Gandhi, James Schafer, Tushar Singh, Elaine Puleo, Gary Mullett, and Carole Wilson. Percept: Indoor navigation for the blind and visually impaired. In *Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE*, pages 856–859. IEEE, 2011.

[6] National Federation of the Blind Fact Sheet. https://nfb.org/fact-sheet-blindness-and-low-vision.

[7] Alessandro Mulloni, Daniel Wagner, Istvan Barakonyi, and Dieter Schmalstieg. Indoor posi- tioning and navigation with camera phones. *IEEE Pervasive Computing*, 8(2), 2009.

[8] Luis A Guerrero, Francisco Vasquez, and Sergio F Ochoa. An indoor navigation system for the visually impaired. *Sensors*, 12(6):8236–8258, 2012.

[9] Muhammad Sami, Yasar Ayaz, Mohsin Jamil, Syed Omer Gilani, and Muhammad Naveed. Text detection and recognition for semantic mapping in indoor navigation. In *IT Convergence and Security (ICITCS), 2015 5th International Conference on*, pages 1–4. IEEE, 2015.

[10] Xiao-Xiao Niu and Ching Y Suen. A novel hybrid cnn–svm classifier for recognizing handwritten digits. *Pattern Recognition*, 45(4):1318–1325, 2012.

[11] Gang Wang, Derek Hoiem, and David Forsyth. Building text features for object image classification. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1367–1374. IEEE, 2009.

[12] Hyeonseob Nam and Bohyung Han. Learning multi-domain convolutional neural networks for visual tracking. In *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*, pages 4293–4302. IEEE, 2016.

[13] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2014 IEEE Conference on*, pages 512–519. IEEE, 2014.

[14] David Nield. What do the ai chips in new smartphones actually do?, Dec 2017. https://fieldguide.gizmodo.com/what-do-the-ai-chips-in-new-smartphones-actually-do-1820913665.

[15] Abdelsalam Helal, Steven Edwin Moore, and Balaji Ramachandran. Drishti: An integrated navigation system for visually impaired and disabled. In *Wearable Computers, 2001. Proceedings. Fifth International Symposium on*, pages 149–156. IEEE, 2001.

[16] Lisa Ran, Sumi Helal, and Steve Moore. Drishti: an integrated indoor/outdoor blind navigation system and service. In *Pervasive Computing and Communications, 2004. PerCom 2004. Proceedings of the Second IEEE Annual Conference on*, pages 23–30. IEEE, 2004.

[17] Hideo Makino, Ikuo Ishii, and Makoto Nakashizuka. Development of navigation system for the blind using gps and mobile phone combination. In *Engineering in Medicine and Biology Society, 1996. Bridging Disciplines for Biomedicine. Proceedings of the 18th Annual International Conference of the IEEE*, volume 2, pages 506–507. IEEE, 1996.

[18] Jack M Loomis, James R Marston, Reginald G Golledge, and Roberta L Klatzky. Personal guidance system for people with visual impairment: A comparison of spatial displays for route guidance. *Journal of visual impairment & blindness*, 99(4):219, 2005.

[19] Thomas Strothotte, Helen Petrie, Valerie Johnson, and Lars Reichert. Mobic: user needs and preliminary design for a mobility aid for blind and elderly. In *The European Context for Assistive Technology: Proceedings of the 2nd TIDE Congress, 26-28 April 1995, Paris*, volume 1, page 348. IOS Press, 1995.

[20] Vivek Pradeep, Gerard Medioni, and James Weiland. Robot vision for the visually impaired. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*, pages 15–22. IEEE, 2010.

[21] Ehud Mendelson. Indoor and outdoor mapping and navigation utilizing rf bluetooth beacons, December 1 2015. US Patent 9,204,257.

[22] Ramsey Faragher and Robert Harle. Location fingerprinting with bluetooth low energy beacons. *IEEE journal on Selected Areas in Communications*, 33(11):2418–2428, 2015.

[23] Cliff Randell and Henk Muller. Context awareness by analysing accelerometer data. In *Wearable Computers, The Fourth International Symposium on*, pages 175–176. IEEE, 2000.

[24] Andrew R Golding and Neal Lesh. Indoor navigation using a diverse set of cheap, wearable sensors. In *Wearable Computers, 1999. Digest of Papers. The Third International Symposium on*, pages 29–36. IEEE, 1999.

[25] Jorge Fuentes-Pacheco, José Ruiz-Ascencio, and Juan Manuel Rendó n-Mancha. Visual simultaneous localization and mapping: a survey. *Artificial Intelligence Review*, 43(1):55–81, 2015.

[26] George E Dahl, Tara N Sainath, and Geoffrey E Hinton. Improving deep neural networks for lvcsr using rectified linear units and dropout. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8609–8613. IEEE, 2013.

[27] Xinyu Zhou, Cong Yao, He Wen, Yuzhi Wang, Shuchang Zhou, Weiran He, and Jiajun Liang. East: an efficient and accurate scene text detector. *arXiv preprint arXiv:1704.03155*, 2017.

[28] Xiao Yang, Dafang He, Wenyi Huang, Zihan Zhou, Alex Ororbia, Dan Kifer, and C Lee Giles. Smart library: Identifying books in a library using richly supervised deep scene text reading. *arXiv preprint arXiv:1611.07385,* 2016.

[29] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Reading text in the wild with convolutional neural networks. *International Journal of Computer Vision*, 116(1):1–20, 2016.

[30] Jinhang Choi, Kevin Irick, Justin Hardin, and Weichao Qiu. Stochastic functional verifica- tion of dnn design through progressive virtual dataset generation. In *Circuits and Systems (ISCAS), 2018 IEEE International Symposium on*. IEEE, 2018.