Leveraging Computer Vision Priciples for Image-Based Shoe Recommendation

Amrit Saxena
Computer Science
Stanford University
amrit@cs.stanford.edu

Vignesh Venkataraman
Computer Science
Stanford University
viggy@stanford.edu

Neal Khosla
Computer Science
Stanford University
nealk@cs.stanford.edu

***Abstract*—We have developed an image-based shoe recommendation system to help modernize the retail industry. This system assesses a dataset of over 10,000 men's shoes and extracts the shape, color, and texture associated with each of these shoes. Using a combination of contour detection, scale-invariant feature transform, color-based clustering, and several other algorithmic approaches, it is able to extract shape, color, and texture feature descriptors with over 80% accuracy. It then recommends shoes that are similar to a user's input image on the basis of these features by leveraging support vector machines, k-means, k-nearest neighbors, and filtering techniques. On the whole, our algorithm takes seconds to compute and has a recommendation accuracy of over 88% for each of shape, color, and texture across the top 10 recommendations for each image.**

I. Introduction

A. Background

The retail experience has changed very little over the course of the past couple of decades. There have certainly been major developments in terms of e-commerce and novel distribution channels. However, there has been little consumer-facing change in the brick-and-mortar retail industry. In response to the relatively slow rate of technological innovation in the retail industry, we set out to utilize computer vision techniques to modernize the industry. We identified real-time recommendation systems for retail goods as an area with significant opportunities for advancement. While e-commerce platforms utilize techniques like collaborative filtering to power recommendations for users, recommendations in brick-and-mortar stores are still very low-tech and are generally driven by word-of-mouth from friends and family and pitches by salesmen. The problem lies in that it is difficult to build a useful user profile vector and recommend things to people in real time as they shop physically. Even if it were possible to track this data, it is fairly difficult to compute recommendations for items in the brick-and-mortar setting due to poorly-defined taxonomies and a general lack of data properly and consistently characterizing products in retail database systems.

We sought to develop a mechanism that could provide extremely accurate and efficient recommendations without having to depend on a large number of data points and artificial intelligence seemed like the perfect tool to solve this problem. As a result, we decided to leverage computer vision and machine learning to build an algorithm that could recommend retail goods on the basis of image similarity as such a process would require no data outside of the images of the goods.

Our belief is that successfully computing recommendations on the basis of image similarity would be a stepping-stone towards building more robust recommendation systems that could significantly improve the way in which people shop as it would allow for more customized, flexible, and accurate guidance in identifying desired goods. As this provides significant value to commercial entities and consumers alike, this is a problem that many people are trying to tackle in the real world right now, albeit with varying levels of success. Acknowledging that the problem that we were seeking to solve was a difficult one, we decided to initially attempt to solve a simpler subproblem. We proceeded by assessing shoes, a retail product that is not overly variable or complex, in order to maximize our chances of substantive results. Beyond the relative lack of complexity in shoe styles, footwear is also a significant industry within the retail space and represents transactions of over $54 billion per year [8]. To classify the shoes, we have chosen to extract features representing the shape, texture, and color of the shoes as we have identified them as key descriptors of shoes. The extraction and assessment of these features constitutes the vast majority of our work.

We had started working on solving this problem as a part of the CS 229 course, and although certain aspects of our project had been successful, the project's results left a lot to be desired on the whole. Furthermore, we felt like the processes that we employed lacked robustness and scalability. After conferring with the CS231a course staff, we decided to reassess our algorithmic approaches and tackle the problem from scratch. A criticism we received from the staff was that there was very little true machine learning in our project, aside from clustering and computation of simple Euclidean distances. As a result, it was difficult to gauge the effectiveness of our similarity algorithm.

In response to this, we are taking a much more machine learning and computer vision-intensive approach to tackle the problem. We are reworking each of our features to be more representative of the visual characteristics they represent, and we are applying numerous computer vision and machine learning techniques to maximize the effectiveness of these improved features in delivering relevant and useful results to the user. Additionally, as per the suggestions of the staff, we have turned this into a partially supervised learning problem by mechanically assigning labels to a sample from our dataset. In addition to creating a far more robust algorithm, we have

also made significant progress towards making our algorithm more flexible to allow for the analysis of more noisy images.

B. Dataset

For our initial analysis, we wanted to compile a dataset of images with consistent lighting and positioning so that we could focus on developing a robust algorithm without having to worry about making sense of the noise in dirty images. As a result, we created a dataset by scraping the data for all of the 10,786 different men's shoes that were listed on Zappos.com at the time of scraping. We chose to use images from Zappos for our project as the site contains a fairly exhaustive set of shoes among its product listings and has images with exceptionally consistent lighting, positioning, and orientation. For each shoe, we scraped a side-view image of a leftshoe facing left. Along with each of these images, we scraped data about each of the products including the targeted gender, the name of the shoe, the brand of the shoe, the shoe price, color, and Zappos' provided categorical taxonomy of the shoe. While Zappos displays seven views per shoe, we chose to only assess one view in our work as the flexibility and performance of our algorithm would be severely impacted if it depended on that many views per shoe. Furthermore, such a dataset would require over 200 GB of memory and would make it far more difficult for us to carry out our work with diminishing returns.

For the second layer of our analysis, we aggregated a dataset by taking pictures of shoes ourselves. These images were far noisier than the Zappos' dataset that we leveraged for the first part of our analysis as there was much more variability in positioning, scale, rotation, illumination, and the background of these images. As such, this dataset allowed us to assess the flexibility of our algorithmic approaches.

II. Previous Work

A. Review of Previous Work

Given the inefficiencies in the retail space, this is a problem that many people are trying to tackle in the real world right now. A clothing recommendation engine, Kaleidoscope, initially tried to recommend similar clothing items to its users by utilizing machine learning-based classification methods but ended up settling for manual classifications due to an inability to properly classify the stylistic elements of retail goods that they were assessing [5]. Another company that was actually fairly successful at classifying similar retail goods on the basis of images was Modista, a computer vision-based retail good recommendation service [7]. We spoke with one of the co-founders of Modista about how to approach out project and in our talks with him, he told us that his team had worked on classifying retail goods by shape and color. He also advised us to not tackle texture-based classification due to its relative difficulty and the effectiveness of shape and color-based classification.

There has also been a fair amount of non-commercial, academic work in this space. While progress has been made to solve individual subproblems of this problem, none of the work has been particularly successful at creating a very accurate and robust image-based retail good recommendation engine.

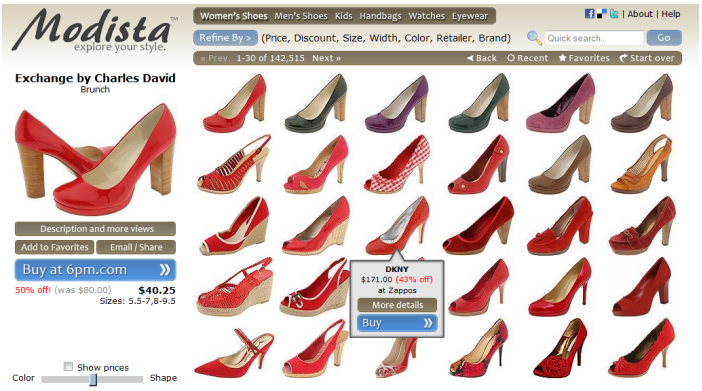

Fig. 1. A set of shoe recommendations generated by Modista.

In early 2013, Willimon et al. used a multilayer approach to classify articles of clothing in a pile of laundry by leveraging a combination of color, texture, shape, and edge information from 2D and 3D in both a local and a global perspective [12]. While they improved upon the previous baselines by about 15%, they still only realized an accuracy rate of about 28%. Nonetheless, specific aspects of the work were applicable to the problems that we were seeking to solve. One aspect of their work that inspired us was that they used SIFT descriptors to gather 2D local texture data and we utilized a similar strategy in our texture feature detection.

We also drew inspiration from Borras et al. through a UAB Bellaterra paper entitled "High-Level Clothes Description Based on Colour-Texture and Structural Features" [11]. In this work, the authors seek to interpret five cloth combinations in a graphical structure to try to ascertain what someone is wearing from an image. While the work that the researchers did is interesting, they only achieved a 64% overall accuracy due to occlusions, variable positioning, and an imperfect extraction and classification of clothing features. As such, the work reinforced the difficulty of the problem that we sought to solve to us.

Another work that directly contributed to the development of our algorithm, was a paper entitled "Perception for the Manipulation of Socks" [13]. In this paper, the researchers were able to classify whether a sock was properly oriented or inside-out with over a 90% accuracy by using local binary patterns for texture recognition and detection. We eventually leveraged a combination of local binary patterns and spatial pyramid matching in the development of our texture feature extraction mechanism.

B. Significance of Our Work

The successful completion of this project has many significant academic and commercial implications. From a commercial standpoint, this work can fundamentally change modern commerce by making brick-and-mortar far more efficient and personalized. While there have been a number of developments in e-commerce to make it a more personal experience in recent years, brick-and-mortar commerce can significantly benefit from image-based recommendation systems. Deploying such technology in a brick-and-mortar setting can make it easier for consumers to find goods that they would like to purchase,

removing an operational inefficiency from the industry. Furthermore, upon aggregating information about consumers as they use an image-based recommendation service in a physical setting, retailers would be able to build systems with more relevant and real-time targeted advertising to consumers as they shop. This would be a significant value add to consumers and retailers alike.

From a theoretical perspective, this work can help optimize feature extraction from images of clothing items. A robust mechanism to classify the shape, color, and texture of shoes could be directly applied to other types of clothing items. From being leveraged in laundry machine-unloading robots to identifying people in surveillance videos, this technological development could have a wide range of implications and applications. Moreover, some of the computer vision techniques used and developed in our approach could be parlayed to make progress in other fields as well. From cancer cell identification in the human body to detecting the presence of mines from satellite images, aspects of this work can be leveraged in solving a variety of significant problems.

III. Technical Part

A. Summary

In our efforts to create a robust recommendation system, we began by analyzing and identifying what features might define a shoe and its style. While we had initially hoped to use our metadata (specifically the Zappos' taxonomy of each shoe) as a metric for the success of our classification of style, we quickly concluded that Zappos provided extremely poor taxonomies. Nearly 50% of shoes in the dataset were classified in the "Sneakers & Athletic Shoes" category. Because this provided taxonomy lacked any real granularity, we pivoted to approach this problem as a partially unsupervised learning problem (we labeled a subset of our images to build a support vector machine-based classification system as a part of our final algorithm). We then shifted our focus to determining a feature set that would enable us to accurately group similar shoes. Through a drawn out brainstorming process that involved both numerous discussions and cursory explorations of different approaches, our group concluded that the three main features that defined shoes were shape, color, and texture.

1) Shape - We identified that shoes with similar shapes tend to belong to the same style categories and therefore concluded that the shape of a shoe would give us significant fundamental insight into the style of a shoe.
2) Texture - We decided that ascertaining the texture of a shoe would make our recommendations far more accurate as shoes with similar shapes and colors can often be of vastly different styles. For example, a low-cut brown running shoe and a low-cut brown casual-wear skate shoe will have extremely similar shapes and colors, but the running shoe would have a mesh-like texture, while the skate shoe would have a much smoother texture.
3) Color - We also noted that from a visual perspective, the color of a shoe is one of the first things that the brain registers, and thus, that color might provide a good feature to base recommendations on.

B. Shape

To ascertain shoe shapes, we first tried a fairly rudimentary clustering approach to determine cluster centers corresponding to different styles of shoes. By using the category taxonomies from Zappos' website, we computed prototypes for cluster centers for each one of the major categories. To do this, we converted each image corresponding to a given category to a grayscale array and computed the average grayscale array corresponding to each category. Once we had these initialized cluster centers, we ran k-means on our entire dataset of images. This technique resulted in a style classification accuracy of about 55% on a randomly-selected, statistically significant sample. As previously mentioned, we realized that one of the primary reasons for this was that Zappos' taxonomies were highly generalized and thus did not do a good job of differentiating between shoes with vastly different shapes and styles and causing this method to significantly underfit our dataset. Thus, for our next approach, we went through all of our images and found nine images that we determined were good representations of most of the distinct shapes and styles of men's shoes (i.e., low-tops, sneakers, high-tops, boots, slippers, clogs, boat shoes, sandals, and flip flops). We then clustered on the basis of these initialized cluster centers and achieved a shape classification accuracy of about 75%. Subsequently, since the only relevant factor for this step of our algorithm was shape, we decided to mask all of the images as illustrated in Figure 2. To compute these masks, we utilized OpenCV's thresholding capabilities [2]. Specifically, we applied the inverse binary thresholding function on grayscale images in order to turn the background of the image black while whiting out the contents of the shoe itself. The inverse binary thresholding function is defined by:

$$\texttt{dst(x,y)} = \begin{cases} 0, & \text{if } \texttt{src(x,y)} > \texttt{thresh} \\ \texttt{maxVal}, & \text{otherwise} \end{cases} \quad (1)$$

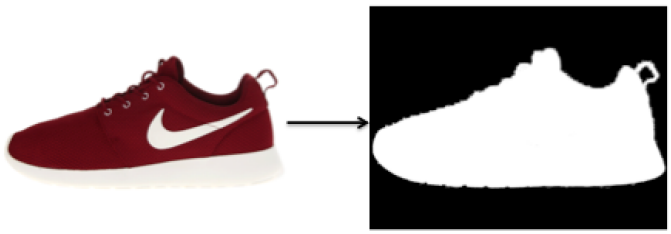

Fig. 2. A mask computed for one of the shoes in our dataset.

Thus, by passing in 254 for our value of `thresh` and 255 for our `maxVal`, the background, which is entirely whitespace, is transformed to black. Similarly, any space that isn't pure white (i.e., the shoe itself) is transformed to white. Now, instead of running k-means clustering on the raw images themselves, we decided to utilize the grayscale array representations of the masks of the images that we had selected as cluster centers before and simply assigned each image in our dataset to the cluster center that had the smallest matrix difference from it as follows:

$$\arg\min_{cluster} \text{non\_zero\_count(cluster - image)} \quad (2)$$

This approach worked well, and we achieved a shape-based style classification accuracy of about 80%.

Although this was an effective approach, it lacked any real machine learning or computer vision basis; it was essentially a comparison of two image masks while counting the number of different pixels. Thus, our next approach began with turning this unlabeled problem into a partially-labeled one. Specifically, we went through and labeled 1000 shoes based on their shape.

We then sought to discover an accurate feature vector that would allow us to classify any shoe into its proper shape class, given just the image of the shoe. A logical starting place for this was the shoe's maximal external contour. This was manifested as the trace of the shoe's outermost image boundary. We computed the external contours of each of the shoes in our dataset by once again employing the inverse binary thresholding method described in Equation 1. We then extracted the contour of maximal size from this list of contours as the largest contour in the list would correspond to a roughly traced outline of the shoe.

Given the contour, we then had to decide what properties of the contour we could potentially use as a feature vector. Some options that we took time to test out included the moments of the contour (these include the centroid and other pixel-weighted averages across the image), the contour's area, the contour's length, a polynomial approximation of the contour using the Ramer-Douglas-Peucker algorithm, and the bounding box of the contour. The eventual goal was to leverage one or more of these properties into a support vector machine (SVM) in order to train and classify shoe shapes. Having already labeled 1000 images, we split them up into a training and testing dataset and exhaustively experimented with each of the features in isolation. Most yielded errors that were over 20%, which would make the SVM classifier even worse in terms of accuracy than our previous approach. Not content with this, we proceeded to test out a few other strategies that all failed for one reason or another. We actually tried feeding the raw masked image (i.e. background is black, shoe area is white) into an SVM with our provided labels and achieved nearly 60% error. Logically, this failure makes sense because of the extraneous data that gets packaged in with the image mask.

Fig. 3. The bounding box computed for one of the shoes in our dataset.

However, selecting the bounding box as our feature vector yielded excellent results. A bounding box is defined by its x-origin, y-origin, width, height, and orientation. However, for the purposes of this classification problem, we are guaranteed to have a perfectly horizontal orientation of the bounding box at all times, as the input images are all photographed in that manner. Thus, the feature vector that we used in the SVM consisted of the x, y, width, and height parameters. We tried a variety of different kernels, and ended up settling upon a radial basis function kernel for the purposes of our classification as described in Equation 3. From this, we then created a classification model that achieved a 94.6% classification accuracy when we trained on 90% of the data and tested on the remaining 10%.

$$K(x_i, x_j) = e^{-\gamma||x_i - x_j||^2}, \gamma > 0 \tag{3}$$

Having developed two very different but quite effective and complementary approaches to classifying shoe shape (i.e., the grayscale-mask difference comparison and the SVM-based approach), we decided to leverage a combination of these approaches in our final algorithm.

C. Texture

To compute textural similarities between shoes, we first employed a texture descriptor called Local Binary Pattern (LBP). Figure 4 depicts the basic concept behind the LBP algorithm. LBP features are represented by first calculating the difference in grayscale value between each pixel in the image and its 8 direct neighbors. Then, the signs of those calculated differences are noted. Since there are 8 neighbors, each with the possibility of having a positive or negative difference, there are 256 possible LBPs. Because we applied the VLFeat library's LBP implementation to our dataset, we adhered to their standard of reducing those 256 possible LBPs to 58 quantized LBP possibilities, which depend on the transitions between negative and positive differences when the neighbors are scanned in anti-clockwise order. Reducing the number of quantized LBP possibilities to 58 allows the computation of a more reliable histogram that is ultimately the final output of this algorithm [6].

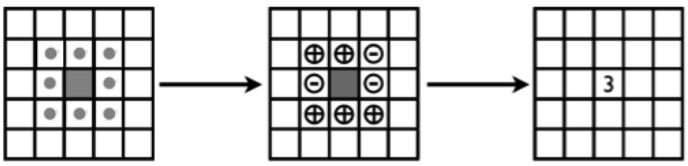

Fig. 4. A demonstration of the basic algorithm behind Local Binary Patterns (LBPs) [13]. LBPs are calculated by first computing the difference between each pixel and its neighbors. Then noting the resulting pattern of positive and negative differences, the sequence of binary values representing positive and negative changes is mapped to a quantized value in the histogram that is outputted.

After calculating the LBP features for each of our images, we compared their texture-based similarity by utilizing OpenCV's compareHist function to calculate the correlation between two histograms [9]. This function denotes the distance between any two histograms H_1 and H_2 as:

$$d(H_1, H_2) = \frac{\sum_i (H_1(i) - \bar{H_1})(H_2(i) - \bar{H_2})}{\sqrt{\sum_i (H_1(i) - \bar{H_1})^2 \sum_i (H_2(i) - \bar{H_2})^2}} \tag{4}$$

where $\bar{H_k}$ is the mean of the values in H_k or $\frac{1}{n}\sum_i H_k(i)$.

After implementing this basic version of LBPs and applying it to our dataset, it became evident that our method of comparing textures was not properly functioning. First, we observed that the comparison of texture as an additional factor in our similarity engine (along with color and shape) was having no effect on the outputted set of recommended shoes, revealing that our engine was not weighting texture correctly. However, in trying to determine how to reweight the algorithm, we discovered that almost all texture histograms for all images were being categorized as 99% or more identical. This led us to believe that our texture algorithm was not giving us any insight into differences in texture between images and thus needed to be refitted. Our first attempt to solve this problem was to switch from comparing the histograms using the compareHist function to measuring the Kullback-Leibler Divergence (KL Divergence) between the two distributions. However, our issues persisted, prompting us to reconsider our original LBP feature determinations. In order to investigate this, we temporarily edited the recommendation algorithm to recommend shoes solely on the basis of texture and were achieving less than 25% accuracy. After analyzing possible causes extensively, we concluded that our algorithm was not weighting the more granular textures in the shoes heavily enough and was just matching the significant white region of the images, which did not contain the shoe as similar. To address this, we decided to implement a modified version of Spatial Pyramid Matching to get more granular features[14]. Spatial Pyramid Matching works by creating histograms of features for progressively smaller sized subdivisions of each image and weighting the more granular histograms more when comparing two images histogram features. Figure 5 depicts the idea behind Spatial Pyramid Matching for a trivial example with 3 quantized features.

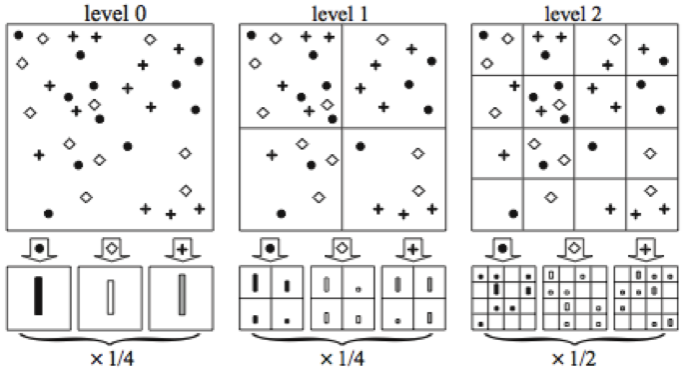

Fig. 5. A depiction of the Pyramid Matching algorithm for a trivial example with 3 quantized features. The image is divided into 3 different levels of resolution for which feature histograms are computed and weighted more heavily for higher resolution subdivisions [14].

For the purposes of our texture-matching algorithm, we leveraged this concept by creating LBP histograms for progressively smaller subdivisions of our images for use in calculating the similarity between two images. We utilized VLFeat's functionality that allows the user to specify different resolution divisions of the image, enabling us to use the same LBP library by calculating extra histograms for finer resolutions than exclusively the entire image. Through iterative consideration of different grid sizes, we were able to settle on dividing our images into 1, 4, and 12 subregions and improved our isolated texture recommendation accuracy to approximately 65%.

While this approach showed promising improvements, it was still far from perfect. As such, we read through many research papers for good texture feature extraction mechanisms, but could find nothing that was generalized or accurate enough to deal with the wide variety of textural patterns that a typical shoe contains. After considering a variety of different approaches, we adopted a bag-of-words model, leveraging scale-invariant feature transform (SIFT) to assess the textures of the shoes in our dataset as per the advice of Professor Savarese.

The bag-of-words model draws its origins in natural language processing applications [1]. As formalized in a computer vision context, the BoW model can be applied to image classification by treating image features as 'words.' Thus, the 'bag' of visual words becomes a simple histogram based on some generated vocabulary of visual words. The generation of a BoW model generally comes down to three key steps: feature detection, feature description, and codebook generation. In general, the first two steps are handled by your choice of feature detector/descriptor, of which there are a vast multitude of choices from which to pick. Codebook generation is left to the user of the BoW model. The goal of this step is to build up some vocabulary of visual words, with which each computed feature descriptor will be compared. The codebook is analogous to the dictionary produced by an NLP implementation of the BoW model.

The first tough design choice we faced was which detector/descriptor algorithm to use. We initially chose SIFT because of its widespread use and the large number of research works that we read that leveraged it in some form. Additionally, there was robust support for the SIFT algorithm OpenCV's Python manifestation. That said, in retrospect, we could have used SURF or ORB or any number of other keypoint detector/descriptor algorithms, but didn't find the need to due to the great success of the SIFT approach.

Fig. 6. The sift features extracted for one of the shoes in our dataset.

To properly use the BoW model with some type of predictive model, we needed our learning problem to be at least partially supervised. As such, we took the first 500 shoes in our dataset and labeled them based on texture. We realized that the biggest weaknesses of our initial texture classification approaches were an illumination and color-based tendency to spit out leather dress shoes when given a dark black running shoe. As such, we sought to build an engine that could properly classify shoes on the basis of the type of texture that they had and sought to classify shoes into the texture categories of leather, mesh/athletic, and other (where other is a catch-all class including canvas, suede, plastic, etc). Having labeled the first 500 shoes with these classes, we then ran the SIFT algorithm on our dataset to detect the keypoints and texture descriptors in each shoe. Once we had computed these, we proceeded with the canonical BoW codebook generation

approach: we ran a k-means clustering algorithm on all of the keypoints and computed 1000 different cluster centers. These 1000 cluster centers corresponded to the "words" in our "bag," and we vector-quantized each keypoint in each image to the closest cluster center as described in Equation 5.

$$\text{kp[image]} \leftarrow \arg \min_{cluster} ||\text{cluster - sift[image]}||_2 \quad (5)$$

From here, we computed per-image histograms of the keypoint centroid distribution. To ensure correctness, we visually compared histograms between shoes of similar and different texture types (i.e. leather to athletic vs. leather to leather) and verified that shoes within the same class had similar histograms, while shoes of different texture classes had noticeable differences in their histograms. Finally, we once again leveraged a RBF kernel-based support vector machine as described in Equation 3 to classify the histograms generated by this approach. This generated classification model had a roughly 80% overall accuracy when trained on 90% of our labeled data and tested upon the remaining 10% of our labeled data. This represented a marked improvement upon all of the approaches we had used before and all of the approaches that we had encountered in the literature that we had read in preparation for this project. As such, if we ran an exclusively texture-based similarity engine, it would be 80% accurate given this three-label setup.

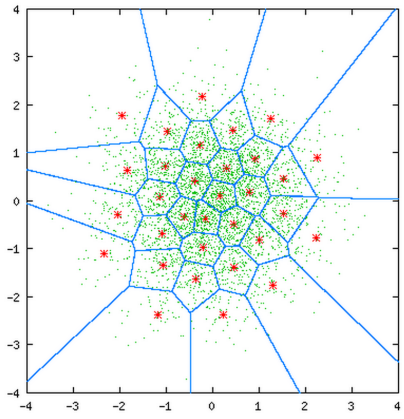

Fig. 7. An illustration of vector quantization to compress data.

One of the weaknesses of this approach is its limited scope of texture classification; it can only distinguish between three very simple classes of textures. Looking forward to future development, we hope to experiment with some texton-based features and a superpixel-based approach. Another weakness of our texture algorithm is its reliance on the bag-of-words model, which is notorious for ignoring the spatial relationships among visual words. A suggested solution to this problem is to implement spatial pyramid matching within the histogram generation step, which will add a spatial hierarchy through the use of coarse and fine grain subregions. That said, we are extremely pleased with the improvements of our model as they currently stand.

D. Color

Our first attempt to create relevant color features for each shoe involved computing color histograms over the entire image for each pixel's RGB values. We then compared the histograms of all of the shoes using a correlation metric in the OpenCV compareHist function library [9]. However, this yielded mixed results: although we were able to correctly match the primary color of the shoe, the histogram comparison was weighted too heavily by the frequency with which each color appeared. The primary color of the image was dominating the histogram's structure, hindering our ability to differentiate based on secondary colors. This suggested that our initial approach would not work, and we started to explore other methodologies.

Our second approach sought to classify the colors of the shoes by utilizing a modified form of k-means clustering to identify the top four most prevalent color cluster centers in a given image. More specifically, we ran the scipy python library's vq.kmeans clustering function for 30 iterations with randomly selected initial cluster centers until convergence, where convergence is defined to be:

$$\sum_i (R_i - \bar{R}_i)^2 + (G_i - \bar{G}_i)^2 + (B_i - \bar{B}_i)^2 \leq 0.00005 \quad (6)$$

where $(\bar{R}_i, \bar{G}_i, \bar{B}_i)$ is the cluster center that the current pixel (R_i, G_i, B_i) has been mapped to [10]. Both the cluster centers and the mappings between pixels and their closest cluster centers are returned for the iteration with the lowest overall error. Figure 8 illustrates an example output of this clustering algorithm run on a shoe in our dataset where the four most prevalent colors are shades of light grey, dark grey, blue, and white.

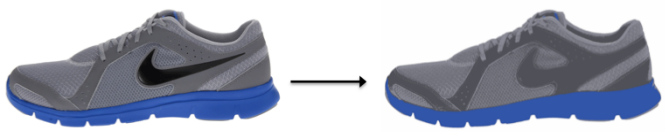

Fig. 8. An example of the output from the k-means based color clustering with four cluster centers that we employed to match colors. In this image, the predominant colors are discovered to be blue, light grey, dark grey, and white.

We then mapped the RGB cluster centers into the CIEL*a*b* (L*a*b*) color space, which is a more visual color metric according to its definition. The L*a*b* color space is designed such that the mathematical distance between two points in the color space approximates perceived visual distance between two colors (i.e., how the human eye would classify their closeness) [3]. Moreover, when compared to the RGB color space, there is much greater uniformity of color spacing, meaning that the relative distances between colors become meaningful measures of ranking color similarity. The conversion from RGB into this space allowed us to calculate similarity between colors using a simple Euclidean distance function and accurately approximate similarity in color as a human would perceive it. These cluster centers were then precomputed for our entire dataset in order to optimize the speed of the algorithm for future color-based comparisons.

Finally, we sorted the four color values of the cluster centers by ascending luminosity, allowing us to make an ordered comparison of the color centroids. The color similarity score of any two images in our data set was calculated to be the sum of the square of the Euclidean distance between each of the color centroids in the L*a*b* color space as described in Equation 7.

$$\arg \min_{i \neq image} ||\text{centroids(image) - centroids(i)}||_2 \qquad (7)$$

Therefore, a smaller similarity score represents a smaller difference between the color distributions of two images and thus a smaller difference in their visual color similarity because of the properties of the L*a*b* color space. A randomly-selected, statistically significant sample was fed into our color similarity engine and the top-10 results for each shoe were manually verified as color matches or not. The similarity measure achieved an accuracy rate of 95%, indicating the success of our color feature construction and similarity matching.

While this is an extremely promising result, we sought to make color classification even more robust by taking into account the frequency of each color appearing in each image. For example, a shoe that is mostly red with blue highlights should not necessarily return a shoe that is mostly blue with red highlights as a top match. To try to achieve this proper weighting, we recomputed all the color cluster centers and also stored their frequencies in the image to file. Then, we multiplied in the absolute value of the difference of frequencies between color centroids as a part of our similarity score computation; that is, if shoe 1's first centroid has a frequency of 75% and shoe 2's first centroid has a frequency of 70%, we will weight the Euclidean distance between shoe 1 and shoe 2's first centroid by 0.05. In practice, this actually didn't do very much to our results, although it did occasionally eliminated extraneous results. We also experimented with transforms into the HSV color space, which is an alternate cylindrical-coordinate-space transform of color [4]. Again, the effects of this were negligible on our algorithm, and thus we decided to stick with the aforementioned approach.

E. Final Algorithm

After identifying these features and optimizing their extractions, we worked to develop our algorithm to compare shoes based on the similarity of these features. Initially, we hypothesized that our data and the problem that we were attempting to solve would inherently lend itself to a clustering algorithm based on the features of each shoe because it would allow us to find the intrinsic "groupings" or structure in the data, if it existed. Unfortunately, we quickly discovered that such a structure did not seem to exist. Our attempts to run k-means on our feature set yielded very poor results as the clusters determined did not result in recommendations that humans would label as similar shoes and instead resembled what seemed to be an almost random assortment of shoes. As a result, we abandoned clustering in favor of a filtering-based system for determining similarity that leveraged a combination of the features that we extracted, as outlined in Algorithms 1, 2, and 3.

In summary, our final algorithm relies on precomputed shape and texture classes for each image in our dataset (which we store on file). When an image is supplied to the algorithm, it is resized and reshaped to proper dimensions and classified based on the shape and texture SVMs, using the feature vectors detailed in Section III (i.e., bounding box for shape and BoW + SIFT for texture). We then filter out any images in our dataset that aren't shape matches for our input shoe due to the high classification accuracy of our shape classification system. Subsequently, we discount the scores of shoes that do not match the image's texture classification to account for the lower classification accuracy of our texture classification system. We then utilize our mask-based and LBP-based shape and texture algorithms to further prune the results, filtering out poor image mask matches and poor LBP matches. Finally, we apply our color matching algorithm (i.e., CIEL*a*b* Euclidean distance with frequency weighting) on what is left in our result set, returning the top 10 closest matches.

Algorithm 1: Shoe Recommendation Algorithm

load $image$ % Shoe image
load $image\_mask$ % Shoe mask
$shape\_SVM \leftarrow computeShapeSVM$
$texture\_SVM \leftarrow computeTextureSVM$
for *each* $shoe \neq image$ *and* $shape\_SVM[shoe] == shape\_SVM[image]$ *in* $dataset$ **do**
    if $texture\_SVM[shoe] == texture\_SVM[image]$ **then**
        $shape\_scores[shoe] \leftarrow nonzero(image\_mask - shoe\_mask)$
    else
        $shape\_scores[shoe] \leftarrow 0.9 \times nonzero(image\_mask - shoe\_mask)$
for *each shoe in top fourth of* $shape\_scores$ *by num of shoes in the same cluster center* **do**
    $texture\_scores \leftarrow compareHist(image\_LBPs, shoe\_LBPs)$
for *each shoe in top third of* $texture\_scores$ **do**
    $overall\_scores \leftarrow \sum (color\_distance)^2$
print top 10 $shoes$ from $overall\_scores$

Algorithm 2: Compute Shape SVM

load $images$ % Images in sample
load $labels$ % Classification lables
load $bboxes$ % Bounding box data for images
for *each shoe in* $images$ **do**
    $data[shoe] \leftarrow bboxes[shoe]$
return train(data, labels)

IV. Experimental Results

For all unsupervised aspects of our algorithms, we measured the accuracy for a randomly selected sample of 400

Algorithm 3: Compute Texture SVM

load $images$ % Images in sample
load $labels$ % Classification lables
for *each shoe in images* **do**
    $descriptors[shoe] \leftarrow SIFT(shoe)$
$centroid, klabels = kmeans(descriptors, 1000)$
$counter = 0$
for *each shoe in images* **do**
    $temp = counter$
    for *each j in* $xrange(temp, temp + len(descriptors[shoe])$ **do**
        $histograms[shoe][klabels[counter]] += 1$
        $counter += 1$
    $histograms[shoe, :] / = len(descriptors[shoe])$
for *each shoe in images* **do**
    $data[shoe] \leftarrow histograms[shoe]$
return train(data, labels)

shoes as this represents a statistically significant sample size at the 95% confidence level with a 5% confidence interval. For supervised aspects of our algorithm, we directly computed the accuracy.

A. Shape

As described in Section III, our shape classification accuracy improved from 55% to 94.6% over four iterations upon our algorithmic approach. The approaches and accuracy rates are showcased in Table I.

TABLE I. SHAPE CLASSIFICATION ACCURACY BY METHOD

Method	Accuracy
pixel-based k-means clustering	55%
pixel and prototype-based k-means clustering	75%
mask and prototype-based k-means clustering	80%
RBF SVM classification of bounding boxes	94.6%

An intriguing set of considerations during our work were the effects that the number of classification categories and the SVM kernel choice had upon our results. In our assessment of the support vector machine-based technique, we considered three different sets of shape classifications for our dataset. The first set of labels contained the following 10 degrees: athletic shoe, converse-type shoe, boat shoe, high-top sneaker, slipper, dress shoe, casual sneaker, boot, sandal, and other. The second set of labels contained the following 5 degrees: low-top shoe, high-top shoe, sandals, boots, and other. And the third set of labels contained the following 3 degrees: low-top, high-top, and other. The accuracies of these three labeling approaches are summarized in Table II.

TABLE II. SHAPE CLASSIFICATION ACCURACY BY LABELS

Method	Accuracy
10 degrees	54%
5 degrees	94.6%
3 degrees	90.5%

As such, we realized that there was a trade-off between the granularity of the classifications and the accuracy of our algorithm. The labels with 11 degrees had a high level of granularity, but a low accuracy rate, while the labels with 5 degrees had a lower level of granularity, but a significantly higher accuracy rate. As we felt that the labels with 5 degrees captured the granularity that we needed for the purposes of our recommendation system, we decided to move forward with the labeling scheme.

Another major decision that we had to make in the computation of our final accuracy measure for the support vector machine-based shape classification approach that we employed was the type of kernel that we would utilize in our SVM.

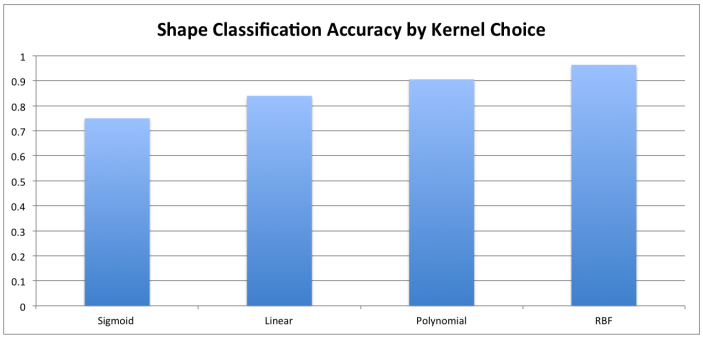

Fig. 9. The accuracy rates in shape classification across RBF, 3-degree polynomial, sigmoid, and linear kernels.

From the results in Figure 9, we can see that the radial basis function kernel outperforms the three other kernel choices that we considered. This makes sense as the radial basis function kernel defines a much larger function space than the other kernel choices. This makes it far more flexible by allowing it to model far more functions with its function space than the sigmoid, linear, and polynomial kernels that we assessed.

B. Texture

As mentioned in Section III, we were able to increase the texture classification accuracy from less than 25% to about 80% by iterating upon our algorithm and testing a variety of algorithmic approaches. These approaches are summarized in Table III.

TABLE III. TEXTURE CLASSIFICATION ACCURACY BY METHOD

Method	Accuracy
KL-divergence on LBP histograms	25%
Spatial Pyramid Matching on LBP histograms	65%
RBF SVM classification of SIFT-generated BoW models	80%

As in our assessment of support vector machine-based classification for shapes, we also assessed the number of degrees in our labelings in our classifications for textures. We tried two labeling schemes. The first labeling scheme had 7 degrees, namely cloth, mesh, leather, suede, rubber, canvas, and other. The second labeling scheme had 3 degrees, namely mesh/athletic, leather, and other. The performance of these two schemes is captured in Table IV.

Once again, we noticed a trade-off between granularity and accuracy. Valuing accuracy in our recommendation system and realizing sufficient granularity in the three-degree labeling scheme, we decided to move forward with that approach in our final algorithm.

TABLE IV. TEXTURE CLASSIFICATION ACCURACY BY LABELS

Method	Accuracy
7 degrees	58%
3 degrees	80%

As in our shape classification, a support vector machine using a RBF kernel yielded the best results. These results are shown Figure 10.

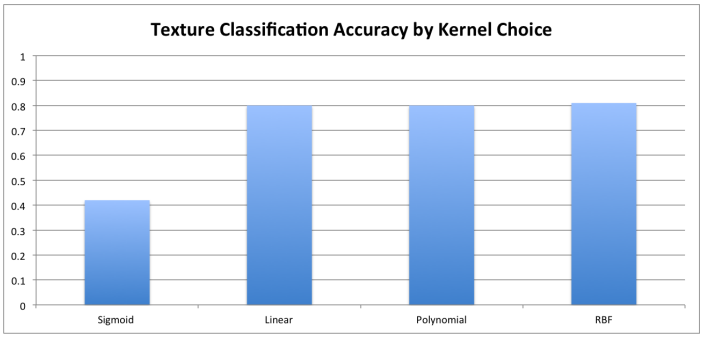

Fig. 10. The error rates in texture classification across RBF, 3-degree polynomial, sigmoid, and linear kernels.

C. Color

As described in Section III, when assessing a randomly-selected, statistically significant sample, we manually verified the top-10 outputted results as color matches or not. Through this process, the color feature accuracy was assessed to be about 95%, indicating successful color feature extraction through our clustering and (L*a*b*) color space conversion-based approach.

D. Overall

Because of the unstructured nature of this problem, determining a system for measuring the accuracy of our recommendation engine was not possible in a traditional sense. Thus, in order to appropriately evaluate our system, we calculated accuracy by simulating the way a user might judge the accuracy of our system. To do this, we ran through a randomly selected, statistically significant sample of images in our dataset and examined the 10 best matches for each of these images. We then judged each of the recommendations on whether it closely matched the original shoe in each category of shape, color and texture, and computed a score from 0-10 for each of these categories based on the number of shoes in the top-10 that closely matched the original shoe in the respective category. In other words, for each category of each image, we had a score that was $\frac{x}{10}$ where x was the number of recommended shoes that closely resembled the original shoe's traits in that category. We then averaged the ratings for all the shoes in order to receive an accuracy score for our algorithm in suggesting similar textures, colors, and shapes. Thus, in total, our accuracy for each attribute was computed as shown in Equation 8.

$$\frac{\sum \text{recommended shoes that matched category}}{\sum \text{total recommendations}} \quad (8)$$

Using this method, we were able to determine our recommendation engine achieved 96% accuracy in suggesting similarly shaped shoes, 94% accuracy in suggesting similarly colored shoes, and 88% accuracy in suggesting similarly textured shoes. Figure 11 demonstrates a standard output from our recommendation system.

Fig. 11. This is a randomly selected output of our recommendation system where the shoe on the left is the original image that the user queried and the 3 shoes on the right are the top-3 recommended shoes.

E. Dealing with Noise

We also worked on experimenting with noisy images in our analysis. We first wrote an algorithm that would be able to extract a shoe from a noisy image and white out the background by once again leveraging a thresholding function like the one in Equation 1 and using OpenCV's external contour detection functionality and were able to convert images taken from a smartphone camera as show in Figure 12.

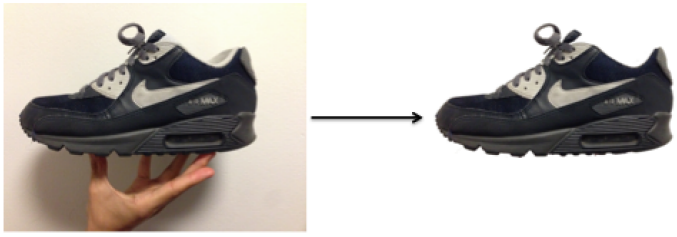

Fig. 12. This is shoe that has been converted to a form that can be used by our recommender system.

This approach worked fairly well, but also depended on the shoe being of a relatively dark color to find its largest external contour and did not perform all that well on light-colored shoes as a result. We then ran this image in our recommendation system and had mixed results as shown in Figure 13.

Fig. 13. This is the output of putting the noisy shoe image through our recommendation system.

These results were promising in that they had the correct color recommendations and generally had the correct shape recommendations. However, they were problematic in that the texture of the noisy shoe image had been misclassified as "leather." As such, only leather shoes were returned by our recommendation engine, while the original shoe was actually had a "meshy/athletic" texture. This seemed to have been the result of illumination that was inconsistent with the

standardized illumination in our clean Zappos dataset. As such, we decided to make our algorithm more flexible and stopped completely filtering out results that didn't match our texture SVM-based classifications by adding the discounting factor mentioned in Section III. Doing this resulted in the far more promising results depicted in Figure 14.

Fig. 14. This is the output of putting the noisy shoe image through our recommendation system with our modified algorithmic approach.

While our current algorithm does a fair job at generating results with a reasonable amount of shape and color accuracy for user-generated, noisy images, there is still work to be done to better capture the texture of such images. That said, based on previous work and our own experiences while working on this project, we realize that near-perfect texture classification is something that should be worked towards, but is extremely difficult to achieve. As such, these results were quite promising in that they suggest that we should work towards making our algorithm even more flexible and heuristic-based than it is right now to avoid a missed texture classification resulting in poor results when the rest of our algorithm is performing well. Moving forward, our algorithm needs to be made more robust to deal with variance in rotation, illumination, scale, and translation more effectively.

V. Conclusions

As alluded to in Section IV, our results are encouraging in that they suggest that we have developed a successful approach to address this problem. Not only do we have high accuracy rates for classifying clean datasets, but we have also been able to parlay some of this success to make our algorithmic approaches compatible with noisy, user-generated images, a key component of this problem moving forward if we want it to eventually become a consumer-facing application.

That said, we certainly have room for improvement. Though it may be difficult to significantly improve upon the 4-12% recommendation error rate in our three feature categories, we can improve our overall algorithm by making it more flexible. This can be seen in the results of Section IV-E, as making our algorithm more flexible gave us fairly good results despite a textural misclassification. Our rationale behind making this change is that the SVM-based texture classification accuracy of our algorithm is approximately 80%. Since this isn't 100% accurate, we thought that employing a weighting scheme to different textural classifications instead of strictly filtering on the basis of these classifications, would ensure that an incorrect classification wouldn't completely prevent the final recommendations from being accurate.

Additionally, our algorithms are not yet robust to a high degree of variability and cannot be deployed in a setting where users can input extremely noisy images. Our initial results upon trying to tackle noisy datasets have been promising and we are excited to continue making our system more flexible and robust. To do so, we need to reevaluate the ways in which we extract and assess different features from our images as we need to be conscious of images with inconsistent illumination, positioning, scale, and orientation. Our algorithm for shape feature detection is dependent on orientation as our algorithm demands contours of the left view of leftshoes. Furthermore, our algorithms for color and texture are dependent on lighting and would require image orientations similar to those in our dataset for any sort of reasonable comparison between two images. To address these issues, we will work on normalizing for illumination and potentially analyzing more views of a shoe than just the left view to be able to account for images of shoes that have a different orientation.

We are pleased with our progress thus far and look forward to further enhancing our work. In the coming quarter, we want to focus on dealing with noisy images and building a mobile application with this technology that people can use to help them shop in the brick-and-mortar setting and feel that CS 231M would be a good setting for us to continue building out this technology.

Acknowledgment

We would like to thank the CS 231A course staff for all of their support, advice, and encouragement throughout the course. We especially appreciate the assistance of Professor Silvio Savarese and CS 231A Head TA, Kevin Wong, for all of their extremely helpful insights to improve the algorithmic approaches that we employed in our project.

References

[1] *Bag of Words Model*. http://cs.nyu.edu/ fergus/teaching/vision_2012/9_BoW.pdf.

[2] *Basic Thresholding Operations - OpenCV Documentation*. http://docs.opencv.org/doc/tutorials/imgproc/threshold/threshold.html.

[3] *CIELAB Color Models Technical Guide*. http://dba.med.sc.edu/price/irf/Adobe_tg/models/cielab.html.

[4] *HSV*. http://www.cs.rit.edu/ ncs/color/t_convert.html.

[5] *Kaleidoscope*. http://kalei.do/.

[6] *lbp.h File Reference*. http://www.vlfeat.org/api/lbp_8h.html.

[7] *Modista*. http://modista.com/.

[8] *National Shoe Retailers Association*. http://www.nsra.org/?page=About_Us.

[9] *OpenCV Histogram Documentation*. http://docs.opencv.org/modules/imgproc/doc/histograms.html.

[10] *SciPy Cluster VQ Kmeans Documentation*. http://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.vq.kmeans.html.

[11] A. Borras, F. Tous, J. L., and Vanrelld., M. Classification of clothing using midlevel layers.

[12] B. Willimon, I. W., and Birchfield., S. Classification of clothing using midlevel layers.

[13] Ping Chuan Wang, Stephen Miller, M. F. T. D., and Abbeel, P. Perception for the manipulation of socks.

[14] S. Lazebnik, C. S., and Ponce., J. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, New York, June 2006, vol. II, pp. 2169-2178.*