Final Project Report on Visual Search in Fashion: Where can I find this shirt?

Fatma Moalla
fatma.moalla@student.ecp.fr

Mohamed Kerroumi
mohamed.kerroumi@student.ecp.fr

Niraj Srinivas
niraj.srinivas@student.ecp.fr

François Le Roux
francois.le-roux@student.ecp.fr

Abstract

Visual Search is an application of Computer Vision that has gained a lot of importance and has been intensely developed in the last two decades. Conceptually humans are able to find images that are very similar with a glance but this is not an evident task for an algorithm. It is indeed not a pixel by pixel search as in text search and there's a need for algorithms to have a local and global perception when trying to match images. We explain why we feel this is an application that has the potential to disrupt the fashion industry given its ability to provide products that the user might subconsciously want even without them explicitly writing down in words. In today's real-life applications of visual search, they are usually powered by deep learning algorithms. But as this project is a part of our course on Visual Computing, we aim to use a database built on fashion products and experiment different descriptors such as SIFT and SURF which are global descriptors and we also use a local descriptor, HOG to find the nearest neighbors given an input image using multiple unsupervised learning methods and finally compare them and also with a state of the art performance algorithm in visual search, this being a fine tuned CNN network. Finally, we attempt to explain why our descriptors have the performance they do and suggest improvements and further experiments that could be performed.

1. Motivation

Visual Image Search is a concept that had gained enormous traction during the year 2000 when Jennifer Lopez had shattered internet search query records during her Grammy Awards presentation. It was at this moment that the industry such as Google realised that people wanted more. They wanted more than just texts and wanted to search by images and this led to the birth of Google's image search. Image searching is exponentially different from textual search as we are not trying to match pixel value by pixel compared to text search. Instead we want to understand the image and find the relevant features of the image.

In our project of visual search, we want to take an image as an input, understand its features and provide suggestions on similar images and all information linked to this image. These images could be anything as long as we have similar images in our database and we expect our algorithms to find these images subsequently.

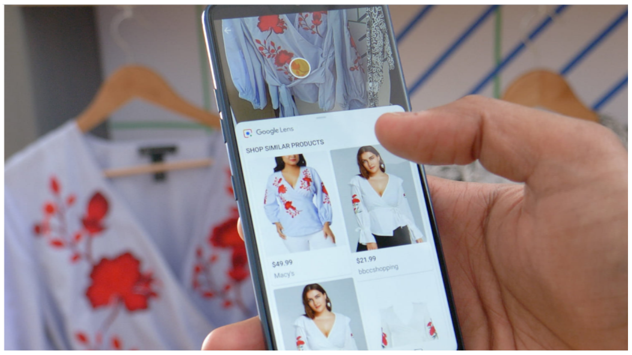
Figure 1. Google Lens Suggesting Products

The questions that we would like our algorithms to answer would be if it is able to find the same object in different images, is it able to find images that are similar to the input in shape, belong to the same category of objects present in images, similarity in size, colour, texture and materials. We could also impose questions if the algorithms are able to retrieve images belonging to the same gender, in case of clothes. Another more real life question that needs to be addressed would be if the algorithm is able to identify similarity even in the presence of other objects and background.

We see potential application of this being implemented in online shopping, especially when customers are searching for products to buy. There is a certain limit on the amount of information text can capture while the user inputs their search query. But looking at the image of the products, we can get a better idea of the type of products, style and colors they prefer and this could be potential way of personalising offers. This was the motivation behind testing our algorithms on fashion products. We also see potential when users want to know what a certain object is when they have absolutely no information on the product or if they just want to know where they can buy this online and compare prices by different providers.

2. Problem Definition

Having described the aim of this project as finding the most similar images given an input, we now define each step involved and the words associated. We say two images are *similar* if they've got similar *features*. Features are basically pieces of information that describe the information present in the data. We use feature extraction methods, which are basically secondary values derived from the data which help understand the data in a different space. We shall use various high and low level feature extractors to understand the images. Low level feature extractors in the sense that they operate directly on the pixel value around a small window and high level extractors much more abstract. The way we obtain our feature vectors and their functioning are detailed in the upcoming sections.

Assuming we have our features for every image, we want to find images which are closest to the input image. We use cosine similarity as a metric to compare how close two vectors are and this is defined as :

$$\text{similarity } = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|}$$

Two vectors, A and B that are close to each other are said to have values closer to one. Thus the images we want to retrieve are expected to be the ones with the highest cosine similarity. But this might not always be the best metric especially when dealing with sparse data as often cosine similarity might lead to values being zero following the dot products, but on the contrary the images might still be close. Thus we decided to choose the Euclidean distance as a metric to compare similarity. Given vectors $\mathbf{p}$ and $\mathbf{q}$, the Euclidean Distance $d(\mathbf{q}, \mathbf{p})$ is defined as:

$$d(\mathbf{q}, \mathbf{p}) = \sqrt{(q_1 - p_1)^2 + \cdots + (q_n - p_n)^2}$$

Our objective is now to reduce this Euclidean Distance and we say that the most similar image is the one that minimizes the metric. We use different tree and hashing based methods to speed up this search and are explained in the methodology section. Without doubt the most difficult part is in finding these feature vectors and there has been lot of progress and work on what type of features to be found, and how to find which are discussed in the next sections.

3. Related Work

Visual Search in the Fashion industry, remains a very inspiring subject for both retailers like ASOS or IKEA for example and Tech-Giants like Google and Amazon. It has also inspired the Computer Vision research community. In fact, Fashion analysis has drawn increasing attention in recent years because of its various applications like clothing recognition and segmentation[5], recommendation[10].

In fact, the most recent papers and state of the art techniques focus mainly on weakly annotated datasets to perform prediction, in response to the industrial needs and the complexity of the annotation process. Corbière and Al. present[1] a method to learn a visual representaxtion adapted for e-commerce products which is based on weakly supervised learning. Their state of the art model learns from noisy datasets crawled on e-commerce website catalogs and does not require any manual labeling. They also demonstrate that the learnt representation of the images using Convolutional neural Networks (CNNs) and more precisely Resnet50 is suitable for image retrieval. They also achieved outstanding results of nearly 92.8% of accuracy on top 5 neighbors based on categories prediction task which is the main object our project.

Another outstanding and very recent paper based of the same dataset and the same Visual search task is *Study on Fashion Image Retrieval Methods for Efficient Fashion Visual Search* paper[9]. In fact Park and Al. leverage and the Fashion Image Retrievel (FIR) and the performance of Resnet50, DenseNet121 for example to get an veruy high accuracy of 82.6% of accuracy of the Top 5 neighbors.

Since we wanted to focus more on the Computer Vision side, rather than the Machine and Deep Learning aspects, we also examined less recent papers that take more advantage on the local features of the images. We were inspired by the *Describing Clothing by Semantic Attributes* paper [2] to use SIFT for local descriptors' extraction. In fact, in this paper, extracts low-level features in a pose-adaptive manner, and combine complementary features for learning attribute classifiers. They also use mutual dependencies between the attributes or categories based on a Conditional Random Field(CRF) model to improve the predictions from independent classifiers. The performance is then validated on randomly-taken photos and not on standard images taken with well-controlled settings. They also use SIFT descriptors and texture descriptors as an input for the SVM classifier that will perform the prediction.
In our approach, we wanted to leverage the techniques of the papers from literature. On the one hand, we focused more on the Computer Vision side to test our solution and that's why we used three different solutions: **SIFT**, **SURF** and **HOG**. On the other side we wanted to compare the latter result with a CNN in order to compare with the state of the art algorithms and also to discuss and assess the local features extraction techniques. In the following part, we will explain precisely the solutions that we used to find the top 5 nearest neighbors for a selection of fashion images.

4. Model and Methodology

In this part, we will describe first the dataset, then the whole pipleine of the model. Second, we will give a detailed explanation of each method that we used to perform Visual search and to find the nearest neighbors for a given image from the Fashion Datatset[7].

4.1. Data collection

The aim of this project is to perform Visual Search in the Fashion world by using the most performing descriptors. Therefore, we were inspired by the Deep Fashion Dataset[7] that contains 300,000 images divided into training, validation and testing set and contains annotations that include bounding boxes.

The following image gives a glimpse of the *Deep Fashion* categories.

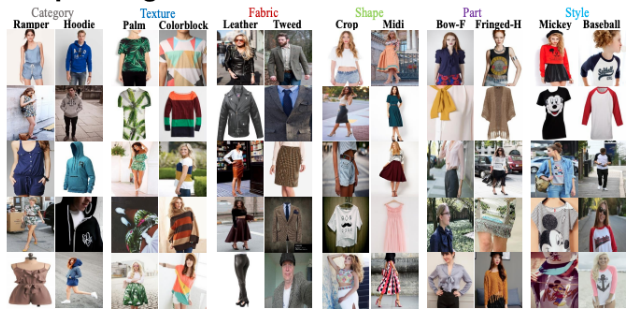

Figure 2. Different categories of the Deep Fashion Dataset

While computing our descriptors and due to computational costs, we were constrained to use another similar, yet lighter, dataset from Kaggle [8]. The initial Kaggle dataset contained 44,000 different images with 50 Subcategories and 1,000 descriptive attributes and clothing landmarks. The categories, for example, represents the type of the clothing like "blouse" or "dress"... or the type of prints "flowered", "potts", ...(Fig 2).

For this project, first, we worked on the 50 Subcategory attributes because these features are easily identifiable and can easily be interpreted.

Second, After several considerations and due to ressources limitations, we limited our dataset to **1217** images containing the 50 categories with a repartition 4-5 images per category. In fact, applying SURF or SIFT on 44,000 images was a very difficult and challenging task because of the computational time and space complexity.

Third, we downgraded the resolution of the images and we resized them in order to limit calculations time. Therefore, our dataset was ready and we were able to apply it on all the descriptors and algorithms that we will be describing in the following section.

4.2. Model pipeline

For this project our objective was to compare the results of 3 different feature-based algorithms (HOG, SIFT and SURF) for visual search with the results of the current state of the art approach, which is to use a Convolutional Neural Network (CNN).

For this, we were able to train the CNN on the whole 44k images dataset, but due to resource limitations we had to create a subset of our Deep Fashion dataset to be able to implement and test the feature-based algorithms. We thus extracted a balanced subset composed of 1217 images where every category of fashion products was equally represented. We also created a test set of 10 images, for each of which the different algorithms would be tasked to retrieve the 5 closest images in the dataset. This way, we were able to implement the 3 feature based-algorithms and the CNN, and to visually compare the quality of their results on the same task.

Having described the pipeline of the project, we will now give a more detailed explanation of each method that we used to perform the Visual search.

4.3. HOG

The Histogram Of Gradients (HOG) [3] is a technique whose main objective is to convert an image to a feature vector, which is important for object recognition.

The keys steps of the algorithm are described below:

- Image preprocessing:

  The HOG is calculated on cells, which in our case are chosen to be of size 16 x 16 pixels. Therefore we need to divide the image into cells, which can have a certain overlap.

- Calculate the image gradient:

  To calculate the HOG descriptor, we need to calculate the horizontal and vertical gradients g_x and g_y around each point in each cell. For this, we can use a Sobel filter. Once these are computed, we need to find the magnitude and direction of the gradients, using these equations :

$$magnitude = \sqrt{g_x^2 + g_y^2}$$

$$\theta = \arctan(\frac{g_x}{g_y})$$

- Calculate histogram in each cell:

  Once we have the magnitude and direction of the gradient for each pixel in the cell, we can put them together in a histogram. For this, we create 9 bins which correspond to different angles (0, 20, 40, 60, ..., 160). Then for each pixel, we add the magnitude value of the gradient to the corresponding bin, proportionally to the direction value of the gradient.

- Calculate the HOG feature vector:

  Finally, we normalize the histograms obtained for each cell, in order for them to not be affected by lighting variations. We then concatenate all these histograms to create the HOG feature vector.

Matching:

To do image matching with the HOG feature vector, we use this method: the similarity between two images is defined as the cosine similarity between their respective HOG vectors. When doing visual search, we thus want to find the 5 images that are the closest to the query image in the sense defined above. To speed up the computations, we use a K-nearest-neighbors algorithm, which allows us to retrieve the approximate 5 nearest images efficiently.

4.4. SIFT: Scale Invariant Feature Transform

Scale-Invariant Feature Transform (SIFT) [6] is a feature based detection algorithm which has as a main objective to build scale-invariant image descriptors. The key steps of the algorithms are described below:

- Feature point detection:

  To detect points of interest in the image, we approximate the Laplacian of Gaussian with the Difference of Gaussian (DoG). This is done by calculating the convolution of the grayscaled image with Gaussian filters at different scales, and taking the difference between the results.

  Then we extract the local extrema in the DoG pyramid, by comparing each point to its 8 neighbors at the same level, 9 neighbors in the level above and 9 neighbors in the level below.

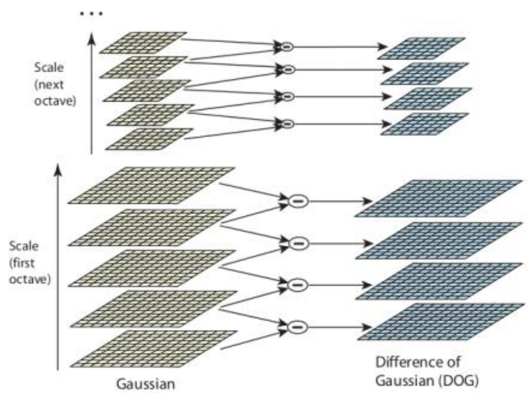

Figure 3. SIFT Difference Of Gaussians

- Feature point localization:

  The first step produces too many keypoint candidates, some of which are unstable. Moreover, they are coarsely localized, both in space and in scale. To refine the localization of the keypoints, we do an interpolation using the quadratic Taylor expansion of the DoG function. Then to filter out the unstable keypoints, we first discard the low-contrast keypoints, by computing the value of the second-order Taylor expansion of the DoG function and setting a threshold for keeping the corresponding keypoint. Secondly, as the DoG function will have strong responses along edges, even if the candidate keypoint is not robust to small amounts of noise, we need to eliminate edge response. For this, we compute the ratio of the eigenvalues of the second-order Hessian matrix and again we set a threshold on this value.

- Orientation assignment:

  To achieve invariance to rotation, we compute the Histogram of Gradients around the point of interest. Once the histogram is filled the orientation corresponding to the highest peak is chosen as the orientation of the keypoint.

- Feature descriptor generation:

  Finally to compute a descriptor vector for each keypoint, which consists of a normalized 128-dimensional vector, we use orientation histograms created on 4 x 4 pixel neighborhoods with 8 bins each. These histograms are weighted by a Gaussian function, and then the concatenation of all these values form the feature vector.

Matching:

For matching, we again use cosine similarity between the feature vectors. As the descriptors are of a large size, to speed up calculations we use a hashing table to store them and to retrieve them more efficiently.

4.5. SURF

Speeded Up Robust Features (SURF) [4] is a feature-based algorithm that is inspired by SIFT, but designed to be faster. The key steps of the algorithm are described below.

- Keypoint detection:

  Instead of using the DoG, SURF uses a blob detector based on the Hessian matrix to find points of interest. The determinant of the Hessian matrix is used as a measure of local change around the point and points are chosen where this determinant is maximal. SURF uses square-shaped filters as an approximation of Gaussian smoothing. To speed up the calculations, we use something called the integral image. It is defined such that the value at any point (x, y) in the integral image is the sum of all the pixels above and to the left of (x, y), inclusive. By using this, computing the convolution of an image with a box filter is very fast.

- Keypoint localization:

  In SURF, to locate interest points at different scales, instead of using a pyramid of images smoothed by a Gaussian filter at different scales, we filter the same image with box filters of different sizes, which is faster to compute. The maxima of the determinant of the Hessian matrix are then interpolated in scale and image space to obtain the localization of the keypoints.

- Orientation assignment:

  To assign an orientation to the keypoints, the Haar wavelet responses in both x and y directions within a circular neighbourhood around the point of interest are computed. The obtained responses are then weighted by a Gaussian function centered at the point of interest. The dominant orientation is estimated by calculating the sum of all

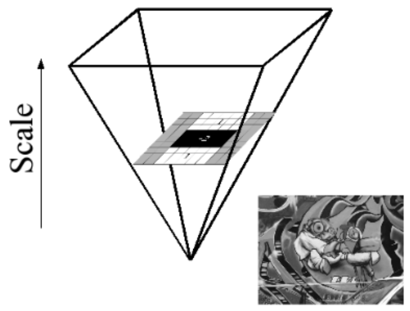

Figure 4. SURF filters of different sizes

responses within a sliding orientation window of size $\pi/3$. The horizontal and vertical responses within the window are summed. This gives a local orientation vector and the longest such vector overall defines the orientation of the point of interest.

- Feature descriptor generation:

  Finally the descriptors are computed by extracting a square region around the point of interest and computing the Haar wavelet responses at regularly spaced sample points. These responses are weighted with a Gaussian, and then concatenated to form the feature vector.

Matching:

To do Visual search with this algorithm, we use the Visual Bag of Words method. We compute all the features for all the images in the dataset, then perform a K-means clustering on them. The cluster centers that are obtained are then used as visual words. To compare two images, we first match every feature of each image to its closest visual word to compute a histogram of the occurrence of each visual word, which we also weight by TF-IDF and normalize. Then the similarity between the two images is defined as the cosine similarity between the two histograms. As previously, we use K-nearest-neighbors to speed up the calculation of the 5 closest images.

4.6. Convolutional Neural Network

Most of state-of-the-art visual search applications use convolutional neural networks in their model, CCNs are very useful in this kind of computer vision task, because they use a spacial mathematical architecture called a convolution layer that sweeps a small window over image, drastically reducing their dimensionality while retaining salient features like shapes, colors, and edges. In fact, the most successful CNNs use several convolution layers to create so-called Deep Networks that are extremely successful at visual recognition tasks.
These CNNs have a large number of settings to configure, and they typically require large datasets and a lot of training in order to reach very high accuracy in computer vision task.
The idea behind the visual search engine with CNN is the use of intermediate results of a pre-trained CNN as **Feature Map** of the input image and finding products with similar feature maps to this one. The logic behind this is that a CNN trained in visual classification typically has two sections: a series of convolutional layers trained at extracting meaningful visual information from the input image and a series of dense layers trained at performing the classification task. Using the output of the convolutional layers to represent images allows us to perform comparisons of the visual content of images.
For this task, we didn't build a network from scratch and decided to use the ResNet-18 network architecture with the pre-trained weights on ImageNet Challenge because of its strong performance at visual classification. We have changed the number of neurons in the top layer to 45 (number of categories) and used the Softmax activation function to predict the probabilities to belong to categories. After that, we fine tuned the weights of the network by performing images classification on the deep fashion dataset we used, this kind of technique is called Transfer Learning. Then we are going to use the output of the second-last layer (outputs a vector of size 512), as feature map in our product search.
Once the model is set up, we trained the model on a GPU for about 5 hours and after only 6 epochs, the model reached more than 99% of accuracy in multi-classification. We then used this trained model to create a dictionary saving for each product image in the training set its corresponding feature map.

Now given a new image, we want to find the most similar images, to do that, we pass this new image through the trained model and extract its feature map,

then we calculate the cosine similarity of this feature map to all 44K feature maps of the training set and pick up k images with the highest cosine similarity with the input image. This process is computationally expensive and in computational complexity notation is an O(N) problem and will take exponentially more time to retrieve similar images as the number of images increases. To solve this problem, we used locality sensitive hashing(LSH) which is an approximate nearest neighbor algorithm which reduces the computational complexity to O(log N). In short, LSH generates a hash value for feature maps while keeping spatiality of images in mind; in particular; images that have similar feature maps will have a higher chance of receiving the same hash value.
Now the approach to find the k nearest neighbors for a new image is slightly different, we extract the feature map of the new image, we will use LSH to create a hash for the given image and then compare the distance from feature maps of the images in the training set which share the same hash value. Hence, instead of doing similarity search over the whole deep fashion dataset, we will only do a similarity search with a subset of images which shares the same hash value with the input image. For our project, we are using lshash3 package for an approximate nearest neighbor search.
For images in the test set, we searched only for the 5 nearest neighbors, and it took only *9 s* to find those images once the model is trained and the hash table generated.

5. Evaluation and results

5.1. Evaluation Method

To compare the performance of the different approaches used to build the visual search engine, we created a test set of 10 images of different categories, and generated for each method the 5 most similar images. then we compared visually the matching between the generated images and we used also the cosine similarity as a metric for image matching.

5.1.1 Qualitative Evaluation

The following figures represent the outputs of the different methods used to find the 5-nearest neighbors for an image in the test set, as you can see, the matching between the input image and the nearest neighbors

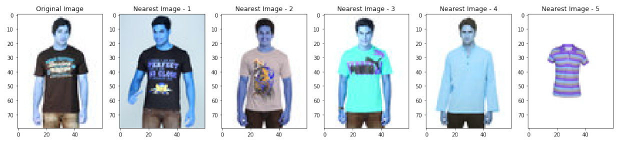

Figure 5. HOG Result

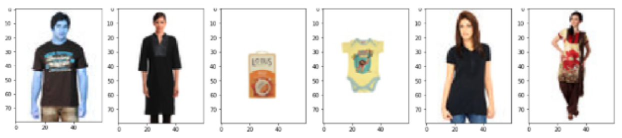

Figure 6. SIFT Result

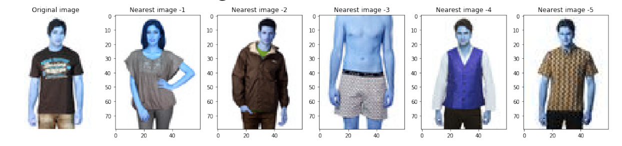

Figure 7. SURF Result

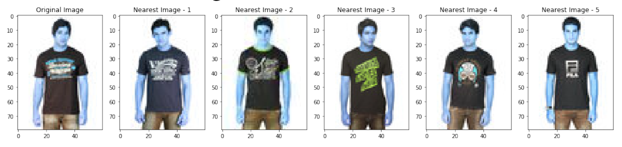

Figure 8. CNN Result

given by the SIFT and the SURF methods is very bad, and the outputted images are very different from the input. HOG gave good results, the outputted images are of the same category but do not have the same color. However there is a perfect visual matching between the input image and the nearest neighbors found by the CNN. Hence we can draw the following comparison on the different methods.

Approach	HOG	SIFT	SURF	CNN
Visual Matching	Good	Very Bad	Medium	Perfect

Table 1. Qualitative comparison of methods

5.1.2 Quantitative Evaluation

We considered the Cosine Similarity as a metric for the evaluation of the visual matching between images.

$$Cosine\_Similarity(img1, img2) = \frac{\langle img1, img2 \rangle}{\|img1\| \|img2\|}$$

We take one of the images in the test set and flatten it, we find the 5-nearest neighbors with the chosen method ,we flatten each of them, we calculate the cosine similarity with the input image, then we average on the 5 images and the 10 images of the test set. The results are below:

Approach	HOG	SIFT	SURF	CNN
Avg Cosine Similarity	96.3%	93.4%	94.6%	98.11%

Table 2. Quantitative comparison of methods

The values of the above table confirm the visual analysis we already did, CNN approach has the best Average Cosine Similarity , HOG approach has a good Average Cosine Similarity, and the ones of SIFT and SURF approach are very low.

5.2. Conclusion

From our algorithms experimented we noticed a few patterns. Local descriptors such as HOG was able to function well with our application of visual search and provided good qualitative and quantitative matching. It was also able to capture semantics of the product and provide gender specific matching. Global descriptors such as SIFT and SURF were computationally more expensive and needed much larger images to find relevant features. They were still able to capture subtle details such as the shape of the product, but failed otherwise. We do believe that global descriptors would be much more powerful in a complex image, given a background and other objects which might be considered as HOG as noise and could result in performance degradation. It was also remarked that features such as cosine similarity of two images from their pixel value was a very bad estimate of the resemblance. Finally with our CNN fine-tuned network, we compared why the state of the art in image search is based on these methods as evidently we just saw their extremely good performance and it's ability to scale.

Our next steps that we would have tried if not for the time constraint, would have been to expand our database with larger images. Adding complexity to the products in the images, and adding background and other objects in the vicinity would make the query image more realistic to our real world and this would be an ideal way to check our algorithms. This is built on the intuition that a global descriptor would be able to capture these features whereas HOG might fail in doing so.

References

[1] C.Corbiere, H.Ben-Younes, A.Rame, and C.Ollion. Leveraging weakly annotated data for fashion image retrieval and label prediction. pages 2268–2274, 10 2017.

[2] H. Chen, A.Gallagher, and B.Girod. Describing clothing by semantic attributes. pages 609–623, 10 2012.

[3] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 2005.

[4] Luc Van Gool Herbert Bay, Tinne Tuytelaars. Surf: Speeded up robust features. `http://people.ee.ethz.ch/~surf/eccv06.pdf`.

[5] S. C. Hidayati, C. You, W. Cheng, and K. Hua. Learning and recognition of clothing genres from full-body images. *IEEE Transactions on Cybernetics*, 48(5):1647–1659, May 2018.

[6] David G. Lowe. Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision, 2004.

[7] L.Ziwei, L.Ping, Q.Shi, W.Xiaogang, and T.Xiaoou. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations, June 2016.

[8] P.Aggarwal. Fashion product images (small), 2019.

[9] S. Park, M. Shin, S. Ham, S.Choe, and Y.Kang. Study on fashion image retrieval methods for efficient fashion visual search. 07 2019.

[10] Y.Ma, J.Jia, S.Zhou, J.Fu, Y.Liu, and Z.Tong. Towards better understanding the clothing fashion styles: A multimodal deep learning approach. 2017.