Getting the Look: Clothing Recognition and Segmentation for Automatic Product Suggestions in Everyday Photos

Yannis Kalantidis[1,2]
ykalant@image.ntua.gr

Lyndon Kennedy[1]
lyndonk@yahoo-inc.com

Li-Jia Li[1]
lijiali@yahoo-inc.com

[1]Yahoo! Research
[2]National Technical University of Athens

ABSTRACT

We present a scalable approach to automatically suggest relevant clothing products, given a single image without metadata. We formulate the problem as *cross-scenario retrieval*: the query is a real-world image, while the products from online shopping catalogs are usually presented in a clean environment. We divide our approach into two main stages: a) Starting from articulated pose estimation, we segment the person area and cluster promising image regions in order to detect the clothing classes present in the query image. b) We use image retrieval techniques to retrieve visually similar products from each of the detected classes. We achieve clothing detection performance comparable to the state-of-the-art on a very recent annotated dataset, while being more than 50 times faster. Finally, we present a large scale clothing suggestion scenario, where the product database contains over one million products.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*search process, retrieval models*; I.4.10 [**Image Processing and Computer Vision**]: Image Representation

Keywords

Clothing suggestion, clothing detection, large-scale clothing retrieval, clothing segmentation, automatic product recommendation

1. INTRODUCTION

Many hours are spent every day in front of celebrity and non-celebrity photographs, either captured by professionals in fashion magazines or by friends on Facebook and Flickr. With online clothing shopping revenue increasing, it is apparent that suggesting related clothing products to the viewer can have substantial impact to the market. As only a very small percentage of such images has metadata related to its clothing content, suggestions through automatic *visual* analysis can be a very appealing alternative to costly manual annotation.

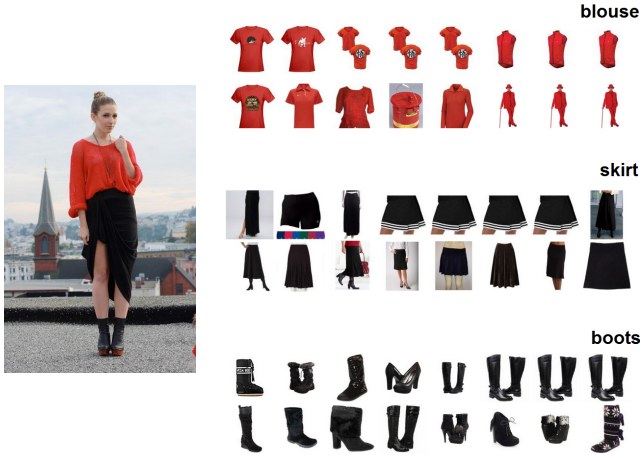

Figure 1: Left: The query image, which is a real world image depicting the person whose clothing style we want to 'copy'. Right: Clothing product suggestions, based on the detected classes and the visual appearance of the corresponding regions.

We formulate automatic suggestion of products from online shopping catalogs as a *cross-scenario retrieval* problem[8], since the query is a real-world image, while the related products are usually presented in a clean and isolated environment. As we want our approach to be able to handle product database sizes containing millions of items, we need to apply large scale image retrieval methods that use indexes of sub-linear complexity. However, the difference in context between the query and results does not allow traditional image retrieval techniques to be directly used.

The example seen in Figure 1 depicts the main use case for the desired application. The query image on the left is a real-world photo with a stylish woman in a prominent position. Product photos from online shopping databases are like the ones shown on the right, *i.e.* clean photos of clothing parts that are either worn by models or are just presented on a white background. Product photos are usually accompanied by category level annotation.

We divide our approach in two main stages: a) Detect the clothing classes present in the query image by classifica-

ICMR'13, April 16–20, 2013, Dallas, Texas, USA.

tion of promising image regions and b) use image retrieval techniques to retrieve visually similar products belonging to each class found present.

The contribution of our work is threefold: We present a novel framework for a fully automated cross-scenario clothing suggestion application that can suggest clothing classes for a query image in the order of a few seconds. We propose a simple and effective segment refinement method, in which we first limit segmentation only on image regions that are found to have a high probability of containing clothing, over-segment and then cluster the over-segmented parts by appearance. Finally, we present a novel region representation via a binary spatial appearance mask normalized on a pose estimation referenced frame that facilitates rapid classification without requiring an actual learning step. The presented framework is fast and scalable, allowing our clothing classification and similar product search to trivially scale to hundreds of product classes and millions of product images respectively. To our knowledge, this is the first time that clothing suggestion together with fine-grained clothing detection and segmentation is performed at this scale and speed.

The rest of the work is organized as follows: Section 2 discusses the related work. Section 3 presents our clothing class detection approach and Section 4 describes the large scale retrieval experiments. Section 5 presents the experimental evaluation of the approach and finally Section 6 concludes the paper and summarizes the work.

2. RELATED WORK

There has been a great deal of work in the last few years on the subject of clothing recognition. However, many works focus on special clothing classes and applications [3, 15, 14, 5, 2] and it is only recently that generic clothing recognition has been directly tackled [16]. The vast majority of the related work, as well as the approach presented here, is based on person detection and pose estimation.

In [5], Gallagher and Chen attempt to jointly solve identity recognition and clothing segmentation. They limit their approach on the torso region, which they segment using graph cuts based on a clothing model learned from one or multiple images believed to be the same person wearing the same clothing. The clothing model of each person utilizes a automatically extracted clothing mask for the torso region. In this work, we extract a *global* clothing prior probability map for the whole body, that is in our case inferred from all training images and clothing classes.

Chen *et al.* [2] focus on attribute learning, *i.e.* learning semantic attributes to describe clothing. They model clothing style rules by a conditional random field on top of the classification predictions from individual attribute classifiers and propose a new application that predicts the dressing style of a person or an event by analyzing a group of photos. Song *et al.* [14] advocate the prediction of human occupations via their clothing. They represent image patches with semantic-level patterns such as clothes and haircut styles and use methods based on sparse coding. As in [5], both the approaches of [2] and [14] focus only on the torso region and therefore cannot be used for generic clothing parsing.

Wang and Ai [15] are interested in images with a large number of people present and propose a novel multi-person clothing segmentation algorithm for highly occluded images. Cushen and Nixon [3] focus on the semantic segmentation of primarily monochromatic clothing and printed/stitched textures. Once again, as in the works described above, only the torso/upper human body part is analyzed and represented in both works.

Recently, Liu *et al.* [8] first introduced the notion of cross-scenario clothing retrieval. In their approach, they use an intermediate annotated auxiliary set to derive sparse reconstructions of the aligned query image human parts and learn a similarity transfer matrix from the auxiliary set to the online shopping set to derive cross-scenario similarities. Their approach is fast, it works however on the more general upper- and lower-part clothing matching and similarity between two pieces of clothing is measured by the number of common attributes. In another very recent work, Liu *et al.*[7] also presented the 'magic closet system' for clothing recommendations. They treat the clothing attributes as latent variables in a latent Support Vector Machine based recommendation model, to provide occasion-oriented clothing recommendation. Both these approaches cannot be applied to finer clothing detection, where a more accurate segmentation of the clothing is needed.

A recent work closely related to ours is the work by Yamaguchi *et al.* [16]. The authors start from superpixels and articulated pose estimation in order to predict and detect the clothing classes present in a real-world image. They also proceed by using the clothing estimates to improve pose estimation and introduce the *Fashionista* dataset that we also use in this work. Since this approach tackles the generic clothing classification problem it is the work that we compare against in the experiments section.

As in [16], an accurate pose estimation provides a good starting point for our algorithm. Bad pose estimations are one of the main reasons behind failure cases (see Figure 2 (right) and Section 5).

3. CLOTHING CLASSIFICATION

In this section, we describe our novel approach for detecting and classifying clothing regions in the query image. For the query image, we start from a successful pose estimation, and present a novel approach to isolate the most promising parts of the image in terms of clothing. We then use segmentation to segment them into visually coherent regions and proceed by clustering the segments to allow spatially distant ones to merge into a single non-connected region (*e.g.* the two sides of an open jacket). To describe the region's location and shape relative to the detected human pose we present a novel binary representation, the *spatial appearance mask*, with which we classify all query regions by proximity to a set of annotated samples from training images. There is no actual learning involved, the training set regions are represented by spatial appearance masks and indexed in an LSH index.

3.1 Articulated Pose Estimation

A popular choice when it comes to clothing recognition is to start from human *pose estimation*. Specifically, as in [16] and [8], we base our approach on the articulated pose estimation algorithm of Yang and Ramanan [17]. Given an image I where a person is successfully detected, we are provided with a set of $N_p = 26$ *body parts*, denoted as $P = \{p_1, \ldots, p_{N_p}\}$, where each p_i represents a square region in the image space. The parts from all detections are ordered in the same way, e.g. the first two parts p_1, p_2 always refer to the person's

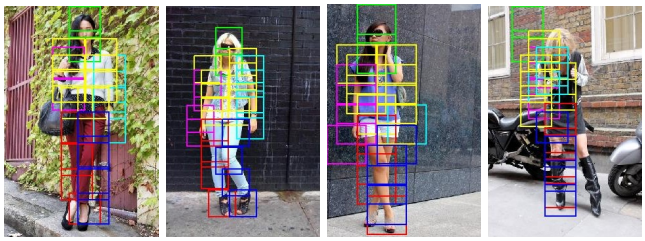

Figure 2: Example results of pose estimation. In many cases, the results are good (as in the first three images); however, there are also some failure cases (as in right-most image).

head area. Figure 2 depicts such pose estimations, with the body parts depicted as colored boxes.

To speed up computations we choose to spatially quantize the body part regions. This quantization step also adds robustness to our representation against small translations, deformations and noise. After scaling all body parts to a fixed width of $w = 78$ pixels, we split each region in a $N \times N$ non-overlapping grid, where each cell is of size $N_c = w/N$. We denote a set of quantized body parts, *i.e.* a set of N_p, $N \times N$ matrices, as $\tilde{P} = \{\tilde{p}_1, \ldots, \tilde{p}_{N_p}\}$. We also define the simple indicator function $I_i(\mathbf{x})$, as a function that is equal to 1 if the region of part p_i contains pixel $\mathbf{x} = (x, y)$.

Since body parts are overlapping, each location can appear in more than one of the parts. The set of parts that $\mathbf{x}$ belongs to is defined as $B(\mathbf{x}) = \{i : I_i(\mathbf{x}) = 1, i = 1, \ldots, 26\}$. Each pixel $\mathbf{x}$ is mapped by a linear function $T_i(\mathbf{x}) : \mathbb{N}^2 \rightarrow \{1, \ldots, N\} \times \{1, \ldots, N\}$ onto one of the cells of the quantized part $\tilde{p}_i$, for each of the quantized body parts of $B(\mathbf{x})$ it belongs to.

The complete quantized pose estimation $\tilde{P}$ can be also expressed as a single $N_p \times N^2$-dimensional vector $\bar{P}$, by first unwrapping each quantized body part $\tilde{p}_i$ as a N^2-dimensional vector and then concatenating all N_p parts. In the rest of this paper we will always use a bar over the corresponding notation to refer to vectorized versions of matrices or sets of matrices. For the rest of this paper, we choose N to be equal to 6 and therefore end up with 936-dimensional vectors after concatenating all quantized body part vectors.

3.1.1 The Global Clothing Prior Probability Map

We want to focus all subsequent steps on image regions that are more likely to contain clothing. We therefore propose to calculate a prior probability map of clothing appearance, normalized on the detected pose estimation regions. As a training set, we need an image collection annotated with clothing segmentation. In the present work we use the Fashionista dataset [16], but any other dataset annotated with clothing may be used. Given a set C of classes $c \in C$, we first create a prior probability map for each clothing class and then accumulate all classes' maps together, normalize them and create a global *clothing prior probability map*, that we denote as $\mathcal{P}$. We will subsequently use this prior map as a binary mask in order to segment only probable human regions for clothing (see Section 3.2). The global clothing prior probability map we calculate is depicted in Figure 3 (right).

To create the map for a given class c, each image pixel $\mathbf{x}_c$, *i.e.* a pixel at location $\mathbf{x}$ annotated with label class c,

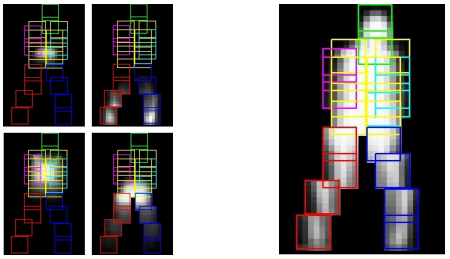

Figure 3: Prior probability maps, quantized and normalized on pose estimation body parts. They are presented here on an arbitrary pose. Left: Probability maps for belt, boots, blouse and skirt (from left to right and top to bottom). Right: The global *clothing prior probability map* $\mathcal{P}$.

casts a vote for this class at every cell $T_i(\mathbf{x}_c)$, for $i \in B(\mathbf{x}_c)$. Accumulating the votes of all pixels annotated with label class c across all images of the training dataset and then normalizing the total sum of each non-empty body part to 1, we end up with a prior probability map $\mathcal{P}_c$ for clothing class c.

The global clothing prior probability map $\mathcal{P}$ is defined as the *union* of all $\mathcal{P}_c, c \in C$. Thresholding $\mathcal{P}$ at a certain probability level π_t we get the binary clothing mask:

$$\mathcal{P}_b = \begin{cases} 0 & \text{where } \mathcal{P} < \pi_t \\ 1 & \text{where } \mathcal{P} \geq \pi_t \end{cases} \quad (1)$$

As any other pose estimation, the quantized maps $\mathcal{P}_c, \mathcal{P}$ and $\mathcal{P}_b$ can also be expressed as a $N_p \times N^2$-dimensional vectors, denoted $\bar{\mathcal{P}}_c, \bar{\mathcal{P}}$ and $\bar{\mathcal{P}}_b$ respectively. A simple approach is to set π_t to 0.5, and therefore $\mathcal{P}_b$ is 1 when the quantized cell is more probable to contain clothing.

Figure 3 (left) shows examples of class probability maps, all presented on an arbitrary pose estimation outline. Without loss of generality, we limit our approach to a single person detection per image in our experiments, keeping the most confident detection returned by [17].

3.2 Segmentation and Clustering

Given a query image I with a successful pose estimation P, we start by applying the binary clothing mask $\bar{\mathcal{P}}_b$ on the detected body parts, and therefore produce an image I_s that is nonzero only in the regions that are more likely to contain clothing:

$$I_s(\mathbf{x}) = I(\mathbf{x}) \times \mathcal{P}_b(T_i(\mathbf{x})) \quad (2)$$

Example of such an image is shown in Figure 4 (left).

We proceed by segmenting the resulting image, using the popular segmentation of Felzenszwalb and Huttenlocher [4]. It is a fast graph-based approach, and the depth of segmentation can be parameterized. After running the algorithm on image I_s, we get an initial segmentation S_{init}, *i.e.* a set of arbitrary shaped segments. Segmentation results are shown in Figure 4 (middle).

It is noticeable that, no matter how lenient the segmentation thresholds are, there are some clothing segments that we need to combine, but are however spatially distant from each other. For example, the segments belonging to the left and right parts of an open jacket (see Figure 4 (left)) could never be merged, although they share the same visual appearance, since the blouse segments appear between them.

Figure 4: Segmentation and clustering steps

To rectify this, we need to somehow further merge the segmentation results, allowing such non-neighboring segments with the same visual appearance to merge. We therefore choose to perform a rapid clustering step on the extracted segments, in the visual feature space. To make the clustering process more meaningful, we set the initial segmentation thresholds low, thus *over-segmenting* the image.

For the clustering process we adopt the recent approach of Avrithis and Kalantidis [1]. *Approximate Gaussian Mixture* (*AGM*) clustering is chosen because it satisfies our two major requirements: it automatically estimates the number of segment clusters K needed on the fly and is very fast. The AGM algorithm is a variant of Gaussian Mixture Models and therefore produces a set of cluster centers μ_k, where $k = 1 \dots K$. Since we do not really need the generalization capabilities of this expressive model, we choose to simply assign each segment s to the centroids with the highest *responsibility* [1].

Clustering needs to be performed on a visual feature space, we therefore extract visual features for each segment s of the initial over-segmentation S_{init} and calculate the corresponding feature vector $f(s)$. The visual features used are discussed in Section 4.1. Features from all segments are given as input to the AGM algorithm and we get *clusters of segments* as output. We then merge all segments that are clustered together, ending up with a set S_K of K segments, where $K < |S_{init}|$. These segments are the tentative clothing regions that we need to classify and determine whether they correspond to an actual clothing piece. Segmentation results after clustering and merging the initial segments are shown in Figure 4 (right).

3.3 Classification

Having robust segments, we now want to detect whether some of them might correspond to a clothing class. Since speed is an issue, we represent the segments as binary vectors and use a multi-probe locality-sensitive hashing (LSH) index to get promising matches sublinearly [6]. Then we proceed with measuring similarity between the query and the results in the top-ranked list and calculate class probabilities by summing overlap similarities. This approach requires no actual learning, just building approximate nearest neighbor structures and is extremely efficient in terms of query time.

3.3.1 Representing the Segments

The approach we follow to represent and classify the segments, derives from the following intuitions: 1) a clothing class is always found at a consistent spatial region both in the query and the training images, when referring to a common normalized pose estimation frame; 2) clothing classes

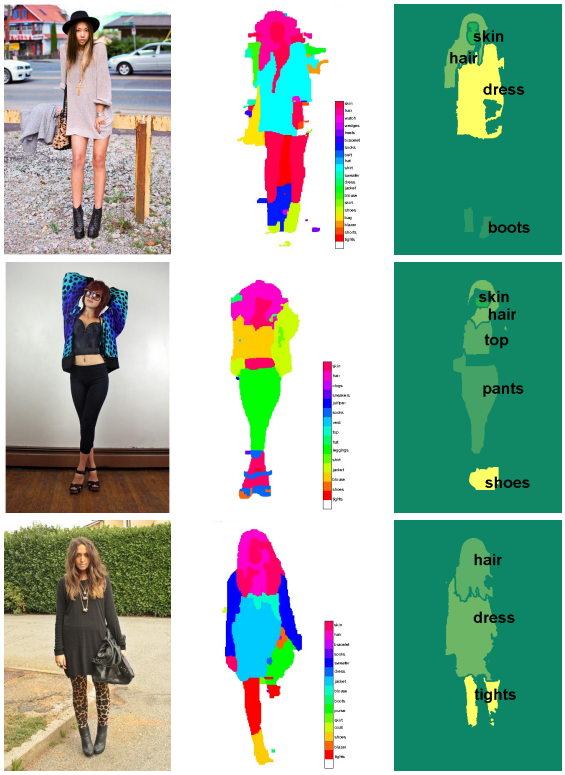

Figure 5: Sample clothing classification results. Left: Original image. Middle: result by [16]. Right: Our result.

cannot be distinguished by color or texture visual features (you can have blue suede shoes,a blue suede skirt and a blue suede hat); 3) basing our approach on shape alone would fail, since clothing classes' boundaries are highly non-consistent among different human poses and viewpoints; 4) we need to choose some representations that facilitate rapid classification for a dataset with up to 10^2 classes.

We therefore propose to use as segment representation a binary spatial appearance mask, projected on the common (normalized) space of the N_p quantized body parts. This will once again result in a $N_p \times N^2$-dimensional vector $\bar{M}_b$ and we are able to represent the segment positions on a normalized frame, without using visual appearance, while at the same time having a flexible binary representation. The segment's shape and size properties are also implicitly captured, as the resulting binary mask is not length normalized.

More precisely, given a pixel at location $\mathbf{x_s}$ belonging to the set of pixels $\mathbf{X}_s$ of a candidate segment s, we can use the functions $T_i(\mathbf{x}_s)$ for $i \in B(\mathbf{x}_s)$ to cast votes at every relevant bin on a—initially empty—quantized body parts set $\tilde{M} = \{\tilde{p}_1, \dots, \tilde{p}_{N_p}\}$. Thresholding $\tilde{M}$ at a certain value γ we get the binary set:

$$\tilde{M}_b = \begin{cases} 0 & \text{where } \mathcal{P} < \gamma \\ 1 & \text{where } \mathcal{P} \geq \gamma \end{cases} \quad (3)$$

We set $\gamma = N_c^2/2$, where N_c is the quantization cell side, *i.e.* the cell will be deemed as active for the mask when at least half of the underlying pixels belong to the segment. The vector corresponding to $\tilde{M}_b$ is noted as $\bar{M}_b$.

3.3.2 Classification Using a Multi-Probe LSH Index

Following the process described in the previous subsections, we have extracted a binary vector $\bar{M}_b$ for every segment that we want to classify. For the training set images the exact clothing segment regions are given, together with their class label. Therefore segmentation and clustering is unnecessary and we only need to follow the process of Section 3.3.1 and extract the normalized binary mask for each segment.

We choose the *Jaccard similarity coefficient* as a similarity measure between the binary vectors. The Jaccard coefficient of sets $\bar{M}_b^1$ and $\bar{M}_b^2$ is denoted $J(\bar{M}_b^1, \bar{M}_b^2)$ and defined as the size of the intersection divided by the size of the union of the sample sets, or in our case binary vectors. Since we need to get reliable matches, *i.e.* matches with significant overlap, we therefore choose to use instead of J the thresholded function $\mathcal{J}^\tau(a,b) = J(a,b)$ iff $J(a,b) \geq \tau$ and 0 otherwise.

Matching each query vector with all training vectors sequentially is a very poor choice in terms of scalability. We therefore choose to first use a sublinear algorithm to get promising candidates and then match the query vector only on the top-ranked list. Therefore, after extracting binary vectors from each clothing segment of the training set, we add the binary vectors of all segments of all classes in a multi-probe LSH index [6], together with their corresponding labels. Since the vectors are binary, *Hamming distance* is set as the dissimilarity measure. Given a query vector $\bar{M_b}^Q$, the LSH index returns the set of the n nearest neighbors of the query from the database, in terms of Hamming distance. The neighbors form a set $\mathcal{N} = \{\bar{M_b}^1, \ldots, \bar{M_b}^n\}$, along with the corresponding class set $\mathcal{C} = \{c_1, \ldots, c_n\}$, where c_1 is the clothing class associated with database vector $\bar{M_b}^1$. We can now define the probability that the segment described by $\bar{M_b}^1$ belongs to class c as

$$P(c|\bar{M_b}^Q) = \frac{\sum_{i=1}^{n} b_c(i) * \mathcal{J}^\tau(\bar{M_b}^Q, \bar{M_b}^i)}{\sum_{i=1}^{n} \mathcal{J}^\tau(\bar{M_b}^Q, \bar{M_b}^i)} \quad (4)$$

where $b_c(i) = 1$ iff $c_i = c$ and 0 otherwise. We classify the query vector to class c^*, where

$$c^* = \begin{cases} \arg\max_c P(c|\bar{M_b}^Q), & P(c|\bar{M_b}^Q) \geq \tau_{class} \\ NULL\ , & P(c|\bar{M_b}^Q) < \tau_{class} \end{cases} \quad (5)$$

Where *NULL* refers to 'no clothing' or 'background' class. Therefore, c^* is the class with the highest probability if this probability is over a threshold, otherwise we mark the segment as *null* or background. It makes sense to use $\tau_{class} = 0.5$ and assign the class if the probability of the class given the region is more than half. For multi-probe LSH, we use the FLANN [10] library implementation. Some classification results are shown in Figure 5.

4. SIMILAR CLOTHING SUGGESTION

Given the clothing classes present in an image, the second step is to retrieve visually similar products from a large product database. As mentioned before, the vast majority of product images are not everyday photographs, but clean photos with the clothing parts in a prominent position and almost always on a white background. We therefore follow a different methodology for product images to accurately grab the actual product region and subsequently describe its visual characteristics. In this section, we first discuss the visual features used to describe a region, either coming from a product or a tentative query image segment. We then present our product image parsing methodology and finally the indexing system used for retrieval.

4.1 Visual Features

We describe the visual attributes of a region by capturing *color* and *texture* characteristics. For product images, feature extraction can be done offline and thus extraction speed is not a big issue. However, at query time we need to extract visual features for many segments, *i.e.* for every one of the initial segmentation S_{init}, and require visual features that can be rapidly extracted. We therefore stick to simple descriptors rather than using more sophisticated ones, *e.g.* like SIFT [9].

To describe color we quantized the RGB values of each pixel to a uniform color dictionary of 29 elements. For texture we used *Local binary patterns* (LBPs) [11] and specifically just calculate the 9-dimensional histogram of *uniform patterns*. As argued in [11], uniform patterns are in fact the vast majority, sometimes over 90 percent, of the 3×3 texture patterns in surface textures.

We l_1-normalize color and texture features independently and then concatenate them to get the final 39-dimensional descriptor.

We also experimented with including a skin detector in the feature set. We tried to concatenate some skin pixel statistics as extra dimensions in the feature vector and this did not improve performance, therefore skin information was not included as a feature. A possible reason for this is that skin-like color clothing parts are no so infrequent in the datasets. We used skin detection when parsing product images, however, since cases of product image with visible body parts of a human model are frequent (see Section 4.2).

4.2 Describing Product Images

To accurately segment the region of the product in the product database images, we use the *grabcut* algorithm [12]. We initialize the algorithm using the whole image as bounding box and run for a few iterations. In most cases we are actually able to get the product's region without keeping background information. To avoid extracting noisy features from the pixels on the boundary, we also filter the binary region area with a small morphological erosion filter of size 3×3. Product images and the segments kept after grabcut are depicted in Figure 6.

The grabcut algorithm extracts the prominent foreground object, so in cases where the clothing product is worn by a person, it will include the model's region too. To avoid keeping person pixels as product pixels, we use skin as a mask and keep only non-skin pixels as the product's region. Since in most cases only a part of the model-person is shown in the product image, *e.g.* only legs are visible in skirt images or torso in blouse images, we cannot use full pose estimation.

For all pixels that remain active after eroding and applying the binary skin mask, we extract color and texture features, as described in Section 4.1. This way, each product is represented by a feature vector comparable to the one extracted by a query image region.

4.3 Indexing and Retrieval

Figure 6: Sample pairs of product images (left) and the regions kept for feature extraction after grabcut (right— background depicted with black color).

After having one feature vector per image for the whole product collection, we need to be able to quickly retrieve such vectors that are similar to a query region vector. Since feature vectors are not sparse, inverted indexes are not efficient in our case. We instead choose to use a fast approximate k-nearest neighbor index, using a forest of *randomized kd-trees* [13]. Once again we used the FLANN library [10] as the ANN implementation.

As the query feature vectors are also classified, we only need to search for similar product *within* that class. We therefore create and store in memory indexes *per class* and at query time only query the corresponding index for each clothing region. For each search, we get the k most similar products and then threshold the distance to actually display only the results that are visually very close to the query. In cases where no products are below the distance threshold, that specific query region is discarded.

The aforementioned approach can easily scale to millions of products on a singe machine. Indexes for 1 million product images can fit in around 450MB of memory. Retrieval speed is determined by the number of leaves checked by the ANN structure and query time per region in our experiments was in the order of a few milliseconds.

5. EXPERIMENTAL EVALUATION

We use two datasets for evaluating the proposed approach. For experiments on clothing class detection we use the publicly available annotated part of the *Fashionista* dataset[1] presented in [16]. It consists of 685 photos real-world photos with a human model in a cluttered but prominent position. The ground truth of this dataset contain 53 different clothing labels, plus the labels *hair*, *skin*, and *null*. For the large scale clothing suggestion scenario, we have crawled a product database of approximately 1.2 million products from Yahoo Shopping[2]. The products we crawled are all related to the clothing classes that we detect.

For the clothing class detection experiment, we follow the protocol of [16]; we measure *pixel accuracy* per image and then compute the mean across all query images. Standard deviation is also reported in the cases where 10-fold cross validation is used. As baseline we set the pixel accuracy under the naive assumption of predicting all regions to be background, again as in [16].

[1] `http://www.cs.sunysb.edu/~kyamagu/research/clothing_parsing/`

[2] `http://shopping.yahoo.com/`

k_init	$\lambda_{AGM} = 0.20$	$\lambda_{AGM} = 0.22$	$\lambda_{AGM} = 0.23$
200	36.3(22.9)	36.3(20.4)	36.3(14.2)
400	29.9(21.6)	29.9(19.1)	29.9(13.9)
500	27.2(20.3)	27.2(18.8)	27.2(13.7)
700	20.8(16.5)	20.8(14.9)	20.8(13.2)

Table 1: Average number of initial(final) segments for different parameter combinations

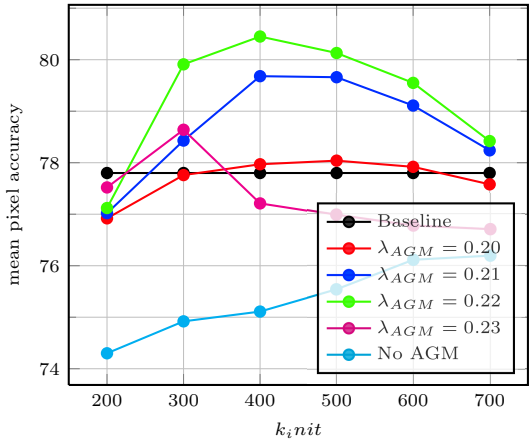

Figure 7: Mean pixel accuracy for different values k_{init} of the threshold function for the initial segmentation [4] and different expansion factor λ_{AGM} values for the subsequent segment clustering [1].

5.1 Parameter Tuning

For parameter tuning, we use a small random sample of 50 images as validation set and leave the rest as training.

The first parameter to tune is the threshold for the initial segmentation [4]. We present results when varying the value k_{init} of the threshold function. Smaller value yields more segments. The rest of the initial segmentation parameters are $\sigma = 0.5$ for smoothing and $k_{min} = 300$ as the minimum component size. The Approximate Gaussian Mixture (AGM) clustering algorithm [1] has two parameters: the expansion factor λ_{AGM} and the merging threshold τ_{AGM}. We set $\tau_{AGM} = 0.55$ as suggested and vary the expansion factor λ_{AGM} to get various clustering results. The smaller λ_{AGM} is, the more final segments will be kept. We run AGM for a fixed number of 10 iterations.

The time needed, after pose estimation, for the extraction of the final segments is on average 700msec, of which 261msec is on average the time needed for the initial segmentation, 30.4msec for the AGM clustering algorithm and the rest for visual feature extraction. Figure 7 plots mean pixel accuracy for different values of k_{init} and the expansion factor. The average number of initial and final segments for different parameter combinations is shown in Table 1.

The best combination of values from this tuning experiment are $k_{init} = 400$ and $\lambda_{AGM} = 0.23$. The first one favors over-segmentation and is slightly smaller than the values suggested by [4].

Method	mean pixel accuracy	average time
[16]	80.7	334 sec
Ours	80.2 ± 0.9	5.8 sec
Baseline	77.6 ± 0.6	—

Table 2: Cross-validated results on the Fashionista dataset. Performance in terms of mean pixel accuracy and timings in terms of average detection time per image are presented.

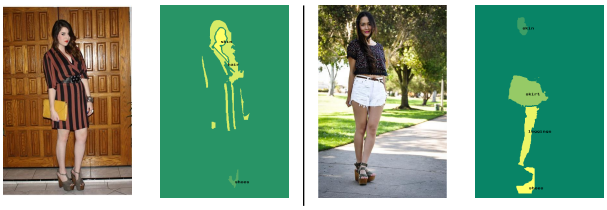

Figure 8: Sample failure cases. Left: Stripes distort the segmentation. Right: Common misclassifications, *e.g.* skin for leggings or shorts for skirt.

5.2 Clothing Class Detection

As mentioned above, we evaluate the performance of the proposed approach following the protocol of the Yamaguchi et al.[16], using a 10-fold cross-validation on the Fashionista dataset. Performance and time results, averaged over all folds are presented in Table 2 for the proposed approach, for the baseline and for [16]. The proposed approach provide results comparable to [16] and does so in *only a small fraction of the time that [16] needs.* Their approach requires over 5 minutes to parse a photo[3] in the unconstrained case, *i.e.* the general case where there is no prior knowledge of the clothing present in the query image, while ours requires only 5-6 seconds. Moreover, since the application we care about is clothing suggestion, we are far more interested in *precision* of the detections than recall. [16] focuses a lot on recall of the garments and usually yields many of false positives in the unconstrained case. This is visible in Figure 5, where the number of garments that the [16] detects is far greater than the true number.

Failure cases of the proposed approach usually start from an erroneous pose estimation. Also, clothes with periodic color changes (*e.g.* stripes) pose a big challenge for our approach due to the initial color segmentation, and are usually cases that clustering cannot rectify. Finally, the misclassification for highly overlapping classes (*e.g.* pants and jeans or blouse and tops) is another main source for failure cases. Figure 8 depicts some failure cases.

5.3 Large Scale Clothing Suggestion

Our motivation behind clothing detection has been the automatic suggestion of clothing. To experiment and evaluate this scenario in a large scale environment, we developed a web-based application. Given a real-world image as query, the application detects the clothing classes present and returns related clothing products for each class to the user.

Our 1.2 million product image database is a subset of Yahoo! Shopping and is loosely annotated by clothing *category.*

[3]Time is measured when running the code provided by the authors of [16].

Clothing category	Proposed (%)	Random (%)
Dress	68	10
Skirt	59	2
Blouse	37	4
Top	55	6
Jackets & coats	43	3
Pants & Jeans	69	12
Boots	66	14
All	**54**	8

Table 3: User evaluation results for our suggestions and random suggestions. We present average precision at 10 results, for some sample clothing categories and for all categories.

pose	segmentation	classification	retrieval	total
1.7sec	0.7sec	3.2sec	0.3sec	5.8sec

Table 4: Time analysis for our clothing suggestion approach.

Since the categorization of the shopping catalog is different from the one we used for clothing detection, some of the initial classes had to be merged together (*e.g.* jackets and coats both belong to a single product category). Figures 1 and 9 depict results of our clothing suggestion application. The query image is shown on the left, and the suggested relevant clothing is on the right.

Given that there is no annotated clothing product database available we are unable to evaluate clothing suggestions directly and perform a user evaluation instead. We modified our online interface and presented each user both clothing suggestions from our algorithm and random products from the detected clothing catagories, mixed in random order. We then asked them to say whether each suggested product is visually relevant or not to the clothes worn in the image. We collect 178 image annotations from 11 users. Results in terms of precision are presented in Table 3. Users noted on average more than half of the suggested items as relevant while for the random products typically less than 10% were relevant.

Table 4 presents the time analysis for the different stages of the proposed clothing suggestion approach. We run all our experiments on a single quad-core machine and the code is implemented in MATLAB and C++. Since many computationally heavy parts of the approach are performed independently on different regions in the image (*e.g.* visual feature extraction, region classification and retrieval) the total query time can be further decreased by parallelization.

6. DISCUSSION

In this paper we present a fully automated clothing suggestion approach that can trivially scale to hundreds of product classes and millions of product images. We achieve clothing detection performance comparable to the state-of-the-art on a very recent annotated dataset, while being more than 50 times faster. Moreover, we present a clothing suggestion application, where the product database contains over one million products. The resulting suggestions might be used as-is, or they could be incorporated in a semi automatic annotation environment, assisting human annotators and significantly speeding up this costly process.

Figure 9: Clothing suggestion results.

As future work, a finer model that takes into account clothing class *co-occurrences* in the training set can boost detection performance, while also help mis-classifications. For distinguishing intra-class variations and styles (*e.g.* long to short sleeve blouses) attributes can be a good solution [2]. Gender classification on the detected people can also boost performance. By generalizing our model, we can include both attributes and gender information for the clothing class detection and suggestion stages of our approach.

More figures for all stages of the proposed approach can be found in the supplementary material.

7. REFERENCES

[1] Y. Avrithis and Y. Kalantidis. Approximate gaussian mixtures for large scale vocabularies. In *ECCV*, 2012.

[2] H. Chen, A. Gallagher, and B. Girod. Describing clothing by semantic attributes. In *ECCV*, 2012.

[3] G. Cushen and M.S. Nixon. Real-time semantic clothing segmentation. In *ISVC*, 2012.

[4] P.F. Felzenszwalb and D.P. Huttenlocher. Efficient graph-based image segmentation. *IJCV*, 59(2):167–181, 2004.

[5] A.C. Gallagher and T. Chen. Clothing cosegmentation for recognizing people. In *CVPR*, 2008.

[6] A. Gionis, P. Indyk, R. Motwani, et al. Similarity search in high dimensions via hashing. In *VLDB*, 1999.

[7] S. Liu, J. Feng, Z. Song, T. Zhang, H. Lu, C. Xu, and S. Yan. Hi, magic closet, tell me what to wear! In *ACM Multimedia*, 2012.

[8] S. Liu, Z. Song, G. Liu, C. Xu, H. Lu, and S. Yan. Street-to-shop: Cross-scenario clothing retrieval via parts alignment and auxiliary set. In *CVPR*, 2012.

[9] D.G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.

[10] Muja M. and Lowe D.G. Fast approximate nearest neighbors with automatic algorithm configuration. In *VISSAPP'09*, 2009.

[11] T. Ojala, M. Pietikainen, and T. Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *PAMI*, 24(7):971–987, 2002.

[12] C. Rother, V. Kolmogorov, and A. Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. In *TOG*, volume 23, pages 309–314, 2004.

[13] C. Silpa-Anan and R. Hartley. Optimised kd-trees for fast image descriptor matching. In *CVPR*, pages 1–8. IEEE, 2008.

[14] Z. Song, M. Wang, X. Hua, and S. Yan. Predicting occupation via human clothing and contexts. In *ICCV*, 2011.

[15] N. Wang and H. Ai. Who blocks who: Simultaneous clothing segmentation for grouping images. In *ICCV*, 2011.

[16] K. Yamaguchi, M.H. Kiapour, L.E. Ortiz, and T.L. Berg. Parsing clothing in fashion photographs. In *CVPR*, 2012.

[17] Y. Yang and D. Ramanan. Articulated pose estimation with flexible mixtures-of-parts. In *CVPR*, 2011.