Dress.com: A CBIR-Based Digital Paper Doll System for Real-Life Dressing

Chueh-Min CHENG and Chun-Wei LIU

***Abstract*--We present the "Dress.com" system, a digital paper doll system which helps users to setup virtual wardrobes mirroring their real-life ones. Once setting up, the system allows users to virtually change their clothing by simply take a picture of target clothing item. A long-term goal of the system is to help users to find appropriate clothing collocations.**

***Index Terms*—Home computing, CBIR system, dressing.**

INTRODUCTION

DECIDING what to wear today is not an easy task. Sometimes it takes hours to try nearly all the shirts in our wardrobes just to find some perfect matches with our new-bought skirts. The problem comes out because we never know how we look like in those clothing items unless we put them on. And it takes time and efforts to physically change clothing.

If we observe one traditional game of girls', the paper doll game as shown in Fig. 1, we can find out some facts. First, the paper doll game is also about clothing collocations. Girls pick

Fig. 1. A piece of traditional paper doll. Note that the images of paper dresses listed on the right are shown as if the dresses are worn by the paper idol on the left. If we move a paper dress onto the paper idol, the idol will look like wearing the chosen dress.

Manuscript received January 19, 2009.

Chueh-Min Cheng is with the Computer Science and Information Engineering Department, National Taiwan University. (e-mail: realjamie@cmlab.csie.ntu.edu.tw).

Chun-Wei Liu is with the Computer Science and Information Engineering Department, National Taiwan University. (e-mail: dreamway@cmlab.csie.ntu.edu.tw).

out one shirt on the paper broad in order to match the skirt on the paper idol. Second, all the printed images of clothing items belong to the paper idol are actually the images of items which are worn by the idol. Therefore, when we put a paper shirt unto a paper doll's body part, it looks just like the idol is in the shirt. We borrow this idea to solve the problem stated above: trying on clothing is time-consuming. Our system first ask the user to setup a digital clothing database, namely, a virtual wardrobe, in which all the clothing items are the images of the items worn by the user. When all the clothing items in user's real-life wardrobe are set up as image items in the virtual wardrobe, the user can easily see all possible clothing collocations without physically changing clothing. This part will be more described in section 2.

However, another problem arises when the number of image items in the virtual wardrobe increases: how to efficiently search a specific item among all of them? One possible solution is to ask users to tag each clothing item with texts, that means, classify each item to proper category. Then, when we want to find a specific clothing item, we just trace all the way down the category tree we built up. This is the way mostly operated on recent online auction platforms such as *eBay* and *Yahoo*. The idea is quite simple and easy to implement, nevertheless, it's hard for users to give a proper tag to an item. If the category tree is not well-built, it could take much time to find the specific item we want. Here we address another solution to search: Content-Based Image Retrieval (CBIR) method. Nowadays, there have been some next-generation image search engines such as *like.com* and *GazoPa.com* which apply CBIR methods to search images by images. Though CBIR method sometimes makes mistakes, we find that it has good performance on searching among a medium size clothing database, as the similar wardrobe size which most people have. Therefore, it seems like CBIR is a good method to our problem. The algorithms we used and the performance are further described in section 2 and 3.

TECHNICAL IMPLEMENTAION

Setting up Clothing Database

To setup the clothing database, namely, the virtual wardrobe, our system first asks the user wearing the target clothing item to take a picture. After this, the system applies *LazySnapping* [3], which is one kind of *GraphCut* algorithms to extract the target item for background.

LazySnapping method first asks the users to interactively specify some foreground and background pixels. After this, classify foreground and background pixels to 64 clusters $\{K_n^F\}$ and $\{K_n^B\}$ by K-means method. Next, compute energy function E(X):

$$E(X) = \sum_{i \in V} E_1(x_i) + \lambda \sum_{(i,j) \in \varepsilon} E_2(x_i, x_j) \ ,$$

E_1 is the *likelihood energy* and E_2 is the *prior energy*, which can be further referred in [3]. Next, apply max-flow *GraphCut* to separate foreground pixels with background pixels.

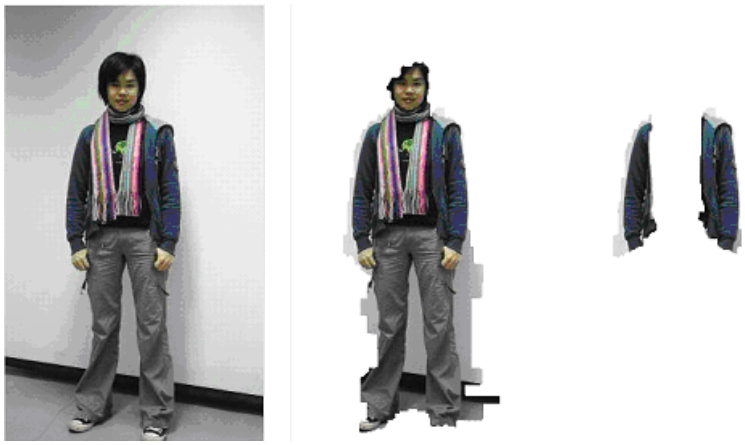

Fig. 2. Applying Lazy Snapping to extract the clothing item and build database. Our system ask the user to take a picture of her dressing (left) and draw some strokes of lines to specify foreground and background pixels. After applying Lazy Snapping, the system first extracts the user from background (middle). Again, the user draws some stroke and extracts her coat from her image (right).

Fig. 2 shows the *GraphCut* results during applying iterative *Lazy Snapping*. We can see that the *GraphCut* methods are suffering from the quality of the picture, the lighting in the environment, and the contrast between the colors in the foreground and those in the background. When the colors on the clothing item we want to specify is similar to those in the background, it

is hard to get a perfect cut of the item. And this is a problem we faced.

Searching for Specific Clothing Item

Given a picture of a clothing item, our system first extracts it from background as described above. After this, the system gets a 48-dimensional feature vector of the foreground item. In order to take both color and texture into account in the searching process, we use fusion of a 48-dimensional texture feature vector and a 24-dimensional color feature vector as our feature vector.

Gabor filters are applied to extract local texture features. Given an input image $I(x, y)$with $(x, y) \in \Omega$(the set of image points), it is first converted to be a gray-level image with a normalized size. It is then convolved with a 2-dimensions Gabor function $g(x, y)$, and we get a Gabor image $r(x, y)$ as follows:

$$r(x, y) = \iint_{\Omega} I(\xi, \eta)\, g(x - \xi, y - \eta) d\xi d\eta \quad (1)$$

where $g(x, y)$ used here is the Peter Kovesi [2] version.

To obtain 24 Gabor images *R*, we set the number of scales to 4 and the number of orientations to 6. Next, we pick the mean and standard deviation of each image as features, so we totally have 48 dimensions of texture vectors T(d):

$$T(d) = \begin{cases} \text{mean}, & d = 1 \bmod 2 \\ \text{standard deviation}, & otherwise \end{cases} \quad (2)$$

To further search with similar color, the original image is converted to *HSV* color space to get the color feature vector. We set the HSV color space to 24 bins by dividing the hue channel into 16 parts and both the saturation and lightness channels into 4 parts. By doing so, we enlarge the weighting on the hue channel. Then, we construct the color histogram and normalize in each bin. Therefore, we have a 24-dimensional color feature vector C(d), where d is the index of dimension.

Finally, we fuse the color vector F(d) as follows:

$$F(d) = \begin{cases} T(d) + \lambda C(\lceil d/2 \rceil), & d = 1 \bmod 2 \\ T(d), & otherwise \end{cases} \quad (3)$$

where d is the index of dimension and λ is the weighting of the color vector.

We once considered to use SIFT detectors to detect for feature points and SIFT feature vectors to match for those points. However, corner detectors are suffering from the changes of folder and shadows on the clothing items. Thus, SIFT method doesn't perform well in our case.

Putting Clothing Item on the Idol

After finding out the specific clothing item in the database, we still need to put the item to the right place on the digital idol. Therefore, we have to do some alignment. We found some references on face alignment and face recognition, but still have no clear idea about how to align clothing items. Possible solution is to apply *Local Binary Pattern* method [4] to detect whether the item is a shirt or a skirt and place it to the right place onto the idol. We keep this part as our future work.

System User Interface

One of the authors tried this system. She first took a picture of her (facing front, from head to toe) and set it to be the idol image. Then, she setup the virtual wardrobe by keep taking pictures of her everyday dressings for several days. Finally, she collected 2 coats, 2 mufflers, 1 T-shirt, 1 sweater, and 2 dresses, and 1 skirt.

In Fig. 3 we can see that our system shows the image of the user (the "paper idol") and a list of some images of her clothing items in the virtual wardrobe. On the bottom of the list are a plus icon and a search icon. The user can add more clothing items at anytime. Our system will lead the user to go through the *GraphCut* process. When the user presses the icons, the corresponding

clothing item will be put onto the paper doll. As stated before, our system also provides the CBIR search for the user to find specific item.

After the small wardrobe was been built, we tested this CBIR system by using the pictures of clothing items at hands to search for the clothing item image built in the digital virtual wardrobe. In most of the cases, the CBIR system successfully retrieves the exact item. The outcome reflects our system really works. However, it takes about 3 minutes for every single search, and it's quite a long time for an interactive system.

Fig. 3. The "Dress.com" system. On the right is the image of the user, worked as the "paper idol." There is a list of clothing items in her wardrobe on the left, which is interactively set up by the user. On the bottom of the list are a plus icon and a search icon. The plus icon allows the user to add more items to her virtual wardrobe. And the search icon provides the CBIR search to find specific item.

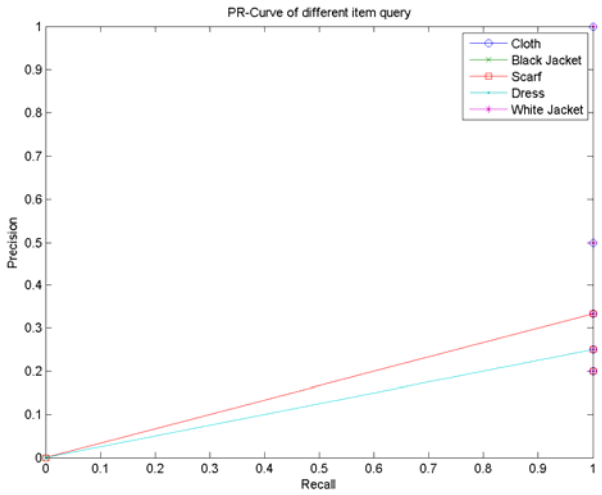

Fig. 4. The PR curve of the CBIR method applied in the"Dress.com."

Fig. 4 shows the PR curve of the CBIR method applied in our system. The result seems to be perfect for most of the cases, but it is possibly because of the small size of our system. We will

enlarge the size and do further research on our system.

ACKNOWLEDGMENT

Our system would not be possible without efforts of the CMLab in NTU. We thank Dr. Winston H. Hsu and Kuang-Ting Chen for their instructions and suggestions on developing our work. Also thank Dr. Ming Ouhyoung, Dr. Yung-Yu Chuang, Dr. Bing-Yu Chen, and Dr. Hao-Hua Chu for their ideas and guides.

REFERENCES

[1] S.E. Grigorescu, N. Petkov and P. Kruizinga, "Comparison of texture features based on Gabor filters," *IEEE Transaction on Image Processing*, vol. 11, pp. 1160-1167, 2002

[2] Peter Kovesi, "GABORCONVOLVE – function for convolving image with log-Garbor filters," http:// www.csse.uwa.edu.au/ ~pk/ Research/MatlabFns/.

[3] Y. Li, J. Sun, C.-K. Tang, and Heung-Yeung Shum. Lazy Snapping, ACM SIGGRAPH 2004.

[4] Timo Ahonen, Abdenour Hadid, and Matti Pietik¨ainen, "Face Description with Local Binary Patterns: Application to Face Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Volume 28 , pp. 2037 – 2041, 2006

Chueh-Min Cheng received her B.S. degree in 2007 from the Computer Science and Information Engineering Department of National Taiwan University, Taipei, Taiwan. Now she is pursing her M.S degree in the same department.

Chun-Wei Liu received his B.S. degree in 2008 from the Computer Science Department of National Chiao Tung University, Hsinchu, Taiwan. Since September 2008, he has been pursing the M.S degree in Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan.