Virtual Clothes Try-on Using Computer Vision Techniques

Hwang, Sunyoung
Seoul National University
0515hwang@snu.ac.kr

Kim, Hyunsoo
Seoul National University
hscornelia@snu.ac.kr

Lee, Minjae
Seoul National University
herry123435@snu.ac.kr

Kim, Kyoungseo
Seoul National University
dooly9931@snu.ac.kr

Abstract

This paper describes a virtual clothes try-on program that exploits computer vision techniques to achieve further robustness on a variety of human posture input and naturality of try-on result image. First, obtain clothes segments divided as body parts with Cross-Domain Complementary Learning. Second, designate correspondence points between clothes segments and body parts using contour and corner detection. Last, conduct TPS warping and gaussian pyramid blending with ordered correspondence points to get the result. We could obtain proper correspondences for some posture inputs, and with manually designated points for arbitrary inputs, the results look natural as if the model really wore the clothes. However, handling arbitrary postures still remains as future work.

1. Introduction

As Internet shopping has become more active, online shopping malls have started to provide virtual try-on programs that allow you to try on clothes without visiting offline stores. However, most early versions of programs just overlap prepared outfit shots in front of the human image. These shots primarily depend on the body proportion of models who wear the clothes, so it is hard to know how the outfit would actually look on the customer. Moreover, customers could only take one pose because prepared clothes images are static.

In this project, our purpose is to code a program that whenever we input a clothes image and a posture photo that we want to check outfit, returns image that looks as if the customer taking the pose tried the clothes on as an output. The program's first step is segmenting body parts into torso, left and right upper arms and forearms, thighs and calves. And finding corresponding points in each segment. We used two kinds of methods in this procedure, and compared them. In the first method, we detected every edge and corner in a clothes image and posture photo using classic computer vision knowledge. And divided images based on them. In the second method, we used deep learning techniques to segment images. And then found corners of each segment. After that, warp and blend each clothes segment into a body posture image using corresponding points of each segment. By this, we can implement the elasticity of clothes and respond to various poses.

Although this could be also implemented with machine learning techniques, this project can suggest an unconventional approach on this topic, using computer vision techniques mainly.

2. Background and related work

2.1. VITON: An Image-based Virtual Try-on Network

VITON is an image-based Virtual Try-on Network without using 3D information in any form, which seamlessly transfers a desired clothing item onto the corresponding region of a person using a coarse-to-fine strategy. [1]

A coarse sample is first generated with a multi-task encoder-decoder conditioned on a detailed clothing-agnostic person representation. The coarse results are further enhanced with a refinement network that learns the optimal composition. We conducted experiments on a newly collected dataset, and promising results are achieved both quantitatively and qualitatively. This indicates that this 2D image-based synthesis pipeline can be used as an alternative to expensive 3D based methods.

Figure 1: Qualitative comparisons of different methods.

Figure 1 presents qualitative results that is a visual comparison of different methods. VITON accurately and seamlessly generates detailed virtual try-on results, confirming the effectiveness of a framework.

Quantitative results are derived based on Inception Score [2] and a user study. Models producing visually diverse and semantically meaningful images will have higher Inception Scores. Inception Scores of PRGAN, CAGAN, CRN, and VITON are 2.688 ± 0.098, 2.981 ± 0.087, 2.449 ± 0.070, and 2.514 ± 0130. User study is conducted on the Amazon Mechanical Turk(AMT) platform. The percentage of trials in which one method is rated better than other methods is adopted as the Human evaluation metric. The percentage of PRGAN, CAGAN, CRN, and VITON is 27.3%, 21.8%, 69.1%, and 77.2%. VITON also obtains a higher human evaluation score than state-of-the-art generative models and outputs more photo-realistic virtual try-on effects.

Our topic is similar to this paper, but we exploited computer vision techniques such as contour detection, corner detection, image warping, and image blending for better functionality of the program.

2.2. Functions used from opencv

In order to successfully implement body part corner detection using plain computer vision techniques, several functions from the opencv library were used. Here are the breakdown for the 3 most often used functions from our project:

- findContours(…, mode, method, …)

This function is used to get the contour of the input image. It is implemented based on the border following method. [3] The border following algorithm first derives a sequence of coordinates from the border between a connected component of 1-pixels and a connected component of 0-pixels (background or hole). Then, outer borders (1-pixels) are separated from hole borders (opposing borders of outer borders) and then using the topological structural analysis of digitized binary images, hierarchy between the found borders are detected.

The "mode" argument of the opencv function determines the borders and the form in which they are returned. Whilst there are several different retrieval mode flags provided in opencv, only the RETR_TREE (returns all outer and hole borders with their corresponding hierarchy), and the RETR_EXTERNAL flag (returns the extreme outer border only) are used in our project.

- goodFeaturesToTrack()

This function returns the corners detected in the input image. For each pixel, the corner quality is measured using either the Harris scoring function or the Shi-Tomasi scoring function. The biggest flaw of the Harris scoring function is that it depends on a parameter k that must be tuned for each image. This flaw's effect becomes even more accentuated in our project, because we must use the same parameters for all input images. This is why in our project, we used the Shi-Tomasi algorithm instead of the Harris corner detection algorithm.

- connectedComponentsWithStats()

Due to the wrinkles formed in clothes, the previous function often detects corners that are grouped together in the same area. This can lead to several complications in later steps, which is why it was essential to group similar points together. One method used in our product was using the connectedComponentsWithStats() function, which finds connected components (a group of points that form "blob"-like regions in images and have high connectivity) and returns their centroid.

2.3. Human body segmentation using deep learning

Cross-Domain Complementary Learning learns a neural network model for multi-person part segmentation using a synthetic dataset and a real dataset. Real and synthetic humans share a common skeleton structure. During learning, the proposed model extracts human skeletons which effectively bridges the synthetic and real domains. Without using human-annotated part segmentation labels, the resultant model works well on real world images. [4]

2.4. Joint detection using deep learning

Realtime multi-person 2D pose estimation is a key component in enabling machines to have an understanding of people in images and videos. In this work, they present a real-time approach to detect the 2D pose of multiple people in an image. The proposed method uses a nonparametric representation, which they refer to as Part Affinity Fields (PAFs), to learn to associate body parts with individuals in the image. This bottom-up system achieves high accuracy and real-time performance, regardless of the number of people in the image. In previous work, PAFs and body part location estimation were refined simultaneously across training stages. They demonstrate that a PAF-only

refinement rather than both PAF and body part location refinement results in a substantial increase in both runtime performance and accuracy. They also present the first combined body and foot keypoint detector, based on an internal annotated foot dataset that they have publicly released. They show that the combined detector not only reduces the inference time compared to running them sequentially, but also maintains the accuracy of each component individually. This work has culminated in the release of OpenPose, the first open-source real-time system for multi-person 2D pose detection, including body, foot, hand, and facial keypoints. [5]

2.5. ThinPlateSpline Warping

TPS(thin-plate spline) warping is a nonlinear warping method. It is similar to mathematically modeling a thin metal spline being twisted by external force. While Perspective warping requires at least 4 correspondence points, TPS only requires 3 correspondence points for the convergence of warping. [6] Since fibers of clothes are twisted and bent easily by small changes in human body pose, this technique can play an important role in this work. createThinPlateSplineShapeTransformer is used, which is an implementation of TPS in opencv of python.

3. Methods

In order to implement the virtual try-on, we must process human images (body below) through two main steps: finding correspondence points, and warping the images using the found points. In our project, we initially aimed to find the correspondence points without using to any deep learning methods (CV method). However despite the usage of multiple heuristics, the results were not good enough to obtain generally acceptable warping images, which is why we resorted to using deep learning (DL method). We did, however, restrict deep learning usage to the minimum.

3.1. Corner detection & image segmentation (CV)

The CV method is extremely sensitive to noise in images (people wearing striped clothes, hands close to the side…) and thus makes corner detection a nearly impossible task. This is why for the CV method, we restricted input images to images of people where the arms and legs are fairly apart (ex: hands on waist, hands wide apart…), so that the corners-to-detect are clearly distinguishable.

3.1.1 Approach 1: Find edges and intersection points

The first approach opted for detecting corners and segmenting the body was to find all of its edges, and find their intersection points. The following are the steps:

1. Find the outer contour of the body using findContours (mentioned in section 1.2.2)
2. Find the edges using HoughLines algorithm and suppress lines that are similar (studied in class)
3. Group lines into horizontal and vertical lines
4. Find intersection points

Grouping the lines into two types uses the k-means clustering algorithm on the lines' angle. Intersection points were found by solving the linear equation formed using the polar coordinates:

$$\begin{pmatrix} \cos(\theta_1) & \sin(\theta_1) \\ \cos(\theta_2) & \sin(\theta_2) \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} \rho_1 \\ \rho_2 \end{pmatrix}$$

The results for each step are as follows:

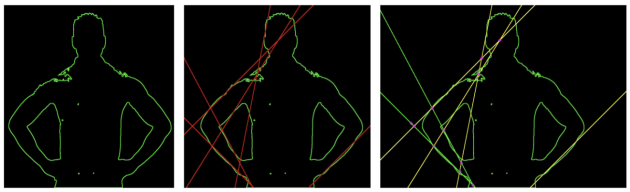


Figure 2: result images of each step of algorithm

As we can see from above, HoughLines was not able to find sufficient lines to find intersection points to cover all the body parts. Moreover, several non-necessary intersection points were discovered, and discerning them from the necessary ones was nearly impossible when the only information available was the contour of the body.

3.1.2. Approach 2: Refine Shi-Tomasi corner detection algorithm

In this approach, we decided to find the corners of the body using the aforementioned Shi-Tomasi corner

detection algorithm. The steps for this approach are as follows:

1. Find outer contour using findContour()
2. Find corners using goodFeaturesToTrack with the Shi-Tomasi algorithm.
3. Manually handpick the corner points.
4. Cut out each part of the body from the original image
5. Smooth out the segmented parts to obtain a more natural segmentation.

The results for each step can be seen in the following figures:

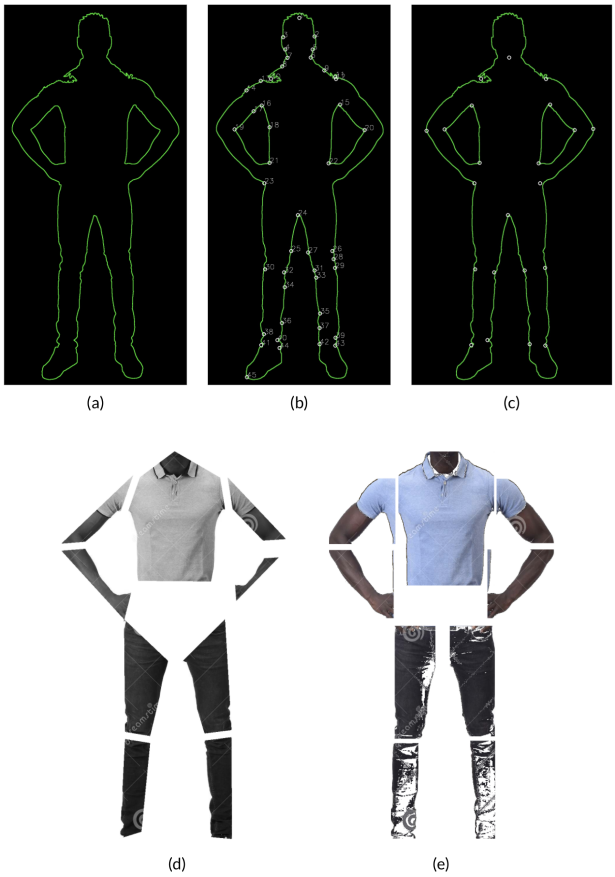


Figure 3: result images for each step of the algorithm

From the resulting image, we can conclude that corner detection yielded fairly good results, except that it requires manual intervention. This manual intervention was absolutely necessary as selecting target correspondence points from all the corners detected without any further information is impossible to achieve for general images. We can also see from image (e), that the segmented images sometimes give unnatural results (ex: left upper-arm segment image contains a part of the upper body).

Lastly, we noticed that while the algorithm may work fairly well on some images, it might yield mediocre results on other images, meaning that there are discrepancies in the algorithms' results. This is mainly because the computer vision techniques used in the algorithm are based on the images' proper features (intensity of colors, image resolution, gradient of intensity…) and a large amount of parameters that require special tuning for each image.

3.2. Corner detection & image segmentation (DL)

The results of image segmentation using computer vision techniques wasn't natural. So we applied a deep learning method to segment images. The methods of corner detection are equal to previous methods.

3.2.1. Image segmentation

We used a pre-trained model of Cross-domain complementary learning. for body part segmentation. First, build and run docker images to segment our clothes and human image. Then put pre-trained model weight, and inference images to 15 parts of the body. The results are png format of 15 images. These images will be used to detect corners for warping segments to the human image.

Figure 4: The result of image segmentation using DL

3.2.2. Corner detection

First, we used Harris corner detection to detect corners from segmented images. This approach is done using the

following steps. First calculate intensity using a small window. The point with large intensity is a corner.

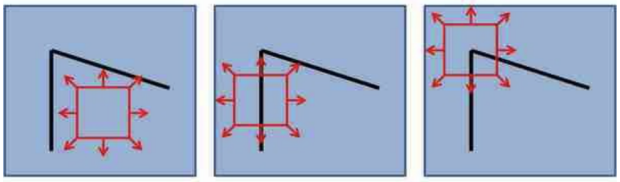
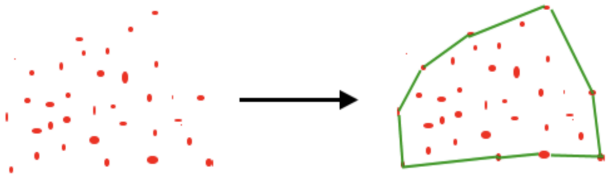


Figure 5: steps of Harris corner detection

but the result is not good because corners were detected in the inner space of clothes. So we used a second approach.

Second approach is to use a convex hull. First, get the contour of the segmented image and get the convex hull from the contour. Then we can get corners from the convex hull. Convex hull is a set that is the smallest convex polygon that contains all the points of it.



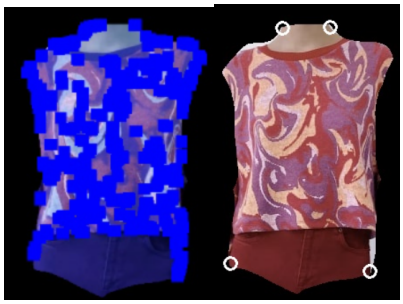


Figure 6: Results of corner detection (Approach 1 & 2)

3.3. Joint detection

We conducted joint detection using the Openpose library. After putting on a pre-trained model weight, we can predict the location of joints. We used this method to match the order of corners between human image and clothes image. But it was unnecessary because the order of corner detection was well-arranged.

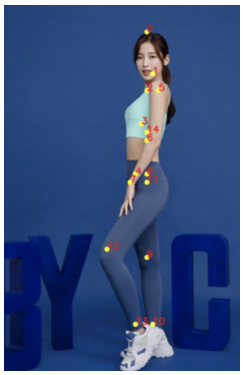


Figure 7: Results of joint detection

3.4. Warping and Blending

With correspondence points and clothes segments obtained by above procedure, conduct image warping to make clothes segments be located at corresponding human body part. Here, we tried both TPS warping and perspective warping, and choosed to use TPS for better performance.

However, the segmentation obtained from deep learning methods is not perfect. They can contain a small fraction of background, which has clearly different RGB values compared to human image background. Thus, we decided to apply an image pyramid blending technique to make the resulting image look natural.

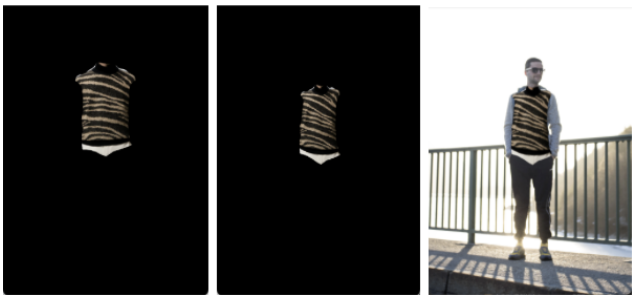


Figure 8: Original Clothes segment, warped clothes segment, blended clothes on human pose

4. Experiment

4.1. Usage of heuristics

The performance of basic computer vision techniques used in our project were improved using diverse heuristics.

4.1.1. Heuristic 1: Dismiss ignorable contours

The findContours function often detects many contours that are extremely small (contours of eyes, fingertips…). These contours may affect the final result and thus must be removed. In order to ignore them, we found the area of each contour and deleted all the contours whose area didn't surpass a certain threshold. The results are as follows:

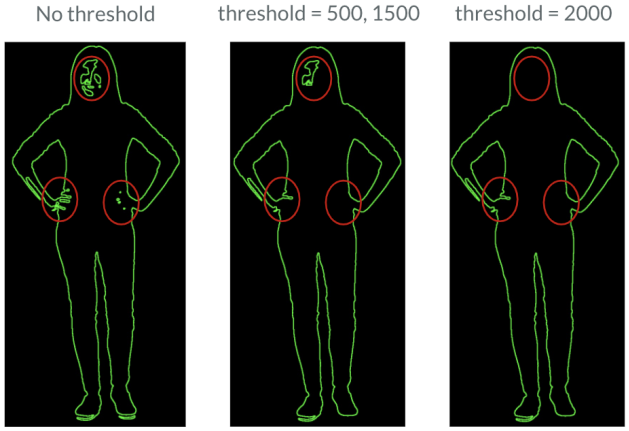


Figure 9: result of ignoring small contours

4.1.2. Heuristic 2: grouping similar points

As mentioned before in section 2.2, the corner detection function often gives several corners in the same area. And despite grouping these corners together using the connectedComponentsWithStats function, we still saw the same phenomenon, which is why we added a heuristic for grouping similar points.

We defined similar points as points that have similar coordinates (absolute value of difference of each coordinate is less than a certain threshold), and replaced each group of points with their centroid.

The results can be seen as follows:

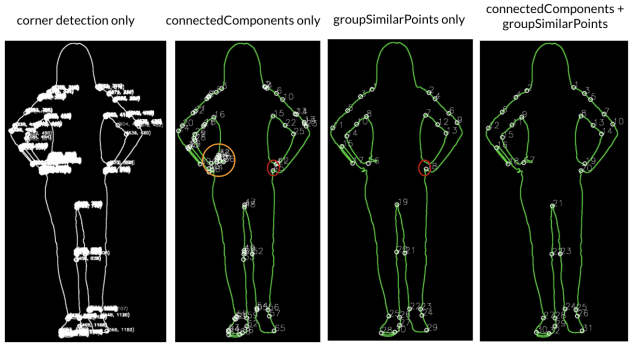


Figure 10: applying different heuristics to group similar points

4.1.3. Heuristic 3: Using rectangles to better detect corners

After segmenting each body part using the OpenPose library, we have to determine the corners of each body part. In order to do this, we apply the same technique as approach 2 (section 3.1.2). However, as mentioned in the section, the algorithm detects all the corners of the image and thus we had to handpick manually the target points. This manual intervention can be avoided if we restrict the input image to body parts, because the corner points that we aim to find are the corners of the segmented body part image, which simplifies the task. Applying the algorithm on the body parts and selecting the four "best" corners given by the algorithm yields the results shown in figure ? (a). We notice that the "best" corners returned by the algorithm aren't the four corners that we are trying to find.

In order to overcome this shortcoming, we used the fact that most of the body parts that we wanted to detect are rectangular, and selected the corner points that are the closest to the four corners of the rectangle surrounding the body part image. This gave the results shown in figure ? (b).

Figure 11: (a) without heuristic, (b) heuristic applied

4.2. Two types of warpings

Observing the results from perspective warping and TPS warping, we noticed that there could be pattern disagreements, hollow spaces, and length disagreements between clothes segments. Since conducting perspective warping is equal to finding a new position of camera in 3-D world coordinate, a specific set of correspondence points can produce significant movement of camera location. On the other hand, TPS is a nonlinear warping, so it will find the most natural position of each clothes pixels by exploiting bending and twisting.

Figure 12: warped clothes with same correspondence points (left: perspective, right: TPS)

4.3. Suitable blending pyramid height

Height of the blending pyramid can also be a good hyper parameter. With height 0 (not blending, just laying the clothes segment on a human pose image), we could see clear boundaries of clothes segments. Using higher height, we could see the boundaries get dim. However, when the pyramid height is bigger than 2, we could see the parts that belong to clothes and human pose also get dim.

Figure 13: blending results (left: height 0, right: height 2)

This phenomenon occurs because of pixel value operation. Since there are lots of white pixels around the clothes and human pose, significant errors from cv2.subtract and cv2.add will occur often. The errors will be simply ignored, so the image after blending will get white and foggy. However, clothes and humans in a white background is a common case, it is reasonable to set pyramid height to 1 for all the cases.

5. Result

Figure 14: final result

We found correspondence points very well for some of the cases, but some cases, we failed. The reason was, since we are detecting with computer vision techniques, there were some pair of points with distance between them is very small. In that case, since TPS is a nonlinear transformation, the warped result bounced to out of human body segmentation region. In those cases, we manually modified the point coordinates, to get the following result.

6. Conclusion

This project presents new methods in implementing virtual try-on programs. Our code allows a decent level of warping and blending clothes segments, so we succeeded to implement elasticity and deformability of clothes. Through those procedures, we could gain an image that looks similar enough to the image that the target person really put on the clothes if corresponding points of each segment are well detected. However, in detecting corner points of each segment, naive and classic computer vision techniques do not guarantee the same results in every posture and clothes

even when the same algorithm is applied. That was because finding robust and consistent features of corner points even in irregular postures and clothes were nearly impossible. To supplement this, we introduced some deep learning techniques in our project, and could segment both images properly. In conclusion, our method produced flexible try-on images for most of the input.

References

[1] Xintong Han, Zuxuan Wu, Zhe Wu, Ruichi Yu, and Larry S. Davis. VITON: An Image-based Virtual Try-on Network. In CVPR, 2018.

[2] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Rad- ford, and X. Chen. Improved techniques for training gans. In. NIPS, 2016.

[3] Satoshi Suzuki and others. Topological structural analysis of digitized binary images by border following. Computer Vision, Graphics, and Image Processing, 30(1):32–46, 1985.

[4] Kevin Lin, Lijuan Wang, Kun Luo, Yinpeng Chen, Zicheng Liu, Ming-Ting Sun. Cross-Domain Complementary Learning Using Pose for Multi-Person Part Segmentation. ICCV, 2019.

[5] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, Yaser Sheikh. OpenPose: Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields. IEEE, 2021.

[6] Michael JD Powell. The uniform convergence of thin plate spline interpolation in two dimensions. Numerische Mathematik, 68(1):107-128, 1994.