Automatic Photo to Photo Shoe Replacement

Mu-Ti Chung, Nam Gyu Kil, Junhwan Kim, Shuoqi Wang

***Abstract*—To improve the shopping experience on fashion items during the pandemic, an algorithm is developed to simulate a virtual fitting room specifically for footwear. Powered by computer vision techniques including detection, segmentation and stitching, the algorithm provides a visual concept of how the look would be without any limitations on brands, models, etc.**

I. Introduction

When it comes to fashion, physically trying on the merchandise has always been the only reliable way when making shopping decisions. With the outbreak of COVID-19 and stay in home mandates, this method is no longer applicable as consumers' clothes and other wearable are mostly accessible only through online means. Although there exists commercial applications like Wanna Kicks that allows users try on shoes through augmented reality (AR) techniques, the number of available shoes is limited since each requires an individual model. This project aims to facilitate the decision process via allowing consumers to virtually try on products. An algorithm[1] is designed to take user's daily photos and multiple orientations of retailer shoes from online shops as input, and output the original user photo with the retailer shoes stitched at the correct position, angle and scale to simulate the actual look.

Implementation details of all three parts will be discussed in Section II. Sample results, both successful and failed cases, will be shown in Section III. Finally, conclusions and future improvements will be given in Section IV.

II. Methodology

A. Detection

The YOLOv3 [1] model is chosen as the detection framework for this project. Its predecessor YOLO [2] gets its name "You Only Look Once" from that the image passes through the network only once. Multiple bounding boxes and corresponding class probabilities are simultaneously predicted. Benefited from this simple, single-pass behavior, the network can be highly optimized at both training and inference time, reaching a high frames-per-second. YOLOv3, based on its previous versions, came with some new features and ideas, including making predictions from multiple scales, a concept alike feature pyramids. Moreover, using binary cross-entropy loss instead of softmax loss for class predictions allows the model to be trained on more complex datasets that includes overlapping labels, such as the Open Images Dataset [3].

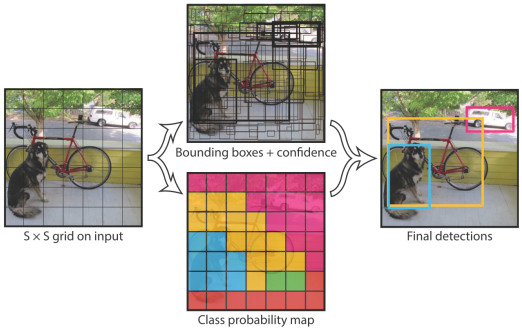

Fig. 1: A sketch of the YOLO model. Dividing the image into $S \times S$ grid, the network predicts four offset parameters that transform anchor boxes to bounding boxes, along with one confidence score for each box. In addition, a class probability is predicted for each grid cell.

Attempts of training a specific network for shoe-detection task have been made but unsuccessful. Given the limited time for the project and that the emphasis was not put on detection networks, eventually the YOLOv3 model was implemented with all the weights pretrained on the Open Images Dataset V6. The dataset contains 601 classes including shoe-related ones, e.g. footwear, boots, and clothes. To ensure that the network only outputs the bounding boxes of shoes, a filter was applied to extract only the predictions with high classification scores in the categories mentioned above and at the same time discard the misclassified none-shoe clothes. Finally, the network outputs the bounding box coordinates to the detected shoes.

B. Segmentation

Segmentation is a process to extract the contour of the shoes from the input images. It needs to be applied on the retail shoes images and user images respectively, therefore different methodologies are required. The retail shoes images from the brand's official website usually have a plain single-color background. Since the content of the retail shoes image is simple, one way to extract the counter of the shoe is treating images as functions and implementing a range map operation which will extract all pixels or super pixels having different color from the

[1]Full code can be accessed on Google Colab or on GitHub

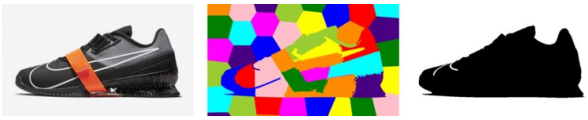

Fig. 2: Example Segmentation of Retail Shoes

background. Figure 2 shows an example of retail image segmentation of a Nike shoe.

Comparing to the retail shoe images, the user image comes with higher complexity. To extract the shoes from the user image, images as graphs better formulates the problem, and the min-cut/max-flow graph cut algorithm can be applied to accomplish the foreground/background segmentation task. A binary mask is created where only the pixel or super pixels similar to the target are labeled. A general method to find those pixels is minimizing the energy function as shown below.

$$E(x) = \sum_{u=1}^{m \times n} x_u f(I(u); \theta_f) + \sum_{u=1}^{m \times n} (1 - x_u) \cdot b(I(u); \theta_b)$$

$I(u)$ represents image, $\theta_{f,b}$ represent the key values of foreground or background, x is a binary function, f and b are functions which determines the weight of edges between pixels and the target. Min-cut/max-flow graph cut is a globally optimal solution to solve such a minimization problem. By representing the images as graph, a flow network can be constructed for images. A flow network is a directed graph where each edge has a non-negative capacity, and the capacity is usually determined by the color similarity and proximity between pixels or super pixels. The Ford & Fulkerson algorithm, an algorithm that can compute max flow in a flow network, can sort pixels or super pixels of image to two parts where the sum of capacities in each part reach maximum. In this project, a pixel or super pixel from the shoe was selected as the target to do the foreground/background segmentation.

In the project, all super pixels were created using simple linear iterative clustering (SLIC) super pixel [4], which clusters pixels in the image plane based on their color similarity and proximity to generate a specific number of super pixels. In python code, the `skimage.segmentation.slic` [5] function was used. The first two parameters of the function `n_segments` and `compactness`, which influence the number and the shape of the super pixels, were determined after a series of tests while others were kept as the default values. Based on the collected data, it was assumed that all input retail shoes images have a near-white background. To obtain the shoe masks from the retail shoe image, a function which can detect all non-near-white pixels and segment them as foreground. A similar result could be obtained with a similar function

Fig. 3: Example segmentation of user shoes. Top figures: good segmentation. Bottom figures: bad segmentation

which can detect all super pixels having non-white center pixel and segment those super pixels as foreground. For the user image, because it is hard to find a global optimal function that determine weight of edges between super pixels, the foreground/background segmentation code we wrote had mixed results as seen in Figure 3.

After multiple iterations of tuning the capacity values based on the color and proximity, our team decided to use the OpenCV's built in `grabCut` function [6]. The algorithm is another variation of the graph-cut algorithm (like Ford & Fulkerson) that uses a Gaussian Mixture Model and Markov Random Field before applying graph cut. Though the results were similar using graph cut on the bounding of the user's shoe photos, we added a mask of probable regions of foreground and background to help the algorithm in segmentation. To find the mask that indicated possible foreground, our team used the scale invariant feature transform (SIFT) [7] feature points on the bounding box of the shoe as the algorithm would detect stronger SIFT feature points within the contours of the shoes. After finding the feature points, we found a convex polygon that enclosed all the feature points and then used that as a mask that indicated possible foreground pixel points. Figure 4 shows an example of the process of using SIFT, convex hull, and GrabCut to find the final segmentation of the shoe in user's photos.

Fig. 4: Example segmentation of user shoes with GrabCut. Left figure: SIFT features. Middle figure: convex hull of SIFT features. Right figure: segmentation using GrabCut

C. Stitching

The task of stitching is divided mainly into two part: 1) finding corresponding feature points between shoes, and 2) finding the best transformation that would convert one shoe shape to another based on these feature points.

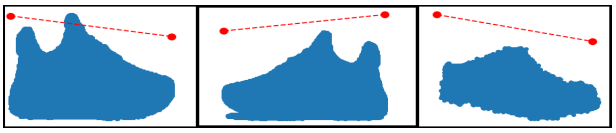

Fig. 5: Direction of the largest eigenvector visualized in red for retailer shoes (left, middle) and user shoe (right)

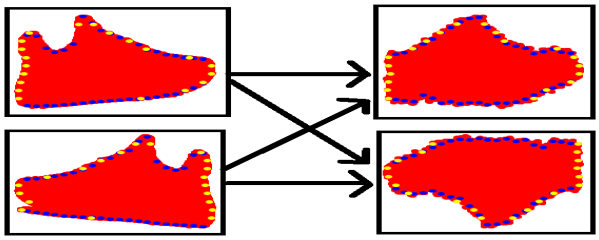

Fig. 6: Contour matching process between left/ right retailer shoes (left column) to different orientations of a single user shoe (right column). Feature points from vertical cuts visualized in blue, horizontal cuts in yellow.

1) Finding corresponding feature points: While ideally corresponding feature points between shoes should be extracted based on sophisticated intra-class invariances, we instead performed a simple contour matching between shoes using PCA and the segmentation map of each shoe.

First, to overcome the different orientation of shoes, we performed PCA on the coordinates of each pixel in the segmentation map to acquire the direction of the largest eigenvector. This is done based on the assumption that the segmented shoe is generally longer in the direction it points to (Figure 5). Then, each shoe is rotated around their centers so that their largest eigenvectors are aligned horizontally. Only specially for the user shoe, we consider rotating the shoe in both directions because sometimes shoes in user photos are faced upside-down with the shoe's sole facing the sky (e.g. running). This later impacts the number of possible orientation matching we need to consider for stitching (Figure 6).

To acquire features points for each shoe, we made proportionally spaced vertical and horizontal cuts on the segmentation map to find contour points. The matching between contour points across shoes are made so that the contour points from the leftmost vertical cut of one shoe corresponds to the contour points from the leftmost vertical cut of another shoe and so forth (Figure 6).

2) Finding the best transformation: We considered two types of transformations in this step: similarity and homography. Similarity transform only allows four degrees of freedom: rotation, scale and horizontal and vertical translation, and homography transform allows eight degrees of freedom. While homography provides a more versatile transformation since it allows higher

Fig. 7: Comparison between similarity (middle) and homogarphy (right) transformation results, given the occluded user shoe segmentation (left).

degree of freedom, it can overfit to the dimensions of the user shoes, which can be undesirable at times. Such situation can be seen in Figure 7 where similarity transform produced a better looking result because the homography transform vertically scaled down the retailer shoe to overfit to the short height of the user shoe.

Both transformations were considered for the best stitching result. Hence the algorithm compares (# transformation type) $\times$ (# retailer shoe orientations) $\times$ (# user shoe orientation) $= 2 \times 2 \times 2 = 8$ possible transformations per user shoe detected in the photo.

To be robust from outliers and bad correspondences between contour points, all transformation matrices were computed using random sample consensus (RANSAC) through OpenCV's `cv2.estimateAffinePartial2D` and `cv2.findHomography` functions [6].

Within each transformation type, the best matching of retail-user orientations was picked first by maximizing the number of inliers output by the RANSAC algorithm, and then by minimzing the least squares error of these inliers.

Finally, to pick between similarity and homography transformation, the best retail-user matching from each transformation were compared by how much the transformed retailer shoe covers the user's shoe when stitched. Such method was picked because an ideal stitching should cover the user shoe entirely, and because the number of inliers produced by RANSAC and homography is always larger than that of similarity transform and is an unsuitable metric for comparing performance.

III. Result

A. Detection

Overall, the YOLOv3 model achieved an accuracy of 84.76% on our self-collected, relatively small dataset of people wearing shoes in different scenarios. Multiple shoes can be successfully detected in different angles (top, bottom, and side views) and scales. However, the model tends to fail when the shoes have similar textures and/or colors to the background. Some results can be seen in Figure 8.

B. Segmentation & Stitching

The stitching algorithm performed well given perfect segmentation results. Also, under the limited scope of

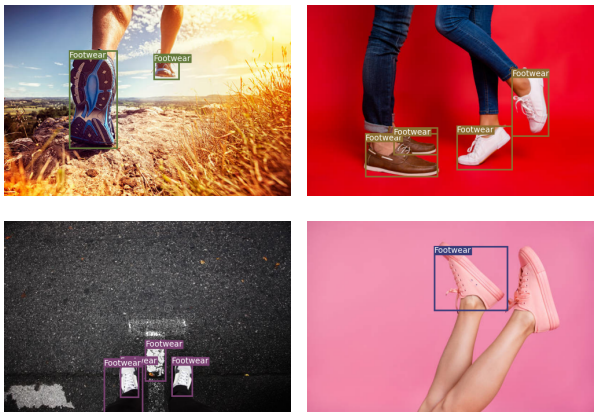

Fig. 8: Examples of detection results.

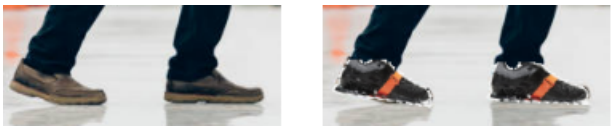

Fig. 9: Example of successfully overlaying retail shoes over user's shoes

this project, we did not consider top and front views of the shoes. Figure 9 shows a successful implementation of our pipeline in overlaying photos from retail websites into user photos. However, with the current segmentation method potentially including the ankle (Figure 3), the stitching algorithm sometimes stitches shoes upside-down in the opposite direction, like the one shown in Figure 10. In addition, it is possible for the performance of the algorithm to quickly deteriorate when too much of the ankle and calf are included in the segmentation because the largest eigenvector is likely to change to the direction of the leg instead of the shoe. Lastly, occlusions would lead to incomplete contours, resulting in bad stitching.

IV. Conclusion

Fashion never comes without actually seeing it, but the pandemic has changed the way how people live and shop. By implementing what we have learned in this semester, an algorithm that is capable of providing decision-making assistance to some extent has been developed. YOLOv3 is successfully modified to fit the needs of the task and detect footwear. Several methods including min-cut/max-flow graph cut and grab cut are tested to perform shoe segmentation under different scenarios. A simple but novel stitching algorithm is designed to match shoe orientations via contour matching. As a proof of concept, the algorithm has accomplished our team's goal.

Possible improvements in the future include using two-stage networks such as Faster R-CNN as the detection network to achieve higher accuracy. Furthermore, combining both the detection and the segmentation task, better performance could be achieved with instance segmentation networks such as Mask R-CNN [8]. The final idea involves utilizing structure from motion methods to build a 3D representation of the shoes from the images with different views. With an actual model of any footwear available on the retailer websites, users can easily try on shoes virtually at arbitrary orientation even in real-time, and the gap between customers and online footwear shops can be narrowed.

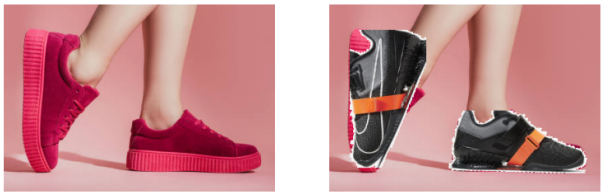

Fig. 10: Stitching result with a shoe stitched upside-down.

References

[1] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *CoRR*, vol. abs/1804.02767, 2018.

[2] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *CoRR*, vol. abs/1506.02640, 2015.

[3] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, A. Kolesnikov, T. Duerig, and V. Ferrari, "The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale," *IJCV*, 2020.

[4] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "SLIC superpixels compared to state-of-the-art superpixel methods," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 11, pp. 2274–2282, 2012.

[5] S. van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, T. Yu, and the scikit-image contributors, "scikit-image: image processing in Python," *PeerJ*, vol. 2, p. e453, 6 2014.

[6] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[7] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2, 1999, pp. 1150–1157 vol.2.

[8] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.