

AI-Based Virtual Try-On System Using Pose Estimation and Garment Overlay

Tejaswini Pabbathi¹, Naga Pranika Valaboju², Megavath Tejashwini³, R. Preethi Rathod⁴, M. Vishwa Vikas⁵, Dr. Kiran B.M⁶

^{1,2,3,4,5} UG Students, Dept. of Computer Science and Engineering, Sphoorthy Engineering College, Hyderabad, Telangana, India

⁶Professor & Head of the Department, Dept. of Computer Science and Engineering, Sphoorthy Engineering College, Hyderabad, Telangana, India

Abstract - Abstract - The rapid growth of e-commerce has created a strong demand for systems that allow users to visualize clothing on themselves before making a purchase. However, most online shopping platforms still do not provide a realistic garment fitting experience, which often leads to high product return rates, typically ranging between 20 to 40 percent in the fashion retail segment.

This paper presents an AI-Based Virtual Try-On System that combines real-time human pose estimation with a segmentation-aware garment overlay pipeline to enable virtual clothing visualization directly within a web browser. The system operates in two modes: a real-time webcam mode for live and interactive garment preview, and a high-quality static image mode for image-based try-on.

MediaPipe Pose is used to detect 33 anatomical body landmarks in real time, which are then utilized to perform perspective-based geometric warping of the garment image onto the detected body region. In addition, a HumanParser-based semantic segmentation module classifies body pixels into multiple categories, including upper clothing, torso skin, face, hair, left arm, right arm, and lower body. This allows the system to place garments more accurately while preserving important features such as the face and limbs.

To further improve visual realism, a luminance transfer technique based on the CIE LAB color space is applied. This helps in capturing the shading and fold characteristics of the original clothing and transferring them onto the garment texture. A GrabCut-based background removal method is also used to process garment images, eliminating the need for pre-masked transparent inputs.

The complete pipeline is implemented as a Flask-based web application, which includes an integrated e-commerce catalog, user authentication, MJPEG-based webcam streaming, and REST API support for static image processing. The system is developed using Python 3.8+, OpenCV 4.11.0, MediaPipe 0.10.9, PyTorch 2.1.0, and CP-VTON+, with cloud-based IDM-VTON available as an optional fallback.

The system achieves real-time performance exceeding 20 frames per second on standard consumer hardware without requiring a dedicated GPU. Experimental results show that the

combination of geometric warping, segmentation-aware clipping, luminance transfer, and alpha blending produces visually consistent and realistic garment overlays, making it a practical solution for virtual try-on applications in online retail.

1. INTRODUCTION

The global fashion e-commerce industry has grown rapidly over the past decade, with market valuations exceeding USD 700 billion in 2023 and expected to surpass USD 1.2 trillion by 2030. Despite this significant growth, one of the key challenges in online clothing retail is that customers are unable to physically try on garments before making a purchase. This limitation often creates uncertainty regarding how a product will fit or look, which in turn leads to a high rate of product returns. It is estimated that clothing items alone contribute to nearly 30–40% of total returns in the e-commerce sector.

To address this issue, virtual try-on systems have been introduced as a potential solution. These systems rely on computer vision and artificial intelligence techniques to simulate how a garment would appear on a person. Early approaches mainly used basic image transformation techniques, which were simple but struggled to handle variations in body shape, pose, and garment structure. Although more advanced deep learning-based methods such as VITON and CP-VTON improved the visual quality of results, they generally require high computational resources, large training datasets, and GPU support, which limits their practical use in real-world applications.

In recent years, lightweight pose estimation frameworks, especially MediaPipe Pose, have made it possible to perform real-time human body tracking even on standard CPU-based systems. When combined with semantic segmentation and efficient image processing techniques, these methods make it feasible to build virtual try-on systems that are both practical and accessible.

In this work, an AI-based virtual try-on system is developed with a focus on efficiency and usability on consumer-grade hardware, while still maintaining visually consistent results. The system integrates pose estimation, segmentation-aware garment placement, geometric warping, and image enhancement techniques into a single pipeline. Unlike many

existing approaches, the system is implemented as a web-based application, allowing users to access it directly through a browser without requiring any specialized hardware or additional software installation.

The system supports two modes of operation: a real-time webcam mode that enables interactive visualization, and a static image mode that produces higher-quality outputs. By combining accuracy, efficiency, and accessibility, the proposed approach aims to provide a practical solution for virtual fitting in modern online retail environments.

2. LITERATURE REVIEW

The problem of virtual try-on has been widely studied in recent years, especially in the area of image-based garment transfer. Early approaches mainly relied on geometric transformations such as thin-plate spline (TPS) to align clothing images with the target body. While these methods were relatively simple from a computational perspective, they often struggled to handle variations in pose, body shape, and garment structure effectively.

A significant advancement in this field was introduced with VITON by Han et al. [1], which proposed a two-stage framework that combines a coarse person representation with a refinement network to generate the final output. Although this approach produced promising results, it frequently resulted in blurry outputs and was not always able to preserve fine garment details.

To overcome these limitations, Wang et al. [2] proposed CP-VTON, which focused on better preservation of garment characteristics using a geometric matching module. This method improved texture retention by estimating transformation parameters based on appearance flow. Later, CP-VTON+ [5] further enhanced the performance by introducing improved loss functions that captured both perceptual and structural information more effectively, resulting in clearer and more detailed outputs.

High-resolution approaches such as HR-VITON [3] aimed to further improve visual quality by addressing issues related to misalignment and occlusion. While these methods achieved sharper and more visually appealing results, they required significant computational resources, including high-end GPUs, which limits their use in real-time or large-scale applications.

Another important aspect of virtual try-on systems is accurate body segmentation. Li et al. [6] showed that precise pixel-level classification of body parts can greatly improve garment placement. Their work emphasized the importance of separating regions such as the face, hair, and arms to avoid unrealistic overlay artifacts. This idea has been widely adopted in more recent systems that use segmentation-aware placement strategies.

Human pose estimation also plays a crucial role in aligning garments with the body. Earlier systems such as OpenPose [7] provided reliable multi-person pose detection, but they required GPU acceleration for real-time performance. More recently, MediaPipe Pose [4] has provided a lightweight alternative that can run efficiently on CPU-based systems while still maintaining good accuracy. This makes it more suitable for practical, real-world applications where computational resources may be limited.

Recent work has also explored generative models, particularly diffusion-based approaches such as IDM-VTON [9], which are capable of producing highly realistic outputs. However, these methods generally require substantial computational power and are not always suitable for real-time use.

In addition to pose estimation and garment alignment, pre-processing of garment images is another key component. Techniques such as GrabCut [10] are commonly used for background removal, allowing garment images to be extracted from standard product photos without the need for manual editing or pre-processed datasets.

Overall, existing research highlights a clear trade-off between visual quality and computational efficiency. While deep learning-based methods provide high-quality results, they often come with increased complexity and resource requirements. The system proposed in this work attempts to balance these factors by combining lightweight pose estimation, segmentation-aware placement, and efficient image processing techniques to provide a practical and accessible virtual try-on solution.

Table -1: Comparison with Related Work

Method	Pose	Segm.	GPU?	Web?	FPS
VITON [1]	HPE	No	Yes	No	<1
CP-VTON [2]	HPE	Partial	Yes	No	<1
HR-VITON [3]	HPE	Yes	Yes	No	<1
IDM-VTON [9]	DNN	Yes	Yes	Cloud	<0.1
Proposed	MediaPipe	Yes	No	Yes	>20

3. SYSTEM ARCHITECTURE

The proposed AI-Based Virtual Try-On System is designed using a modular architecture that consists of six main com-

ponents, all connected through a Flask-based web application layer. Figure 1 shows the overall system architecture along with the process flow. The design supports both real-time webcam processing and high-quality static image try-on using the same core pipeline, with minor optimizations applied depending on the mode of operation.

such as elbows and wrists are also available and can be used for future extensions like sleeve alignment.

3.2 Garment Processing Module

The Garment Processing Module (`garment_manager.py`) is responsible for loading, preprocessing, and caching garment images. Since most e-commerce images are provided without transparency, the system applies an automatic background removal pipeline.

For images with light backgrounds, adaptive thresholding combined with HSV saturation masking is used to capture both white and colored garment regions. For darker backgrounds, inverse thresholding is applied. In cases where the background is more complex, the GrabCut algorithm is used for segmentation.

After background removal, the binary mask is refined using morphological operations such as closing and opening to remove noise and fill gaps. The largest connected component is retained as the garment region, and the edges are smoothed using Gaussian blurring. The final output is stored as a four-channel BGRA image for further processing.

3.3 Body Segmentation Module

The Body Segmentation Module (`segmentation.py`) uses a HumanParser neural network to perform pixel-level classification of body regions. The module defines multiple labels corresponding to different anatomical regions, including upper clothing, torso skin, face, hair, arms, and lower body.

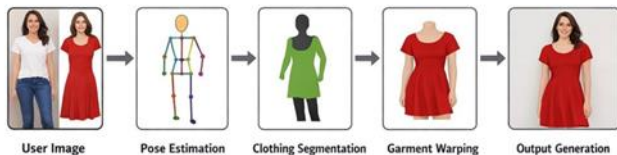
The `parse(image, pose_detector)` method generates a label map that matches the dimensions of the input image. This label map is later used to create a placement mask, ensuring that the garment is applied only to appropriate regions. To maintain real-time performance, segmentation is applied only in static image mode, while webcam mode skips this step.

3.4 Garment Overlay Engine

The Garment Overlay Engine (`garment_overlay.py`) is responsible for aligning and placing the garment onto the person's body. The `overlay_garment(frame, garment, landmarks_dict, scale_factor)` function computes a body quadrilateral using the detected shoulder and hip landmarks.

The garment image is then warped to fit this region using perspective transformation. A configurable scale factor is used to slightly expand the target region for better coverage. The module also supports alpha blending and optional temporal smoothing using an exponential moving average to reduce flickering in real-time mode. A debug mode is available to visualize landmark positions and alignment during development.

System Architecture



Virtual Try-On Process Flow

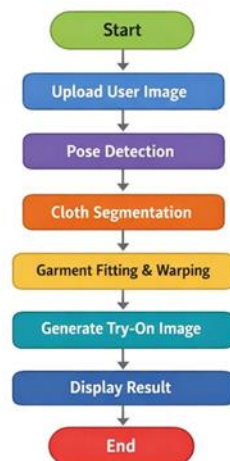


Fig -1: System Architecture and Process Flow of the Proposed Virtual Try-On System

3.1 Pose Detection Module

The Pose Detection Module (`pose_detection.py`) is built on top of the MediaPipe Pose API and follows a stateful detector-tracker approach. When operating in webcam mode (`static_image_mode=False`), the system first performs an initial pose detection and then tracks the landmarks across subsequent frames using the region of interest predicted from the previous frame. This helps in reducing the computational load per frame.

The module provides several utility functions, including a `detect(frame)` method to process input frames, a `get_landmarks_dict()` method that returns landmark coordinates in the form of $(x, y, z, visibility)$, and an `is_pose_detected()` method to verify detection status. A `draw_landmarks(frame)` function is also available for visualization. The key landmarks used for garment placement include the left and right shoulders (indices 11 and 12) and the left and right hips (indices 23 and 24). Additional landmarks

3.5 Luminance Transfer Engine

The Luminance Transfer Engine implements the shading mechanism that improves visual realism. The original image is first converted to the CIE LAB color space, and the luminance (L) channel is extracted.

A shade map is computed by comparing local luminance values with the mean luminance, and this map is clipped and smoothed to avoid extreme variations. The resulting shading information is applied to the garment image in HSV colour space, allowing it to adapt to the lighting conditions of the original image.

Additionally, a colour harmonization step slightly adjusts the garment's colour to match the scene. This helps in reducing the flat appearance of the garment and improves the perception of depth and realism.

3.6 Web Application Layer

The Web Application Layer (app.py) integrates all modules into a single user-facing system. It is built using Flask and provides multiple routes for user interaction, including pages for login, registration, product browsing, webcam try-on, and static image upload.

The application also includes API endpoints for processing images and streaming webcam data. All machine learning components are initialized lazily to optimize performance and ensure thread safety. Uploaded files are managed on the server, and results are served through authenticated routes.

Overall, the architecture is designed to ensure efficient processing, modular design, and ease of use, making the system suitable for deployment on standard hardware without requiring specialized resources.

4. METHODOLOGY

The complete virtual try-on pipeline for static image mode consists of six sequential processing stages. For real-time webcam mode, a simplified version using only three stages (Stages 1, 2, and 4) is executed in order to maintain performance above 20 FPS. In addition, the system follows a cloud-first strategy where it attempts IDM-VTON processing through the HuggingFace Spaces API before falling back to the local pipeline, ensuring the best available output quality.

Stage 1: Pose Detection and Validation

The input person image is loaded using OpenCV (cv2.imread) and passed to the MediaPipe Pose detector configured with `static_image_mode=True`, `model_complexity=1`, `smooth_landmarks=True`, `min_detection_confidence=0.5`, and `min_tracking_confidence=0.5`. The system checks whether the four primary upper-body landmarks (indices 11, 12, 23, and 24) are detected with visibility values greater than 0.5. If these conditions are not met, the in-

put is rejected and the user is prompted to upload a clearer image.

The detected landmark coordinates are initially in normalized form within the range [0, 1] and are converted to absolute pixel coordinates by multiplying them with the image width and height.

Stage 2: Garment Background Removal

The garment image is loaded using `cv2.IMREAD_UNCHANGED` to preserve any existing alpha channel. If the image does not include an alpha channel (i.e., it is a 3-channel BGR image), the automatic background removal pipeline described earlier is applied.

The processed garment image is then cropped to the bounding box of non-transparent pixels ($\alpha > 10$) with a small margin of 5 pixels. If the image exceeds 800 pixels in either dimension, it is resized proportionally while maintaining its aspect ratio. A Garment object is then created using the processed image along with its size metadata.

Stage 3: Segmentation-Aware Placement Mask

The HumanParser model is used to generate a pixel-wise label map for the person image. A clothing mask is initialized and assigned values of 255 for pixels corresponding to LABEL_UPPER and LABEL_TORSO_SKIN.

To improve boundary handling, the mask is dilated using a 15×15 elliptical kernel for two iterations. A proximity zone is then created by further dilation, which defines a region where garment pixels are allowed near the clothing area.

A protection mask is created by marking pixels corresponding to LABEL_FACE, LABEL_HAIR, LABEL_ARM_LEFT, and LABEL_ARM_RIGHT. The final placement mask is computed by combining these masks using logical operations, ensuring that garment placement is restricted to appropriate regions while excluding protected areas.

The resulting mask is smoothed using a Gaussian filter to produce soft edges for blending.

Stage 4: Perspective Warping

The garment is aligned with the body using perspective transformation. A quadrilateral is defined using the four key landmarks: left shoulder, right shoulder, right hip, and left hip.

A scale factor (1.15) is applied to slightly expand the target region. The garment image is treated as a rectangle and mapped to the body quadrilateral using `cv2.getPerspectiveTransform`. The warped garment is generated using `cv2.warpPerspective` with bilinear interpolation.

At the same time, a warped garment mask is created by applying the same transformation to a binary canvas and extracting the relevant region through thresholding.

Stage 5: Luminance Transfer

The luminance transfer step enhances visual realism by adapting the garment’s shading to match the original image. The person image is first converted to the CIE LAB color space, and the luminance (L) channel is extracted.

A shade map is computed as the ratio of local luminance to the mean luminance. This map is clipped to a predefined range and smoothed using a Gaussian filter to reduce noise.

The resulting shade map is applied to the Value channel of the garment image in HSV color space. Additionally, a color harmonization step adjusts the garment’s color slightly toward the original image’s color distribution, improving overall consistency.

Stage 6: Composite Blending and Edge Refinement

The final output is generated by blending the shaded garment with the original image using a soft alpha mask. The alpha mask is obtained by applying morphological operations followed by Gaussian smoothing to create gradual transitions.

The composite image is calculated using weighted blending between the garment and the original image. To further improve visual quality, an edge refinement step is applied near the garment boundaries using a bilateral filter. This helps smooth out color differences while preserving important edge details.

The final result is a visually consistent image where the garment appears naturally fitted onto the person.

5. IMPLEMENTATION DETAILS

The system is implemented entirely in Python 3.8+ and does not require a dedicated GPU for executing the primary local pipeline. The full set of dependencies used in the system is listed in Table 2. All machine learning components are initialized lazily during the first request using a shared dictionary protected by a threading.Lock, which ensures thread safety when handling multiple users simultaneously.

Table -2: Technology Stack and Dependencies

Component	Library	Version / Role
Pose Estimation	MediaPipe	0.10.9 / 33 landmarks
Computer Vi-	OpenCV	4.11.0 / Warp,

Component	Library	Version / Role
sion		blend
Deep Learning	PyTorch	2.1.0 / Model backend
Image Processing	Pillow	10.1.0 / I/O support
Try-On Model	CP-VTON+	Geometric matching
Web Framework	Flask	3.1.3 / Web server
Cloud Try-On	IDM-VTON	HF Spaces API
HF Client	gradio_client	2.2.0 / API client
Numerical Ops	NumPy	1.24.3 / Array ops
Language	Python	3.8+ / Primary lang.

The application is designed to run on standard consumer hardware with a minimum requirement of 4GB RAM, although 8GB is recommended for stable performance when all components are loaded together.

The project is organized into modular Python source files within the src/ directory. The main modules include pose_detection.py for handling MediaPipe-based landmark detection and caching, garment_manager.py for garment loading and preprocessing, garment_overlay.py for perspective transformation and overlay operations, segmentation.py for HumanParser-based segmentation, cpvton_adapter.py for integrating CP-VTON+, and cloud_tryon.py for accessing IDM-VTON through the HuggingFace API.

Additional modules include body_measurements.py, which performs pixel-to-centimeter calibration using detected landmarks, and size_recommender.py, which provides garment size suggestions based on shoulder and torso measurements.

For the webcam mode, the application uses an MJPEG streaming approach. The /api/video-feed route generates a multipart response where frames are encoded as JPEG images at 80% quality. Each frame is processed through the pose detection and overlay pipeline in real time, and the result is stored in a shared state dictionary for capture requests.

The /api/process-static endpoint handles static image processing by accepting multipart form data containing both person and garment images. These files are saved using unique filenames to avoid conflicts, processed through the full pipeline, and the output image is returned as a URL.

User authentication is implemented using Flask sessions. The application supports login, logout, and registration features, along with a few predefined test accounts. Uploaded images and generated results are stored securely and accessed only through authenticated routes to ensure privacy and controlled access.

6. RESULTS AND DISCUSSION

The proposed system was tested on a standard desktop configuration consisting of an Intel Core i5 processor, 8GB RAM, and integrated graphics without a dedicated GPU. This setup was chosen to demonstrate that the system can perform effectively on commonly available hardware.

6.1 Real-Time Webcam Performance

In webcam mode, the system consistently achieved frame rates above 20 frames per second during continuous operation. The MediaPipe Pose module required approximately 15–25 milliseconds per frame for landmark detection, while the perspective warping step added around 3–5 milliseconds. The blending operations contributed minimal additional latency.

Overall, the total processing time per frame was around 25–35 milliseconds, corresponding to approximately 28–40 FPS. This performance level is sufficient for smooth and interactive visualization.

6.2 Static Image Try-On Quality

For static image processing, the system produced high-quality outputs using the full six-stage pipeline. The processing time ranged between 2.1 and 4.3 seconds depending on image complexity and segmentation processing time.

The segmentation-aware placement ensured that garments were applied only to appropriate regions, effectively preserving features such as the face, hair, and arms. The luminance transfer step improved realism by adapting shading and fold patterns from the original clothing.

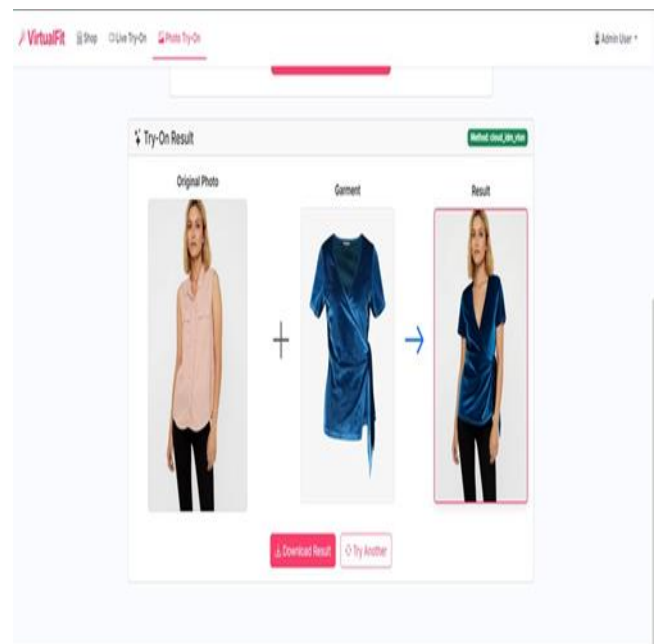


Fig -2: Static Image Try-On Result (Method: cloud_idm_vton)

6.3 Background Removal Accuracy

The background removal module performed effectively across different types of garment images. Images with simple backgrounds were processed quickly using thresholding techniques, while more complex images required GrabCut processing. In both cases, the system produced clean garment silhouettes with minimal noise.

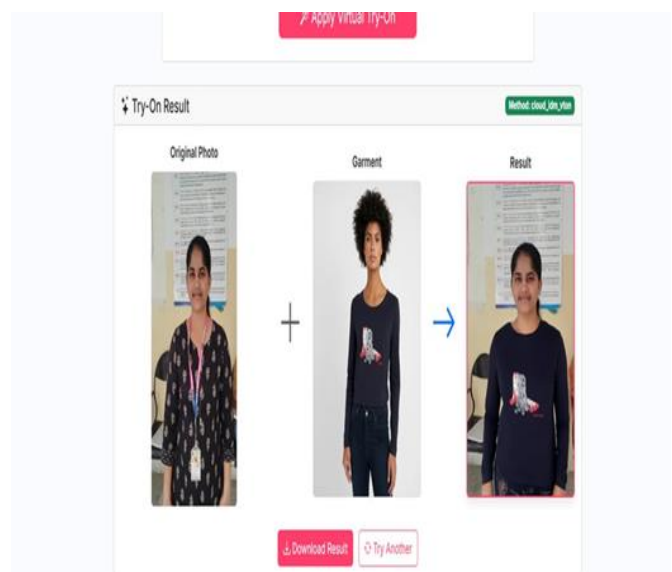


Fig -3: Real User Try-On Result Showing Original Photo, Garment, and Output

Table -3: Performance Summary

Metric	Webcam Mode	Static Mode
Processing Speed	>20 FPS	2.1-4.3 sec
GPU Required	No	No
Segmentation Clipping	No	Yes
Luminance Transfer	No	Yes
Face Protection	Partial	Full
Min. RAM Required	4GB	4GB
Cloud Fallback	Yes	Yes

6.4 Discussion

The use of segmentation-aware placement provided clear improvements over basic overlay techniques by preventing unrealistic artifacts such as garments covering the face or floating above the body. Similarly, the luminance transfer mechanism helped address the common issue of flat-looking garment textures.

However, certain limitations were observed. The segmentation module occasionally faced challenges with loose or complex clothing boundaries. In addition, the perspective warping approach does not fully capture fabric behavior, which may result in minor distortions for certain garment types.

Despite these limitations, the system offers a practical and efficient solution. The web-based implementation further improves accessibility by allowing users to interact with the system directly through a browser without requiring specialized hardware or installation.

7. CONCLUSIONS

In this work, an AI-Based Virtual Try-On System was developed that combines pose estimation, segmentation, and image processing techniques to provide a realistic virtual clothing experience. The system is capable of operating in real time without requiring GPU support, making it accessible on standard consumer hardware.

The integration of segmentation-aware placement, luminance transfer, and geometric warping contributes to improved visual quality and accurate garment alignment. The

use of a web-based interface further enhances usability by allowing easy access through a browser.

The results demonstrate that the system is effective for practical virtual try-on applications and can help improve user experience in online shopping platforms.

Future work can focus on extending support to additional clothing types, improving garment fitting accuracy, and incorporating more advanced techniques such as cloth simulation and multi-layer garment handling.

ACKNOWLEDGEMENT

The authors sincerely thank Dr. Kiran B.M., Professor & Head of the Department, Department of Computer Science and Engineering, Sphoorthy Engineering College, Hyderabad, for invaluable guidance, technical mentorship, and consistent support throughout this project. The authors also acknowledge the Department of Computer Science and Engineering, Sphoorthy Engineering College, Hyderabad, Telangana, for providing the infrastructure and computational resources necessary to carry out this research.

REFERENCES

- [1] X. Han, Z. Wu, Z. Wu, R. Yu, and L. S. Davis, "VITON: An image-based virtual try-on network," in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Salt Lake City, USA, 2018, pp. 7543-7552.
- [2] B. Wang, H. Zheng, X. Liang, Y. Chen, L. Lin, and M. Yang, "Toward characteristic-preserving image-based virtual try-on network," in Proc. Eur. Conf. Comput. Vis. (ECCV), Munich, Germany, 2018, pp. 589-604.
- [3] S. Lee, G. Gu, S. Park, S. Choi, and J. Choo, "High-resolution virtual try-on with misalignment and occlusion-handled conditions," in Proc. Eur. Conf. Comput. Vis. (ECCV), Tel Aviv, Israel, 2022, pp. 440-456.
- [4] V. Bazarevsky, I. Grishchenko, K. Raveendran, T. Zhu, F. Zhang, and M. Grundmann, "BlazePose: On-device real-time body pose tracking," arXiv preprint arXiv:2006.10204, 2020.
- [5] M. Minar, T. T. Thi, H. Ahn, P. Herghelegiu, and H. Rothkrantz, "CP-VTON+: Clothing shape and texture preserving image-based virtual try-on," in Proc. IEEE/CVF CVPR Workshops (CVPRW), 2020.
- [6] P. Li, Y. Xu, Y. Wei, and Y. Yang, "Self-correction for human parsing," IEEE Trans. Pattern Anal. Mach. Intell., vol. 44, no. 6, pp. 3260-3271, Jun. 2022.
- [7] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh, "OpenPose: Realtime multi-person 2D pose estimation using part affinity fields," IEEE Trans. Pattern Anal. Mach. Intell., vol. 43, no. 1, pp. 172-186, Jan. 2021.

- [8] A. Raj, P. Sangkloy, H. Chang, J. Lu, D. Ceylan, and J. Hays, "SwapNet: Image based garment transfer," in Proc. Eur. Conf. Comput. Vis. (ECCV), Munich, Germany, 2018, pp. 666-682.
- [9] Y. Choi, S. Kwak, K. Lee, H. Kim, and J. Choo, "Improving diffusion models for virtual try-on," arXiv preprint arXiv:2403.05139, 2024.
- [10] C. Rother, V. Kolmogorov, and A. Blake, "GrabCut: Interactive foreground extraction using iterated graph cuts," ACM Trans. Graph., vol. 23, no. 3, pp. 309-314, Aug. 2004.