VTONShoes: Virtual Try-on of Shoes in Augmented Reality on a Mobile Device

Wenshuang Song *
School of Information,
Renmin University of China
Baidu ART

Yanhe Gong†
Baidu ART

Yongcai Wang‡
School of Information,
Renmin University of China

Figure 1: The results are generated from VTONShoes. The top row shows input images from the monocular camera, while the bottom row shows the rendered 3D shoe models in different foot poses.

Abstract

The virtual try-on (VTON) system in augmented reality (AR) has attracted significant research interest. This paper presents a novel real-time AR virtual shoe try-on system (VTONShoes). Users can see the virtual shoes with full degrees of freedom on the mobile device. In particular, we propose an efficient framework to detect, classify, and recover 6-DoF pose of shoes from the captured images and then accurately render the 3D shoe model on the screen in real-time and in full degrees of freedom. For accurate pose recovery, dense keypoints are designed on the 3D shoe model. An efficient joint 2D keypoint localization and leg silhouette segmentation module (KeyPointLoc) is designed to predict keypoint projections on 2D images and the shoe-leg occlusion relationship. In order to reduce jitter between frames, an optimization-based framework with the longest continuous invariant subarray constraints is proposed to minimize classification errors caused by model switching, and a smoothing module with Exponential Weights Decay is presented to post-process the rendered results. We also developed a large-scale dataset named *Diverse-Shoes* which contains images extracted from 80K videos, annotated with the shoe bounding box, transformation matrices, and silhouettes of legs. Our system has achieved a smooth and stable try-on effect on mainstream devices with a real-time speed of around 25 to 45 FPS on mainstream mobile phones, which significantly outperforms state-of-the-art methods for real-time performance and rendering accuracy.

Index Terms: Human-centered computing—Mixed/augmented reality—6-DoF pose estimation—Virtual try-on;

*e-mail: wenshuangsong@ruc.edu.cn
†e-mail: pkgongyh@126.com
‡e-mail: ycw@ruc.edu.cn, corresponding author. He is also a researcher at the Metaverse Research Center, Renmin University of China

1 Introduction

Recently, AR technology has gone through two stages, from focusing on the interaction between virtual objects and the environment to emphasizing the interaction between virtual objects and humans. Mobile AR technology, e.g., virtual fitting mirrors and virtual try-on system [12], has entered the field of e-commerce and been vertically applied to the beauty, makeup, and clothing industry, bringing a novel online shopping experience.

Virtual try-on (VTON) [12,21,31] is a challenging task in human-computer interaction, which can be roughly divided into two approaches. One is 2D image-based VTON technologies, and the other is 3D model-based methods. Because of the high costs of building 3D models, 2D image-based VTON approaches are becoming prevalent, e.g., virtual makeup and virtual try-on for clothes with an invariable posture, which do not need to generate a 3d model and real-time poses of the virtual objects. In existing studies, VITON [12] synthesizes a photo-realistic new image by overlaying a product image seamlessly onto the corresponding region of a clothed person without leveraging a 3D model. CP-VTON [31] can generate more realistic image-based virtual try-on results that preserve well key characteristics of the in-shop clothes. CP-VTON+ [21], after the baseline CP-VTON, improves the composition mask using the input clothing mask and a concrete loss function. ACGPN [34] takes the spatial semantic layout into consideration to generate photo-realistic try-on images when large occlusions and human poses are presented in the reference person.

However, virtual try-on systems for watches, glasses [10,17], and shoes [2,8,35] are selfie-led AR shopping. During virtual shopping,

2022 IEEE International Symposium on Mixed and Augmented Reality (ISMAR) | 978-1-6654-5325-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/ISMAR55827.2022.00038

978-1-6654-5325-7/22/$31.00 ©2022 IEEE
DOI 10.1109/ISMAR55827.2022.00038

users desire to see the product's real appearances for different body poses. This kind of method needs to render a 3D model on screen to bring users an immersive real shopping experience. Kobayashi et al. [17] proposal a fast and accurate face tracking tool that enables the system to automatically display 3D virtual glasses following a user's head motion.

But virtual try-on for shoes is less studied at present, which is still a challenging task. Requirements for virtual shoe try-on mainly include the following aspects: (1) Accurately recovering the foot's 3D poses from a single 2D image is the basis for 3D model rendering. (2) Rendering the 3D shoe model over the 2D shoes on the screen immersively and smoothly while the user's feet move freely. (3) The virtual try-on system should run in real-time and stably on mobile devices, balancing between computation efficiency and rendering accuracy.

In existing studies, PIVTONS [8] collects the first paired feet and shoes virtual try-on dataset, called Zalando-shoes, which contains 14,062 shoes among the 11 categories of shoes. The shoe image contains only a single view of the shoes, but the try-on result shows other views of shoes depending on the foot pose in the target image. They formulate the virtual try-on task as an automatic and labor-free image completion task and design an end-to-end neural network composed of a feature point detector. But their method hasn't provided real-time virtual try-on experience. ARShoe [2] proposes a real-time AR virtual shoe try-on system for smartphones. They propose to localize a set of sparse keypoints on shoes by regressing their heatmaps. Although their method is efficient, it is not always accurate and smooth, e.g., mutual occlusion or self-occlusion between the two shoes. And it can not cover full degrees of freedom on some challenging poses, e.g., in the direction of the heel.

In this work, we build an efficient framework called VTONShoes, a multi-stage network for co-learning the entire task, including shoe detection, shoe classification, and an *efficient joint 2D keypoint localization and leg silhouette segmentation module (KeyPointLoc)*. We adopt a *top-down* approach to firstly detect each shoe's bounding box and then estimate each shoe's pose.

The critical problem is to recover 6-DoF pose of the shoe from the detected shoe clip image. Since 6-DoF pose estimation from a 2D image is an ill-posed problem, to improve recovering accuracy, we design dense keypoints on the 3D shoe model and propose a joint keypoint localization and shoe-leg silhouette segmentation module (*KeyPointLoc*) to process the shoe clip images, which predicts the projected shoe keypoints on the image and the shoe-leg occlusion relationship.

To train the *KeyPointLoc* module, we construct and annotate a large-scale 2D-3D shoe dataset called *Diverse-Shoes*. The dataset collects images extracted from about 80K shoe motion videos from 200 unique people with different poses, shoe categories, and backgrounds. The annotations include the bounding box of each shoe area, left or right shoe, pose transformation matrix, legs silhouette segmentation areas, etc. We also propose a new drag-and-drop 3D pose annotation tool to annotate the above information simultaneously. Each image's projected keypoints on the image plane are used as the supervised signal to train the *KeyPointLoc* module. After keypoints prediction on the image, a SolvePnP algorithm is exploited to recover 6-DoF pose of the shoes on the image. Then the 3D shoe models are rendered on the screen to overlay the shoe area based on the recovered shoe poses. VTONShoes achieve an immersive real-time experience covering full degrees of freedom of shoes and can run stably on mainstream mobile devices at speeds from 25 to 45 FPS. It significantly surpasses existing shoe try-on systems in terms of stability, accuracy, and visual experiences. Key contributions of this paper include:

- We introduce an efficient multi-stage network for co-learning the real-time augmented reality shoes try-on system, including shoe detection, classification, and keypoints prediction tasks.
- A joint shoe keypoint localization and shoe-leg silhouette segmentation module *KeyPointLoc* is proposed, relying on a dense representation with 40 keypoints and SolvePnP-based 6-DoF pose recovery.
- We collect a large-scale 2D-3D shoe dataset named *Diverse-Shoes* which contains about 80K videos annotated with the bounding box, transformation matrices, and silhouettes of legs by our novel drag-and-drop 3D pose annotation tool.
- Our system achieves a smooth and stable try-on effect on mainstream devices with a real-time speed and achieves much better real-time performance and computational efficiency than state-of-the-art methods.

2 Related Work

2.1 2D Pose Estimation

Since there are usually two shoes on the screen, the shoe keypoint localization task belongs to the multi-objective keypoint localization task, which is similar to human pose estimation. Recently, existing 2D human pose estimation methods can be divided into two categories: (1) heatmap-based methods [6, 22, 23, 32], which dominate in the field of human pose estimation by modeling the output distribution through the likelihood heatmaps but suffer from high computation and storage demands; (2) regression-based methods [19, 29, 33] that directly map the input to the output, which are more efficient but suffer from inferior performance. Li et al. [19] propose a novel regression paradigm with Residual Log-likelihood Estimation (RLE) to capture the underlying output distribution.

Multi-person pose estimation can be further divided by the routine. *Bottom-up* [6] methods detect all keypoints in a given image, then the keypoints are grouped to human instances; *Top-down* [7] methods employ a person detector and perform a single-person pose estimation for each detection. Openpose [6] presents the first bottom-up representation of association scores via Part Affinity Fields (PAFs), utilizing a set of 2D vector fields that encode the location and orientation of limbs over the image. ARShoe [2] firstly detects all the possible keypoints in an input image, followed by a grouping process to produce the final instance-aware keypoints, which is also a bottom-up method. In our work, we choose a top-down strategy to firstly detect each shoe and then locate keypoints on each shoe. This can improve accuracy, and the pose estimation tasks on two shoes can be run in parallel.

2.2 Virtual Try-on in Augmented Reality

VTON (Virtual Trying On) system in augmented reality has attracted significant interest from many researchers due to its great commercial potential. VTON [12, 21, 31] is also a challenging task. There have two mainstream approaches in VTON: a) 2D image-based VTON technologies; b) 3D model-based methods.

Specifically speaking, 2D image-based VTON approaches, e.g., virtual makeup and virtual try-on for clothes with an invariable posture, do not need to generate a 3d model of the virtual object. VITON [12] seamlessly transfers the desired clothing item onto the corresponding region of a person using a coarse-to-fine strategy. CP-VTON [31] implements a novel learnable thin-plate spline transformation via a tailored convolutional neural network to well align the in-shop clothes with the target person. It's worth noting that the network parameters are trained from paired images of in-shop clothes and a wearer. CP-VTON+ [21] reveals the origins of cases with rich-textured or long-sleeved clothing and redesigns the pipeline by employing better input representations. ACGPN [34] adopts a split-transform-merge strategy to adaptively determine the generation or preservation of the distinct human parts in the synthesized image. PT-VTON [36] proposes a multi-stage system that

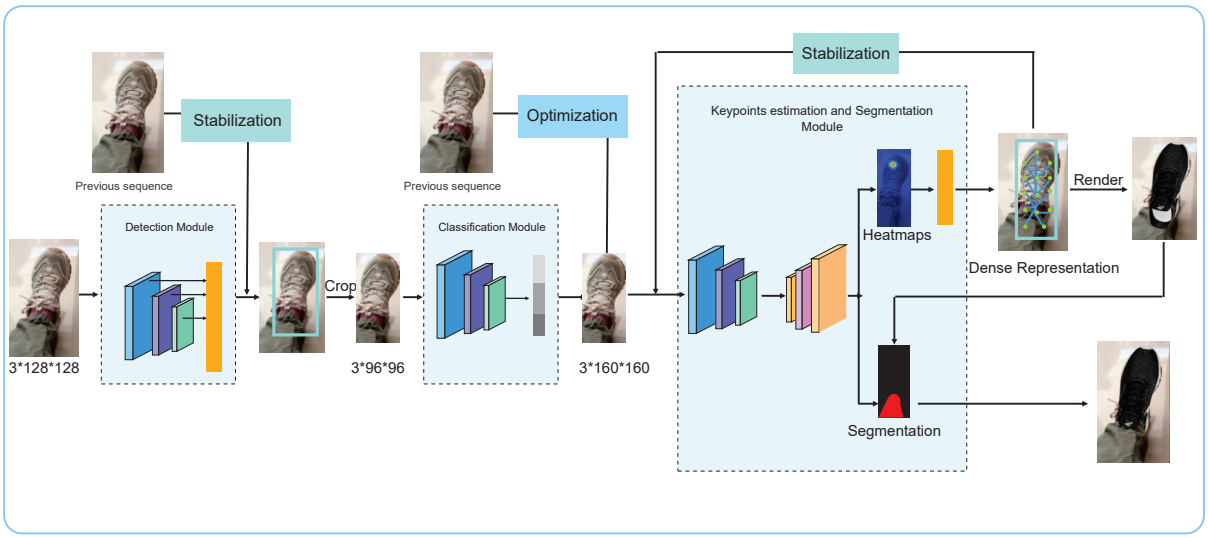

Figure 2: Overview of the proposed Virtual shoe try-on framework. We first detect the shoes in the incoming RGB image through the MobileNet-FSSD network, which is based on MobileNetV2 [27] and Single Shot multi-box Detector (SSD) [15]. To smooth the Region Of Interest (ROI), this area is combined with the output of the previous frame. Subsequently, we crop the ROI and feed it into the Classfication module and KeyPointLoc module, respectively. The encoder-decoder network is used to predict 2D keypoints of the shoes and the occlusion relationship between shoes and legs. The predicted 2D keypoints and the 40 keypoints on the shoe model will be used to estimate the final 6-DoF pose by SolvePnP. Please refer to Sec. 3 for detailed description.

accepts user image input of any pose for cloth transferring, including generating a side-view or a rear-view, which many solutions do not support, paving a potential path to an image-based 360-degrees virtual try-on system.

On the contrary, 3D model-based approaches are highly required in virtual try-on for watches, glasses [10, 17], and shoes [2, 8, 35]. This kind of method needs to render the 3D model on the screen. At the same time, the occlusion relationship with clothes should be considered to get an immersive shopping experience. Kobayashi et al. [17] employs a Convolutional Experts Constrained Local Model (CE-CLM) algorithm as a facial landmark detection and face tracking method for fast and accurate face tracking so that they can display 3D virtual glasses following the user's head motion in realtime. Feng et al. [10] implements a novel virtual glasses try-on method based on head pose estimation without using any other special hardware. In order to estimate the continuous pose angle of a face, they use 3D face reconstruction and pose estimation.

2.3 Virtual Try-on For Shoes

Virtual try-on for shoes is less studied at present. Antonio et al. [14] present a stereoscopic rendering system that gives users the illusion of physically seeing the virtual shoe model. Eisert et al. [9] use augmented reality techniques in order to create a Virtual Mirror for the real-time visualization of customized sports shoes. They presented a 3D motion tracker, which exploits mainly silhouette information to achieve robust pose estimation for shoes from a single camera view. Yang et al. [35] allow users to real-time try on 3D shoes in a video stream using commercial depth-sensing technologies. Marker-less tracking driven by an iterative closest point (ICP) algorithm was implemented to correctly position the model with respect to the moving foot during the try-on process. However, the methods using specialized equipment are not widely accessible.

Recent work studies virtual try-on for shoes based on images. PIVTONS [8] formulate a conditional image completion problem. They firstly detect the bounding box of shoes in the source image, hide the region inside the bounding box, and generate the target image considering the region outside the bounding box and the conditions of the target shoe. But the system is not real-time. ARShoe [2] proposes a real-time augmented reality virtual shoe try-on system for smartphones, which adopts a novel multi-branch network to realize pose estimation. But the shoe rendering accuracy and smoothness following shoe poses still need further investigation. WannaKicks[1] has a stunning performance in the area of virtual try-on for shoes. Meanwhile, Kivisense[2] AR Sneaker Try-on system is a Web application for virtual try-on for shoes, but there are challenges with difficult angles like legs crossed or occlusions occurring. Vyking[3] has good adaptability to truncation but can not cover full degrees of freedom. ARTao[4] can cover full degrees of freedom but is not robust when occlusions occur, or the camera moves fast.

3 Methodology

3.1 Overview

During virtual try-on of shoes, a user can view virtual shoes from different viewing angles while holding the mobile device using different orientations. Our task is to estimate 6-DoF pose of the user's shoes and render the shoe model appropriately in real-time. The input of the virtual try-on system is the sequential images $\mathbf{I}_l$ extracted from video captured by a mobile phone camera, where $l \in \{0, ..., t\}$. The output is the rendered 3D shoe model on the screen according to the estimated 6-DoF pose in real-time. The proposed network architecture is illustrated in Fig. 2. The method incorporates four modules:

(1) *Shoe detection module*: a deep neural network detector, which detects the bounding box of each shoe.

(2) *Classification module*: which distinguishes left or right shoe by using the detector's bounding box as input.

(3) *KeyPointLoc module*: an efficient joint 2D keypoint localization and leg silhouette segmentation module, which infers the 2D

[1]https://wanna.fashion/
[2]https://tryon.kivisense.com/
[3]https://www.vyking.io/
[4]https://www.taobao.com/

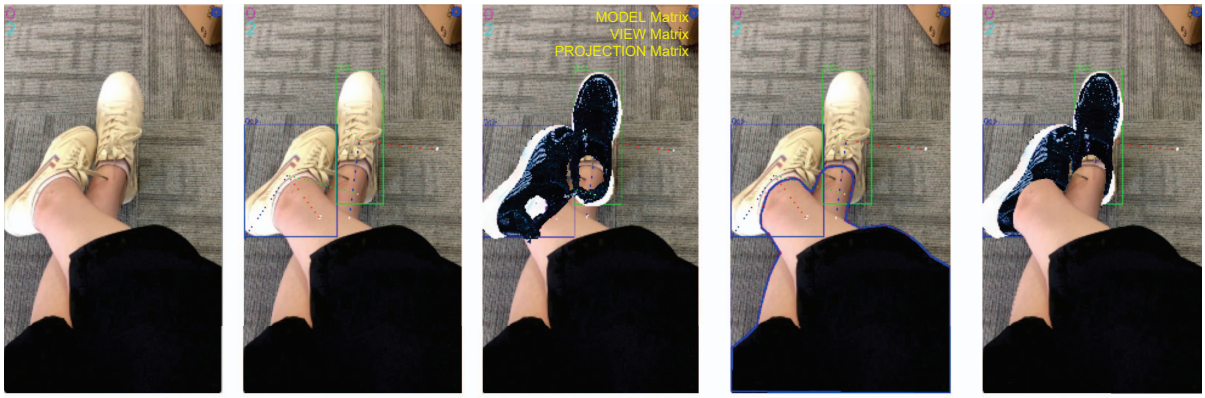

Figure 3: An illustrative example of the proposed annotation tool. From left to right: input images, the bounding box of detection and classification result, the 3D pose of the virtual shoes, the segmentation of the legs, and the final pose of the virtual shoes, respectively.

projections $v_{pred} \in \mathbb{R}^{J\times 2}$ of the 3D shoe model's dense keypoints on the shoe clip image and the occlusion relationship between legs and shoes.

(4) *6-DoF pose estimation module*: it computes 6-DoF pose according to the estimated 2D keypoints from KeyPointLoc module, the camera intrinsics, and the fixed 3D keypoints on the shoe model by a SolvePnP algorithm [1]. Then, it renders the 3D shoe model on the screen according to the estimated 6-DoF pose.

The following sections present the details of the proposed framework.

3.2 Shoes Detection and Stabilization

As introduced in the related work section, camera-based multiple object pose estimation can be divided into *top-down* [7] and *bottom up* [6] two kinds of approaches. For pose recovery accuracy, we will assign each shoe an equal number of keypoints, so we adopt a *top-down* approach to firstly detect the bounding box of each shoe and then estimate the pose of each shoe.

The input image is firstly fed into the MobileNet-FSSD (Feature Fusion Single Shot Multibox Detector) network [20] to detect the shoe bounding box. Afterward, we obtain a stabilized Region Of Interest (ROI) through an Exponential Weights Decay mechanism using ROIs detected in early images. Then we crop the ROI and feed it into the classification module to distinguish between left or right shoe. More specifically, we show an overview of the proposed Detector pipeline in Fig. 2. Assuming $\mathbf{I}_l$ is the current image, thus the previous frame is $\mathbf{I}_{l-1}$.

Network Architecture: The network firstly detects the shoes. Afterward are shoe classification, keypoint detection, pose estimation, and rendering of the 3d shoe model. This helps to reduce the computational latency for improving the real-time performance. We select MobileNetV2 [27] as feature extractors for shoe detection with a modified version of the Feature Fusion Single Shot Multibox Detector(FSSD) [20] pre-trained on the ImageNet dataset [26] as the head of the detector. Inspired by MobileNetV2 [27], we extract features with a layer module: the inverted residual with linear bottleneck, which takes a low-dimensional compressed representation as input. The intermediate representation layer uses lightweight depth-wise convolutions to filter features as a source of non-linearity. More details about the structure of the inverted residual with linear bottleneck is shown in our supplementary material. The input dimension of the network is $3 \times 128 \times 128$. It has 3 channels, with an image height and width of 128. The shoe network will output the coordinates of the bounding box for the shoe ROI.

Stabilization Module: For shoe detection from successive images, the detected bounding box has inevitable jitters. Inspired by [3], the early perceived frames present a high correlation with the current frame. By introducing an Exponential Weights Decay mechanism, we achieve more accurate prediction scores by stabilizing the shoe bounding box. We exploit the correlation between the current box and the previous ones to weigh the neighboring images. The current bounding-box coordinates, denoted by P_{out} are calculated as:

$$P_{out} = \sum_{i=1}^{T} P_i \times \frac{e^{-|i-t|}}{\sum_{k=1}^{T} e^{-|k-t|}}, \tag{1}$$

where i represents the frame's index, k represents the frame's index in sliding window, t represents the current frame's index, and T corresponds to the number of the examined frames. This value was set heuristically to 8 ($T = 8$). P_i are the coordinates of the i bounding box. So, $\frac{e^{-|i-t|}}{\sum_{k=1}^{T} e^{-|k-t|}}$ is the exponentially decreasing weight.

3.3 Shoe Classifier and Optimization Module

Because we have different 3D shoe model, e.g., left or right shoe model, the cropped ROI from Shoe Detector are then fed into the classification module to distinguish left or right shoe. After that, the result of the Classifier will be sent to KeyPointLoc module for locating the keypoints.

Classification Network Architecture: We select MnasNet [30] as feature extractor for shoe classification pre-trained on ImageNet dataset [26]. The input dimension of the network is $3 \times 96 \times 96$, which are equivalent to 3 channel, with a height of 96 and a width of 96. The network will output a left or right label.

Optimization Module: To achieve a smooth estimation, we further optimize the temporal information of the classification model because we find the error of classification leads to switching of the shoe model between left and right. We propose an invariant subarray-based classification optimization algorithm. We stack a sequence of frames as an array. Only when the predicted category is continuous and the occurrence times are higher than a threshold will the result be updated. Otherwise, it will be regarded as an outlier and will not be processed. In this way, we achieve a stable classification result.

3.4 An Efficient Joint 2D Keypoint Localization and Leg Silhouette Segmentation Module

In this section, an encoder-decoder structure is proposed for keypoint localization and leg silhouette segmentation on the cropped shoe image. The routine is outlined in Fig. 2. The keypoint localization branch takes the result from Shoe Classifier as input and outputs a set of J low-resolution heatmaps, where J is the number of keypoints.

We set J to 40 in our module. Each heatmap $P_{HM} \in \mathbb{R}^{H \times W}$ represents a 2D probability distribution of one predicted keypoint. The leg silhouette segmentation branch is used to generate a reasonable occlusion relationship between shoes and legs.

A dense representation with 40 keypoints: On the standard 3D shoe model, we selected 40 fixed keypoints as the feature points of the shoe. Because we find that SolvePnP is sensitive to the number of keypoints, and simply adding points can't improve the effect. We propose a keypoint selection method based on singular value decomposition, which can get a better effect. When we look at a shoe in different poses on the screen, the 40 keypoints on the shoe will be projected to different positions on the 2D image plane. The *KeyPointLoc* module takes the shoe clip image as input and aims to predict the 40 keypoints' positions on the image plane. To train this module, we developed an annotated dataset.

Diverse-Shoes dataset: We construct and annotate a large-scale 2D-3D shoe dataset called *Diverse-Shoes*. The dataset collects about 80K shoe motion videos from 200 unique people with different poses, shoe categories, and backgrounds. At first, videos are processed to extract images containing shoes. For each image containing shoes, we use an annotation tool to manually crop the shoe area and drag the standard 3D shoe model from its initial pose to fit the shoe's area on the 2D image suitably. We then manually draw a boundary for leg silhouette segmentation. Fig.3 shows an example of this process. When the 3D model's projected image best fits the shoe in the 2D image, the annotation tool will record the following information:

(1) the bounding box and the classes of each shoe in the 2D image;

(2) the *model matrix*, *view matrix*, and *projection matrix* that transform the shoe's 3D model to its 2D projected image.

(3) the leg silhouette segmentation result.

Using this annotation information, in the training process of *KeyPointLoc* model, the 2D shoe images are used as input data. For each shoe image, its annotated *model matrix*, *view matrix*, and *projection matrix* will be applied to the keypoints on the standard 3D shoe model to generate the projected keypoints (corresponding to the shoe's pose in the image) in the screen image. The generated projected keypoints, and the leg silhouette segmentation result will be used as the supervised signal to train the joint keypoint detection and leg silhouette segmentation module.

Project 3D Model Keypoints to 2D Screen: The *model view projection* is a common series of matrix transformations [18] that transform a 3D model from *model space* to its projected 2D image in *window space*. The transformation has three transformation matrices, corresponding to *model matrix* (M), *view matrix* (V), and *projection matrix* (P) respectively. In our work, we get these three matrices through the annotation tool.

The keypoints on the standard shoe model are in *model space*, which are denoted by $V_{model} \in \mathbb{R}^{3 \times J}$, where J is the number of keypoints, and we set as 40 here. In homogeneous coordinates, $V_{model} = [\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{1}]^\top$. By applying *model matrix* and *view matrix* to it, we will yield the keypoints in *camera space*:

$$V_{camera} = [\mathbf{x}', \mathbf{y}', \mathbf{z}', \mathbf{1}]^\top = V \cdot M \cdot V_{model} \tag{2}$$

By applying *projection matrix* to $V_{camera} = [\mathbf{x}', \mathbf{y}', \mathbf{z}', \mathbf{1}]^\top$, a 2D coordinate in homogeneous form is produced in *clip space*:

$$v_{clip} = [\mathbf{x}'', \mathbf{y}'', \mathbf{1}]^\top = P \cdot V \cdot M \cdot V_{model} \tag{3}$$

The points in the clip space are then transformed into normalized device coordinates (NDC) via implicit perspective division. Finally, during rasterization, a viewport transform is applied to interpolate the keypoints' positions, resulting in their positions on the screen. This way, we obtain the ground-truth of the keypoints coordinates in *device space*, denoted by v_{device}. Then we can train the KeyPointLoc module.

Train loss: To train this module, the loss functions are keypoints loss(L_{keyp}) and segmentation loss(L_{seg}). The keypoints localization network output is a set of J low-resolution heatmaps. Each heatmap $P_{HM} \in \mathbb{R}^{H \times W}$ represents a 2D probability distribution of one predicted keypoints, where W and H are the width and height of the image. The predicted keypoints in the 2D screen space $v_{pred} \in \mathbb{R}^{2 \times J}$ are the peak locations on these heatmaps. To train this module, the keypoints loss function is:

$$L_{keyp}(P_{HM}, v_{device}) = \sum_{h}^{H} \sum_{w}^{W} (P_{HM}^{(h,w)} - G(v_{device})^{(h,w)})^2 \tag{4}$$

The loss measures L_2 distance between the predicted heatmaps P_{HM} and the heatmaps $G(v_{device})$ rendered from the ground truth v_{device} through a Gaussian kernel [22].

The segmentation loss(L_{seg}) is CrossEntropy. Our overall loss function for this part is $L_{keyp+seg} = \lambda_s L_{seg} + \lambda_k L_{keyp}$, where λ_s and λ_k are coefficients to balance the contributions of each loss. In our implementation, they are set to 0.5 and 0.5, respectively.

3.5 Six Degrees of Freedom (6-DoF) Pose Estimation

After predicting the keypoints on the device screen, the predicted 2D keypoints $v_{pred} \in \mathbb{R}^{J \times 2}$ are on device space. The fixed 3D points on the 3D shoe model are denoted by $V_{model} \in \mathbb{R}^{J \times 3}$. To render the 3D shoe model properly on the device screen, a SolvePnP algorithm is exploited to recover 6-DoF pose of the shoe captured by the phone based on its predicted 2D keypoints v_{pred}.

The pose of a shoe is described by its rotation and translation matrices. We denote $\mathbf{R} \in \mathbb{SO}(3)$, $\mathbf{t} \in \mathbb{R}^3$, where $\mathbf{R}$ and $\mathbf{t} = [t^x, t^y, t^z]^\top$ are the rotation matrix and the translation matrix of the shoe. We use the term $\mathbb{SO}(n)$ to denote the *special orthogonal group* and denote camera intrinsics matrix $\mathbf{K}$. Since we have 2D keypoints on the device screen and 3D keypoints on the shoe model, we estimate the parameter of rotation and translation with the solvePnP algorithm.

SolvePnP to recover shoe pose: For the solvePnP algorithm, we choose DLT (Direct Linear Transformation) [1] to solve the rotation and translation matrix. DLT is a common solution of SolvePnP which needs at least 6 points. We have 40 pairs of keypoints, and the redundancy improves the robustness. Given their 3D to 2D point correspondences, DLT computes a 4x4 transformation matrix.

The projection from 3D point to 2D point can be expressed as:

$$\lambda_i \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \mathbf{K}[\mathbf{R} \mid \mathbf{t}] \begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix}, i = 1, \cdots, 40 \tag{5}$$

Using the 40 pairs of 2D-3D correspondence between the screen image and the 3D shoe model, linear equations are set up, and $\mathbf{R}, \mathbf{t}$ are calculated.

Converting OpenCV cameras to OpenGL cameras: In order to render the shoe model on the screen, the $[\mathbf{R}|\mathbf{t}]$ matrix generated from SolverPnP can not be directly used because it is the transformation represented in OpenCV. But we need to render the 3D model in OpenGL. Because image coordinate systems is different in OpenCV and OpenGL, we convert OpenCV intrinsic camera matrix $\mathbf{K}$ and the extrinsic matrix $[\mathbf{R}|\mathbf{t}]$ to OpenGL *modelview matrix* and *projection matrix:* $\mathbf{P}$. Note that the *modelview matrix* is $V \times M$ in Eqn. 2.

In OpenCV coordinate systems, the principal axis is aligned with $+z$ axis. Specifically, the $+z$ axis is pointing towards the field of view of the camera. Whereas, the principal axis is aligned with $-z$ axis in OpenGL image coordinate systems. Given this distinction, the x axis need to rotate 180 degrees between the two different coordinate systems. Then we denote $\mathbf{T_{axis}}$ as the rotation matrix of 180 degrees about the x axis. By left multiplying $\mathbf{T_{axis}}$ to the extrinsic matrix

$[\mathbf{R}|\mathbf{t}]$, we obtain the $[\mathbf{R}|\mathbf{t}]_{\mathbf{GL}}$ in OpenGL, which can be formulized as:

$$[\mathbf{R} \mid \mathbf{t}]_{GL} = \mathbf{T_{axis}} [\mathbf{R} \mid \mathbf{t}] \tag{6}$$

where $\mathbf{T_{axis}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$. Then we add a row to make it square:

$$\mathbf{M_{modelview}} = \begin{bmatrix} & \mathbf{R_{GL}} & & \mathbf{t_{GL}} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7}$$

In this way, we obtain the *modelview matrix* in OpenGL.

As mentioned in Eqn.3, in order to perform clipping, OpenGL's *projection matrix* needs to project all the points in the perspective frustum into a normalized device coordinate (NDC) space, and only the points in the NDC space can be displayed on the screen. The OpenGL camera has two faces, near and far, and the clipping points are between the two planes. And only the points between near and far faces can be projected into NDC space. Therefore, our *projection matrix* not only has the same perspective as the intrinsic parameter matrix $\mathbf{K}$, but also projects points onto the NDC space. Because perspective projection transformation is most widely used in computer graphics, which is more in line with our actual observation, so we choose the perspective projection $\mathbf{P_{ps}}$ as our final transform matrix. To sum up, according to the above constraints, we can get the final form of OpenGL *projection matrix*:

$$\mathbf{P_{ps}} = \begin{bmatrix} \frac{2f_x}{W} & 0 & 1-\frac{2c_x}{W} & 0 \\ 0 & \frac{2f_y}{H} & \frac{2c_y}{H}-1 & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix} \tag{8}$$

In this way, we obtain the OpenGL *projection matrix* and *modelview matrix*. Then by Eqn.3, we can render our shoe model with the estimated pose on the screen.

Annotation tool: Now there are several available 6-DoF pose annotation tools include multi-camera labeling tools [13, 25] and 3D bounding box labeling tools [11, 28]. Multi-camera labeling tools need multiple cameras to shoot from different angles. According to the relationship between the cameras, the 3D information of the recovery points can be accurately labeled. But the cost is high, and the operation is complicated. The 3D bounding box labeling tools realize the selection and annotation of objects in a 2D image by pulling an outer bounding box. However, the bounding box can hardly be labeled with high accuracy.

We propose an annotation tool to simulate the rendering assignment of the final 3D shoe model. Based on the 3D rendering assignment, the 3D model of shoes is rotated, translated, and scaled manually to cover the position of shoes in the screen image, as shown in Fig. 3. Therefore, the final transformation matrices, including *model matrix*, *view matrix*, and *projection matrix* can be obtained. To make the annotation more efficient, we propose a standard model. We selected 40 points from the 3D standard model. The standard 3D shoe model is a canonical model, and the new 3D shoe model will be aligned to the canonical model. So we don't need to label each 3D model manually. We only need to drag the 3D standard model to the correct pose on the image with our annotation tool, which improves the labeling efficiency greatly. According to the transformation matrix and the camera intrinsics, the 3D points of the model can be projected to 2D point coordinates on the screen, which are used for training the KeyPointLoc module.

4 EXPERIMENT

In this section, extensive experiments are conducted to demonstrate the effectiveness of VTONShoes.

4.1 Experimental Setting

Implementation details: All models in our experiments are trained with Adam [16] optimizer with an initial learning rate of 1×10^{-4}, and it decays 10 times when the loss is stable until the network converges. The entire network was trained for a total of 500 epochs, and the batch size was set to 64. When training shoe detection net, the parameters of Bottleneck in the encoder stage is initialized with ResNet50 that is pre-trained on the ImageNet [26]. All experiments were conducted on four NVIDIA RTX 2080ti GPU with the PyTorch framework.

Dataset Distribution: Table 1 demonstrate the data partition in our proposed *Diverse-Shoes* dataset. The dataset is split into three parts: Train, Evaluation, and Test. We utilize the Test part of the *Diverse-Shoes* dataset to evaluate each part of our network.

Compared Network Models: We compare different networks for keypoint localization and Silhouette segmentation. For keypoint localization, we compare our proposed network with Lightweith OpenPose [24]. For Silhouette segmentation, we compare our method with that in YOLACT [5]. For the classification, detection, and keypoint localization tasks, we also compared our proposed method with other backbone networks, including ResNet, MnasNet, and Shufflenet.

Evaluation metrics: We use the following metrics in quantitative evaluations: *Acc* is the correct classified number of samples divided by the total number of samples in the classifier. *mIoU* is the Mean Intersection over Union, corresponding to a threshold of 0.5. *mAP* is the mean Average Precision, that is, the average value of average precision (AP) for each category. *MPJPE* measures the mean per joint position error by Euclidean distance (mm) between the estimated and ground-truth coordinates. *Param* indicates the number of parameters. *FLOPs* means the floating point operations or the amount of computation used to measure the complexity of an algorithm. *FPS* is the number of frames per second. *Accel* captures the acceleration of 2D keypoints localization in pixel/s^2 or mm/s^2 to reflect temporal coherence. *OPA* is the offset of the principal axis (the axis of the toe and heel) to evaluate 6-DoF pose estimation.

4.2 Evaluation of Each Module

We verified the contribution of each module in the proposed framework, and the effectiveness of each module in the VTONShoes framework is evaluated based on the *Diverse-Shoes* datasets.

Effectiveness of our dense keypoint representation. SolvePnP

Table 1: Datasets distribution on different framework (frames).

	train	val	test
Detection	313372	8514	3032
Classfication	383382	8963	5478
KeyPointLoc	343056	8772	3883

Table 2: Comparative singular values and condition numbers of the cube and the shoe at different keypoints number.

model	keypoints number	condition number	singular values
Cube	4	4.255	6.511 3.343 1.559
	6	4.030	7.653 4.671 1.899
	8	3.000	8.485 5.657 2.828
	12	2.977	9.510 6.325 3.091
Shoes	8	7.8068	208.219 95.903 26.671
	12	2.556	237.548 170.527 92.924
	16	2.351	295.255 140.875 125.546

Table 3: Comparison of different number of keypoints.

the number of keypoints	Accel (mm/s^2)	OPA (degree)
8	53	55
20	48	37
40	**23**	**15**

Figure 4: An illustration of our network results. From left to right: the result of our classifier, the result of our detection, and the result of our KeyPointLoc.

Figure 5: The result of left-right classification without (top) and with (bottom) the optimization module in the classifier. Without the optimization module, the system gets lots of misclassification results.

is sensitive to the number of keypoints, and simply adding points can't improve the accuracy. Inspired by *Numerical Methods for Least Squares Problems* [4], we compare the singular value (describe the mean distribution along an axis) and the condition number on the shoe and the cube 3D model when a different number of keypoints are selected. A singular value of a real matrix is the positive square root of an eigenvalue of the symmetric matrix. The condition number of a function measures how much the output value of the function can change for a small change in the input argument*Numerical Analysis*. A small condition number is desired for reliability.

As shown in Table 2, we can see that the singular values are different, and the condition number of the cube and the shoes decrease as the number of keypoints increases. A coarse keypoint representation like 8 keypoints is sensitive to large angles. The dense representation improves the reliability compared to the coarse representation of shoe keypoints. As demonstrated in Table 3, with the increase of the number of keypoints, the *Accel*(the acceleration of 2D keypoints localization) error decrease, and the *OPA*(the offset of the principal axis) becomes smaller. In the following sections, we use dense representation with 40 keypoints.

Effectiveness of the optimization module in the classifier. To show the effectiveness of the optimization module in the classifier, we compare the classification results with and without optimization. Fig. 5 visually shows the results with and without the optimization module in the classifier. The prediction results of the classifier with the optimization module have a reliable prediction (the bottom row of Fig. 5) on the input sequences. The network without the optimization module shows much worse prediction results in experiments, as seen from the top row of Fig. 5. Since left and right shoe are similar, the optimization module shows its necessity and effectiveness in improving the classification accuracy.

Effectiveness of the stabilization module in the detector. To verify the impact of the stabilization module in the detector, we evaluate the *Accel* and *OPA* performances with and without this module for shoe detection. The results are given in Table 4. The network with the stabilization block presents a smaller *Accel* error and *OPA*. *Accel* error describes the acceleration of 2D keypoint localizations estimated from KeyPointLoc module and reflects temporal coherence. *OPA* is the offset of the principal axis that evaluates 6-DoF pose estimation. With the decrease of the *Accel* error, the jitter is reduced, and the shoe detection becomes smoother. Because the final rendering result relies on the detecting stage, the smoothing of the shoe detection benefits the downstream keypoint localization, 6-DoF pose estimation, and rendering, making them more accurate and smooth.

Table 4: Comparison of with or without the stable module in the detector.

	Accel (mm/s^2)	**OPA (degree)**
w/o stabilization module	45	32
w/ stabilization module	**34**	**15**

We further compare the keypoint localization module, segmentation module, and 6-DoF pose estimation module with state-of-the-art methods, respectively.

Table 5: Results for Keypoint localization and Silhouette segmentation.

	Network	**mIoU**	**MPJPE**
Keypoints localization	Lightweith OpenPose [24]	-	23.78
	Ours	-	**22.82**
Silhouette segmentation	YOLACT [5]	82.7	-
	Ours	**83.5**	-

Keypoint Localizatoin: Due to the limited computation resources of mobile devices, we compare our keypoint localization

module with a lightweight network, i.e., Lightweight OpenPose [24]. Table 5 shows the comparison results. Our keypoint localization module achieves better accuracy on the Test set than Lightweight OpenPose. The *MPJPE* improves from 23.78 to 22.82.

Silhouette segmentation We further compare the silhouette segmentation module with YOLACT [5]. It can be seen that our method improves the *mIoU* from 82.7 to 83.5.

6-DoF Pose Estimation: Because there is no open source benchmark for existing shoe 6-DoF pose estimation, we only evaluate our method on a qualitative experiment with state-of-the-art methods. The evaluation results are shown in Fig.3 of our supplementary materials. Our method has achieved similar or better results on the Test set than ArShoe [2]. In this section, we utilize *OPA* (the offset of the principal axis) for evaluation, in which the average error is below 15 degrees.

4.3 Computation Cost and Running Time Evaluation

We evaluate the networks' number of parameters and the execution times for one frame on different smartphones.

Table 6: Comparison of computation cost.

Framework	Flops (M)	Param (K)
Detection	39.18	833.748
Classfication	6.33	211.569
KeyPointLoc	79.80	217.162

Table 7: Run time of VTONShoes on different smartphones (ms).

	iPhone X	iPhone 12	OnePlus 8RT	OnePlus 6T
Detection	6.232	4.154	6.096	7.528
Classfication	3.268	1.957	3.829	6.463
KeyPointLoc	19.601	16.362	23.627	29.527
Overall time	29.101	22.473	33.552	43.518

Amount of calculation and parameters: Because the computational performance is limited on mobile devices, a lightweight network is required. Towards that end, our network combines the depth-wise convolution operation to reduce the number of parameters in the model. As seen from Table 6, the complexity of our model for the different stages is listed. It shows that the proposed pipeline has a total of *Flops* (0.122 G) and has overall 1.232 M parameters.

Processing time on different smartphones: We further evaluate our network on modern mobile devices: iPhone X, iPhone 12, OnePlus 8RT, and OnePlus 6T, respectively. Using the results of iPhone12 as an example, we can see from Table 7: for shoe detection, 4.154ms/frame is achieved using an input image size of 128×128. For shoe classification, 1.957ms/frame is achieved using the input of 96×96. For the shoe KeyPointLoc module, 16.362ms/frame is achieved using the input of 160×160. Finally, the overall speed can achieve 45FPS (1000/22) on iPhone 12. The speed on OnePlus 6T is still nearly 25 FPS. Because mobile phones will reduce the CPU frequency when the phone heats up, our results are only evaluated during the first one minute.

Comparison with state-of-the-art methods: We have compared state-of-the-art methods. More details about the comparison experiment results can be found in Fig.4-6 of our additional supplementary materials, including CPU usage, Memory usage, and FPS usage.

4.4 Performances using Different Backbone Networks

We further provide quantitative results on how the selection of different backbone networks affects the performances of other modules. The results in Table 8 demonstrate that, our detection network can reach 0.94 *mAP* on the Test dataset on *Diverse-Shoes*. The proposed classification network can reach 0.99 *Acc*. The *mIoU* of the segmentation stage in the proposed module is 83.5. Besides, the *MPJPE* about the keypoints in our framework is 22.82. Table 8 shows that all the above performances achieved in VTONshoes are better than the compared methods using other kinds of backbone networks.

Table 8: Comparative results using different backbone networks. CLS, DET, and KLC are the abbreviations of classification, detector, and keypoint localization, respectively.

	Backbone	Acc	mIoU	mAP	MPJPE	Param (K)	Flops (M)
CLS	ShufflenetV2	98.75	-	-	-	234.76	6.58
	VTONshoes	**99.0**	-	-	-	**211.569**	**6.33**
DET	Resnet50	-	-	0.92	-	912.09	45.78
	MobileNet	-	-	0.93	-	932.97	33.72
	VTONshoes	-	-	**0.94**	-	**833.748**	**39.18**
KLC	Resnet	-	82.1	-	29.08	267.30	88.09
	Hrnet	-	82.9	-	28.49	240.23	95.34
	VTONshoes	-	**83.5**	-	**22.82**	**217.162**	**79.80**

4.5 Qualitative Evaluation

We show the visualization results of the proposed pipeline in Fig. 1 and Fig. 4. Different perspectives of the shoe pose are shown in each column of Fig. 1. Our framework can accurately detect the localization of each shoe while performing favorably in cases where occlusions occur even at some challenging poses. Fig. 4 shows the intermediate results of each module. We compare the results obtained by VTONShoes and ARShoe [2] on the same input images, which is shown in Fig.3 of our supplementary materials. As we can see, VTONShoes predict shoe poses accurately, which outperforms ARShoe [2]. VTONShoes show more smooth, accurate, and reliable performance in shoe rendering. More details about the virtual try-on experiences and the user study can be found in our video and additional supplementary materials.

5 Limitation and Future Work

Although the results are encouraging, our method also has some limitations, such as dramatic changes in motion. So some engineering optimization should be involved for a better inference effect. In the future, the skeleton relationship of the shoe model will be introduced to synthesize more realistic abnormal motion. In future work, we plan to develop efficient algorithms to handle more challenging pose estimation. Finally, we hope to combine future work with virtual fitting to create a more immersive fitting experience for users.

6 Conclusion

This paper presents a novel virtual try-on system for shoes used in Augmented Reality. We construct and annotate a large-scale 2D-3D shoe dataset called *Diverse-Shoes* for training and evaluating the *KeyPointLoc* module. We develop a dense representation for shoes to accurately recover 6-DoF pose of shoes. We conduct two smoothing schemes for shoe detection and classification, respectively, to improve the accuracy and smoothness. Extensive experiments demonstrate that VTONShoes achieve an immersive real-time experience covering full degrees of freedom of shoes and can run stably on mainstream mobile devices at speeds from 25 to 45 FPS. By optimizing accuracy and efficiency simultaneously, our results demonstrate the potential of a virtual try-on system for real-time applications in AR scenarios.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China Grant No. 61972404, Public Computing Cloud of Renmin University of China, and The Metaverse Research Center of Renmin University of China.

REFERENCES

[1] Direct linear transformation from comparator coordinates into object space coordinates in close-range photogrammetry*. *Photogrammetric Engineering Remote Sensing*, 81(2):103–107, 2015. 4, 5

[2] S. An, G. Che, J. Guo, H. Zhu, J. Ye, F. Zhou, Z. Zhu, D. Wei, A. Liu, and W. Zhang. Arshoe: Real-time augmented reality shoe try-on system on smartphones. In *Proceedings of the 29th ACM International Conference on Multimedia*, pp. 1111–1119, 2021. 1, 2, 3, 8

[3] S. An, X. Zhang, D. Wei, H. Zhu, J. Yang, and K. A. Tsintotas. Fast monocular hand pose estimation on embedded systems, 2021. 4

[4] Björck. *Numerical Methods for Least Squares Problems*. Society for Industrial and Applied Mathematics, 1996. 7

[5] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee. Yolact: Real-time instance segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9156–9165, 2019. 6, 7, 8

[6] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(1):172–186, 2021. 2, 4

[7] Y. Chen, Z. Wang, Y. Peng, Z. Zhang, G. Yu, and J. Sun. Cascaded pyramid network for multi-person pose estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7103–7112, 2018. 2, 4

[8] C.-T. Chou, C.-H. Lee, K. Zhang, H.-C. Lee, and W. H. Hsu. Pivtons: Pose invariant virtual try-on shoe with conditional image completion. In *Proceedings of the Asian Conference on Computer Vision (ACCV)*, pp. 654–668, 2019. 1, 2, 3

[9] P. Eisert, P. Fechteler, and J. Rurainsky. 3-d tracking of shoes for virtual mirror applications. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–6, 2008. 3

[10] Z. Feng, F. Jiang, and R. Shen. Virtual glasses try-on based on large pose estimation. *Procedia Comput. Sci.*, 131(C):226–233, may 2018. 1, 3

[11] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3354–3361, 2012. 6

[12] X. Han, Z. Wu, Z. Wu, R. Yu, and L. S. Davis. Viton: An image-based virtual try-on network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7543–7552, 2018. 1, 2

[13] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu. Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(7):1325–1339, 2014. 6

[14] A. Jimeno-Morenilla, J. L. Sanchez-Romero, and F. Salas-Perez. Augmented and virtual reality techniques for footwear. *Computers in Industry*, 64(9):1371–1382, 2013. 3

[15] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab. SSD-6D: Making rgb-based 3d detection and 6d pose estimation great again. In *IEEE Conf. Comp. Vis.*, pp. 1521–1529, 2017. 3

[16] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference for Learning Representations (ICLR)*, pp. 1–15, 2015. 6

[17] T. Kobayashi, Y. Sugiura, H. Saito, and Y. Uema. Automatic eyeglasses replacement for a 3d virtual try-on system. In *Proceedings of the 10th Augmented Human International Conference 2019*, AH2019. Association for Computing Machinery, New York, NY, USA, 2019. 1, 2, 3

[18] E. Lengyel. *Mathematics for 3D game programming and computer graphics*. Charles River Media, Inc., 2003. 5

[19] J. Li, S. Bian, A. Zeng, C. Wang, B. Pang, W. Liu, and C. Lu. Human pose regression with residual log-likelihood estimation. 2021. 2

[20] Z. Li and F. Zhou. Fssd: feature fusion single shot multibox detector. *arXiv preprint arXiv:1712.00960*, 2017. 4

[21] M. R. Minar, T. T. Tuan, H. Ahn, P. Rosin, and Y.-K. Lai. Cp-vton+: Clothing shape and texture preserving image-based virtual try-on. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2020. 1, 2

[22] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation, 2016. 2, 5

[23] A. Nibali, Z. He, S. Morgan, and L. Prendergast. Numerical coordinate regression with convolutional neural networks. 2018. 2

[24] D. Osokin. Real-time 2d multi-person pose estimation on cpu: Lightweight openpose. *arXiv preprint arXiv:1811.12004*, 2018. 6, 7, 8

[25] N. D. Reddy, M. Vo, and S. G. Narasimhan. Carfusion: Combining point tracking and part detection for dynamic 3d reconstruction of vehicles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1906–1915, 2018. 6

[26] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015. 4, 6

[27] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. 2018. 3, 4

[28] S. Song, S. P. Lichtenberg, and J. Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 567–576, 2015. 6

[29] X. Sun, J. Shang, S. Liang, and Y. Wei. Compositional human pose regression. *Computer Vision and Image Understanding*, 2017. 2

[30] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le. Mnasnet: Platform-aware neural architecture search for mobile. 2018. 4

[31] B. Wang, H. Zheng, X. Liang, Y. Chen, L. Lin, and M. Yang. Toward characteristic-preserving image-based virtual try-on network. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 607–623, 2018. 1, 2

[32] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh. Convolutional pose machines, 2016. 2

[33] S. Xiao, B. Xiao, L. Shuang, and Y. Wei. Integral human pose regression. 2017. 2

[34] H. Yang, R. Zhang, X. Guo, W. Liu, W. Zuo, and P. Luo. Towards photo-realistic virtual try-on by adaptively generating-preserving image content. In *IEEE Conf. Comp. Vis. Pattern Recogn.*, pp. 7850–7859, 2020. 1, 2

[35] Y.-I. Yang, C.-K. Yang, and C.-H. Chu. Virtual try-on of footwear in mixed reality using depth sensors. p. 309–312. Association for Computing Machinery, New York, NY, USA, 2013. 1, 3

[36] H. Zhou, T. Lan, and G. Venkataramani. Pt-vton: an image-based virtual try-on network with progressive pose attention transfer, 2021. 2