Real-Size Experience for Virtual Try-On

FuChe Wu
Providence University
200, Sec. 7, Taiwan Boulevard, Shalu Dist., Taichung City 43301 , Taiwan
fcwu@gm.pu.edu.tw

Wei-Cheng Chen
HOKEI CORP.
No. 103, Daming 1st Rd., Tanzi Dist., Taichung City
427007, Taiwan
frank@mail.hokei.com.tw

Andrew Dellinger
Elon University
100 Campus Drive
Elon, NC 27244
adellinger@elon.edu

Han-Wen Cheng
HOKEI CORP.
No. 103, Daming 1st Rd., Tanzi Dist., Taichung City
427007, Taiwan
eve@mail.hokei.com.tw

ABSTRACT

In this system, we have established an e-commerce platform that allows users to virtually try on products and experience their sizes. The virtual try-on experience goes beyond just exploring the styles of the items; size is a crucial factor. However, achieving consistent sizes between the real and synthesized worlds requires a calibration process. In a virtual try-on e-commerce system, the challenge lies in enabling users to perceive the true-to-life size experience. Camera calibration plays a pivotal role in this process. By obtaining the parameters of the camera, it becomes possible to display accurate sizes that align with the user's proportions. We propose a straightforward calibration method that leverages a mobile web interface to acquire the camera parameters. This approach ensures that users are provided with realistic sizes, enhancing their virtual try-on experience. Additionally, through coordinate transformations, we convert the obtained parameters into the three.js framework, allowing virtual objects to be tried on in a virtual setting. Our system encompasses the virtual try-on of items such as hats, glasses, earrings, rings, and watches. Users have the flexibility to choose different sizes, enabling them to explore various fitting effects. The URL for accessing the demonstration site is https://showcase.id-yours.com/main/id-yours-glasses.

Keywords
Virtual Try-On, Camera Calibration, Augmented Reality

1. INTRODUCTION

The advent of virtual try-on technologies has revolutionized the way consumers experience and engage with e-commerce platforms, particularly in the fashion and accessory industries. Ensuring an authentic real-size experience is crucial for building user confidence and facilitating informed purchasing decisions. In this context, camera calibration plays a pivotal role in accurately representing object sizes within the virtual environment.

This paper introduces a streamlined approach to camera calibration, leveraging a mobile web application for efficient parameter acquisition. By implementing coordinate transformation techniques, the obtained parameters seamlessly integrate with three.js, enabling users to virtually try on accessories such as hats, glasses, earrings, rings, and watches in their true sizes. The system also empowers users to explore different size options, enhancing the overall virtual try-on experience. The proposed methodology addresses the challenges associated with real-size representation in virtual try-on system, contributing to the advancement of immersive and reliable virtual try-on solutions in the e-commerce landscape.

Camera calibration is a fundamental process in computer vision and imaging technology that involves determining the intrinsic and extrinsic parameters of a camera. The intrinsic parameters include focal length, optical center, and lens distortion, while extrinsic parameters involve the position and orientation of the camera in the 3D world. The goal of camera calibration is to establish a mathematical relationship between the 3D world and the 2D image captured by the camera.

2. Previous Work

Due to the maturity of AR filter tools such as Spark AR or Lens Studio, and the stability of deep learning in body feature tracking with technologies like mediapipe, the virtual try-on has witnessed a plethora of commercial applications. Renowned brands such as Prada, Marc Jacobs, L'Oréal, Nike, Baume & Mercier, Ray-Ban, Sephora, among others, have also ventured into online virtual try-on services in recent years. To address the size issue, it is necessary to provide a known object for calibration, such as a credit card. However, this is limited to the moment of capturing the photo, as without camera calibration, the distance between objects cannot be freely adjusted. To solve this issue, some people may opt for depth cameras [Yan14, Aze16]. However, this approach is not as convenient for applications on regular user smartphones. Yu et al. [Yu23] addresses this issue by first calibrating the smartphone. The rear camera captures the environment to determine the phone's position. When waving in front of the front camera to capture a portrait, the person's position can be obtained. However, since smartphone calibration is required beforehand, it may not be suitable for the general user.

Camera calibration has a history spanning several years. The standard practice involves using a known-sized planar target as a calibration tool [Hei97, Stu99, Zha00, Mei07]. However, for the average user, obtaining and using such tools can be challenging. Therefore, we aim to simplify the calibration process by using a straightforward hand-waving gesture, treating our hands as known planar targets for camera calibration.

While deep learning has become prevalent in recent years, with numerous outstanding studies utilizing it for camera calibration[Ken15, Bog18, Lee21, Pon22, Jin23], the majority of successful results are observed in outdoor settings with roads and tall buildings. In indoor scenarios where there are windows, tables, and chairs providing reference points, better outcomes can be achieved. However, when the background consists of blank walls, the obtained calibration values may not be as satisfactory.

.

3. Camera Calibration

FacialSCDnet [Ber22] estimates the distance between the face and the camera based on facial features. Chen et [Che21, Che22] al.'s research, utilizing the Mano [Rom22] project, allows the acquisition of hand shape and size, applicable on mobile devices. Inspired by their work, can we also use a simple waving gesture to calibrate our camera? While the face can also serve as a calibration pattern, instructing users to move their faces to different positions in front of the camera is relatively challenging. However, using hands provides higher degrees of freedom. It allows for variations in height, distance, and angle, minimizing the risk of encountering local minima in the obtained solutions.

As shown in Figure 1, we treat our hand as a planar pattern. Placed at different positions and distances on the screen, it serves to calibrate the camera's parameters. However, since our hand is not truly planar, a plane is used to approximate the landmarks on the hand's features before calibration. These landmarks are then projected onto the plane to obtain the camera's intrinsic and extrinsic parameters.

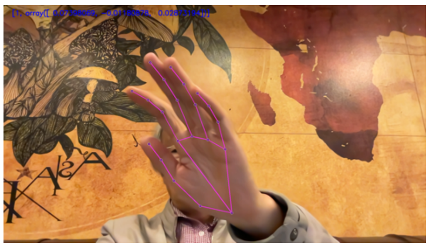


Figure 1. Waving as a calibration pattern

Also, if the user's hand positioning is incorrect, such as the inability to detect a flat plane, we will exclude that frame's image data to avoid affecting the final calibration accuracy.

After obtaining the projection matrix of the camera, we can understand the relationship between the 3D space and its projection in the 2D image space. However, the correct size is still unknown. To address this issue, for a more accurate approach, one can use a credit card as a ruler. If precision is not a critical requirement, estimating finger sizes from training data is an acceptable alternative.

4. Coordinate transform

After obtaining the camera parameters through OpenCV, applying them to OpenGL for rendering involves a coordinate transformation process, as described by Costa et al[Cos19].

As illustrated in Figure 2, the coordinate system on the left represents OpenCV, where y is downward, and z is forward. In the middle, the coordinate system is for OpenGL, with y pointing upward, and -z pointing forward. On the right is the Normalized Device Coordinates after passing through the Clip-Space Frustum.

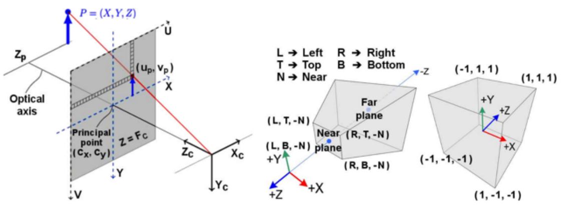


Figure 2. The difference in coordinate systems between OpenCV and OpenGL.

Therefore, we can derive the OpenGL projection matrix from the intrinsic parameters of OpenCV, while also accounting for the axis direction.

Assuming that the intrinsic parameters of OpenCV are

$$K = \begin{bmatrix} f_x & 0 & C_x \\ 0 & f_y & C_y \\ 0 & 0 & 1 \end{bmatrix}$$

Here, f_x and f_y represent the focal lengths in the x and y directions, while c_x and c_y denote the principal points in the x and y directions.

$$P_{opengl} =$$

$$\begin{bmatrix} \frac{2f_x}{width} & 0 & \frac{width - 2C_x}{width} & 0 \\ 0 & \frac{2f_y}{height} & -\frac{height - 2C_y}{height} & 0 \\ 0 & 0 & -\frac{Z_{far} + Z_{near}}{Z_{far} - Z_{near}} & -\frac{2Z_{far}Z_{near}}{Z_{far} - Z_{near}} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

Where P_{opengl} is the OpenGL projection matrix, $width, height$ represent the width and height of the image, respectively. Z_{far}, Z_{near} denote the far and near planes of the frustum, respectively.

Through this relationship, the parameters corrected by OpenCV can easily combine the camera's 2D space with the given spatial coordinates. As shown in Figure 3, we can display virtual cubes using three.js at the intersections of the image pattern.

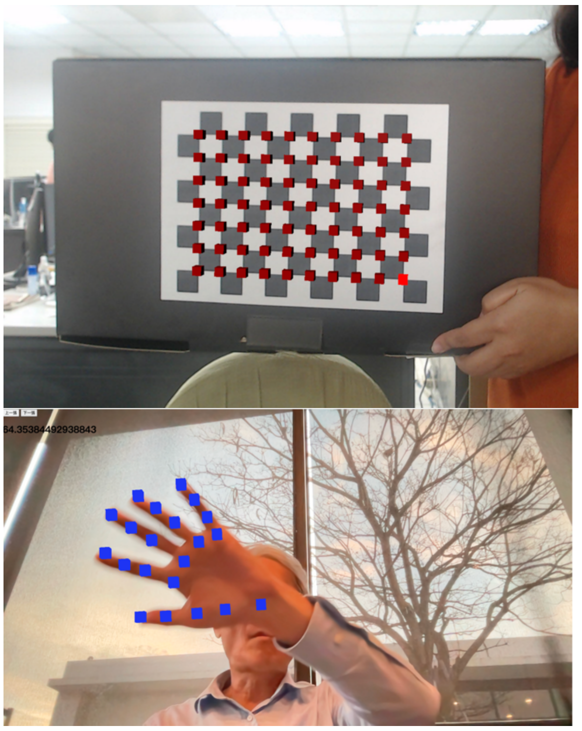


Figure 3. Place a cube at each intersection point after calibration

We also tested the camera calibration method in different environments. As shown in Figure 4, when the ambient light is dim, the virtual cubes displayed in three.js exhibit positional deviations. This indicates that when the camera uses color or brightness-sensitive features during calibration, the strength and quality of the light can affect the accuracy of the calibration.

For this reason, the current solution involves converting the image to grayscale, then calculating the total sum of pixel values to estimate the average light intensity. Following this, each RGB channel is normalized based on this light intensity, and the normalized values are constrained within the valid range (0 to 255). Finally, these channels are merged back together to produce the adjusted image.

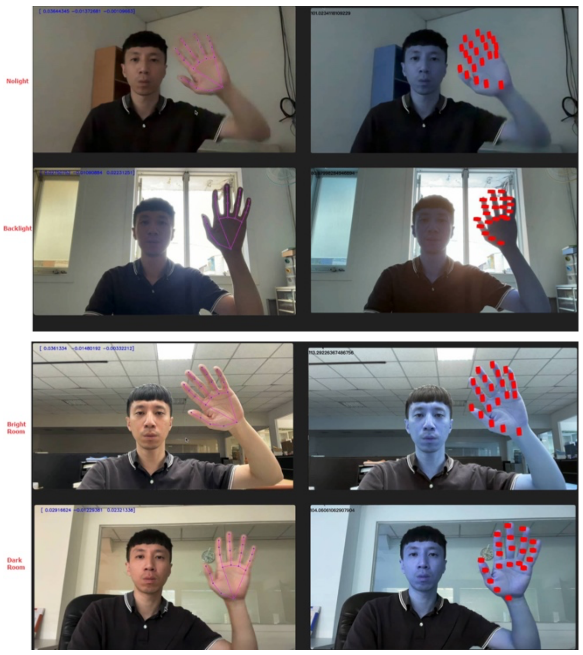


Figure 4. Different environment to place a cube at each intersection point

5. Implementations

After obtaining the calibration parameters of the camera, we can easily integrate virtual objects with images. Our current implementation primarily focuses on wearable items for the head and hands. For the head, we can simulate hats, glasses, and earrings, while for the hands, we have implemented rings and watches. To address occlusion issues when wearing these items, we have incorporated a generic model of a person's head. This virtual head model serves three purposes: first, to determine the alignment between virtual objects and a real person's head; second, to establish the relative positioning between wearable items and the person; and third, to address occlusion relationships between wearable items and the person's head.

To achieve efficient inference on mobile devices or web applications, we have adopted the Mediapipe solution. Within this framework, the entire process, from image preprocessing, head tracking, landmark detection, to the inference of the three-dimensional position of the head, is structured as a set of calculators. These calculators can be organized into a graph pipeline, and the results are encapsulated as WebAssembly for utilization in the front-end of web applications. We have established a WebGL environment and employed three.js for rendering.

The head itself is a rigid body, with facial landmarks using the nose as the local origin. Apart from facial expressions, there aren't many degrees of freedom. Therefore, simulations for trying on hats, glasses, and earrings on the head are relatively straightforward. Obtaining the transformation for try-on points, whether in translation or rotation, poses minimal challenges, as shown in Figures 5, 8, and 9.

Also, the detectable range starts from the camera and extends up to 2 meters for accurately recognizing a complete face, which is considered the normal tracking range. From 2 to 4 meters, the stability of detection gradually decreases. Beyond 4 meters, tracking will fail.

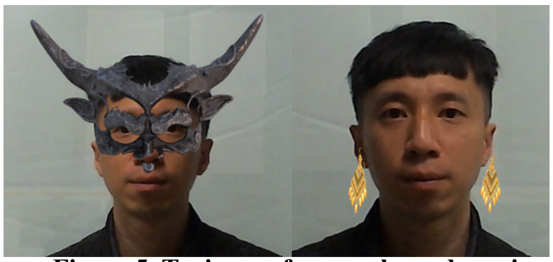


Figure 5. Trying on face masks and earring

In addition, we have incorporated physics simulation into the earring try-on application, sequence of images as shown in Figures 6.

We created an earring model using bones of a 3D model, associating various parts of the earring with the bone to control its deformation and movement. Next, rigidbody components were added to the earring model to impart physical properties such as gravity and collision in Figure 7. Finally, constraints were applied to restrict the movement of the earring model in the physics simulation, such as adding rotational constraints to mimic realistic swaying effects. In this physics simulation, the skeleton is used to define the shape and structure of the earring. The rigidbody represents the physical properties of the earring, such as mass, inertia, and collision attributes. Constraints are used to define the relationship and constraints between the earring and the head or other objects, such as how the earring should connect to the head or whether the earring can rotate. This application of physics simulation enhances the realism and immersion of the try-on experience, allowing users to better experience the appearance and feel of the earrings in a real environment.

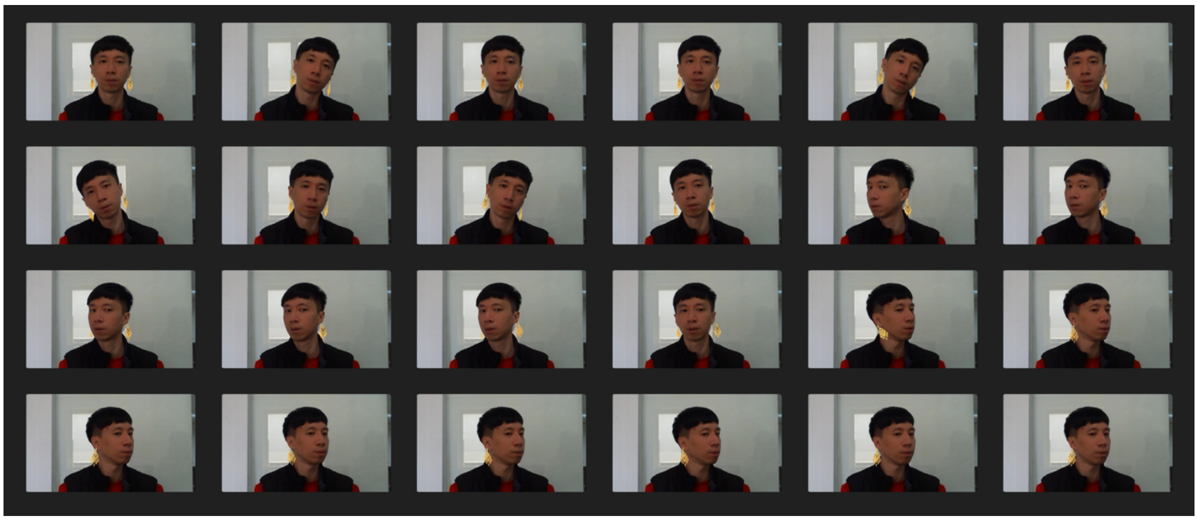
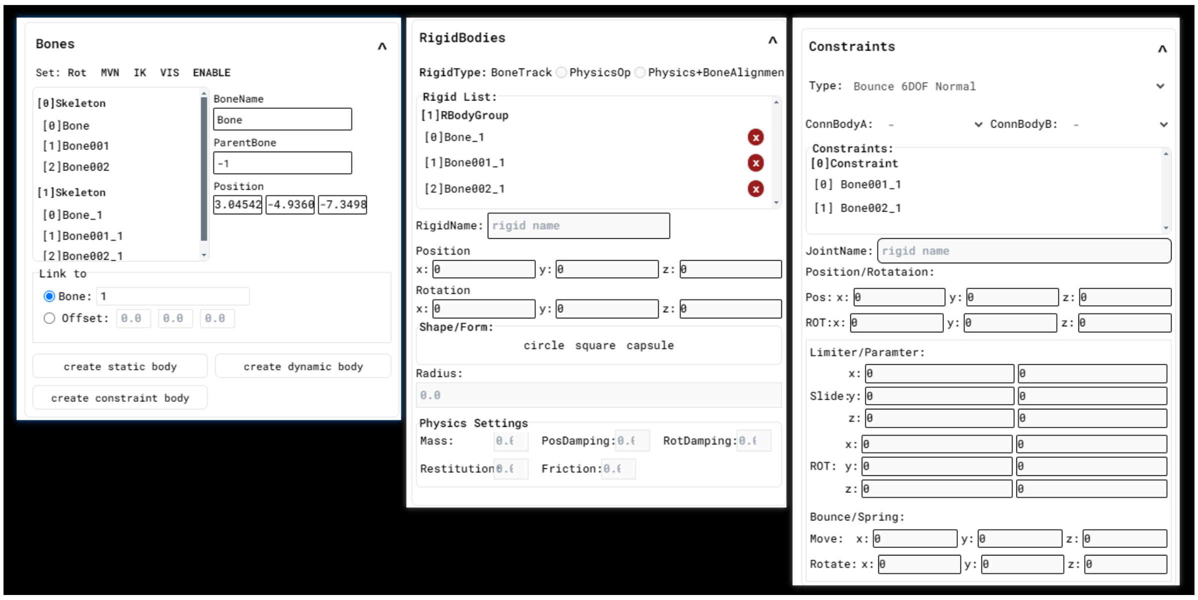


Figure 6. Earring physics simulation



```
Bones                                                  ^
Set: Rot  MVN  IK  VIS  ENABLE
[0]Skeleton              BoneName
 [0]Bone                 Bone
 [1]Bone001              ParentBone
 [2]Bone002              -1
[1]Skeleton              Position
 [0]Bone_1               3.04542  -4.9360  -7.3498
 [1]Bone001_1
 [2]Bone002_1
Link to
 (●) Bone: 1
 ( ) Offset: 0.0  0.0  0.0
create static body       create dynamic body
create constraint body

RigidBodies                                            ^
RigidType: BoneTrack ( ) PhysicsOp ( ) Physics+BoneAlignmen
Rigid List:
[1]RBodyGroup
 [0]Bone_1
 [1]Bone001_1
 [2]Bone002_1
RigidName: rigid name
Position
x: 0          y: 0          z: 0
Rotation
x: 0          y: 0          z: 0
Shape/Form:
        circle  square  capsule
Radius:
0.0
Physics Settings
Mass:       0.(   PosDamping:0.(   RotDamping:0.(
Restitution0.(   Friction:0.(

Constraints                                            ^
Type:  Bounce 6DOF Normal
ConnBodyA:  -          ConnBodyB:  -
Constraints:
[0]Constraint
 [0] Bone001_1
 [1] Bone002_1
JointName: rigid name
Position/Rotataion:
Pos: x: 0       y: 0       z: 0
ROT:x: 0        y: 0       z: 0
Limiter/Paramter:
        x: 0          0
  Slide:y: 0          0
        z: 0          0
        x: 0          0
  ROT:  y: 0          0
        z: 0          0
Bounce/Spring:
Move:   x: 0     y: 0     z: 0
Rotate: x: 0     y: 0     z: 0
```

Figure 7. Physics configuration

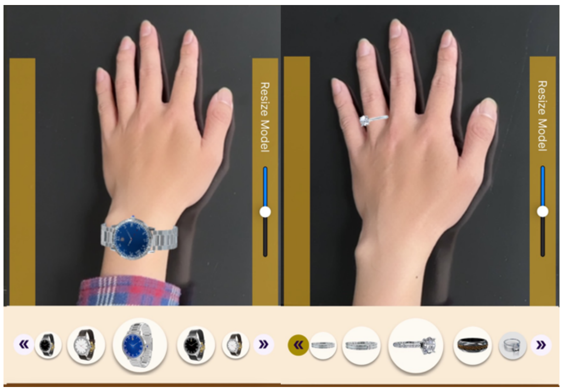


Figure 8. Trying on watches and rings

However, when it comes to trying on accessories on the hands, the presence of numerous joints introduces increased degrees of freedom. For example, in the case of trying on a ring, we not only need to acquire the position of the metacarpophalangeal (MCP) joint but also determine its rotation. The approach involves obtaining the positions of the wrist, index finger MCP joint, and pinky MCP joint—considering these three landmarks as representing the palm and treating them as a plane. Subsequently, we calculate the rotation of the ring finger relative to this plane. Calculating the angle between two vectors can be done using the arccosine (inverse cosine) function, which is based on the law of cosines. As illustrated in Figure 8, this method demonstrates the result of trying on a ring.

The hand landmarks do not include the wrist part, making trying on a watch quite challenging. To determine the fitting point of the watch and its rotation angle, we segmented the skin area using the Mediapipe Selfie Segmentation solution. Next, we applied Zhang et al.'s thinning method in OpenCV to obtain skeletons, allowing us to find suitable points as try-on candidate points. The search starts from the unique position of hand landmark 0, exploring skeleton points based on their neighborhood relationships. An ideal try-on point must satisfy two conditions: being distant from other landmarks (i.e., not in the direction of the palm and fingers) and having sufficient distance. By considering these two conditions, we can identify the most suitable point. The identified dorsal part of the hand, its skeleton, and the final try-on point are illustrated in Figure 9. Once this point is found, we extract a 3D point from the pose skeleton, whose projection onto the 2D image closely matches our identified 2D try-on point. The orientation of the watch face must be perpendicular to the tangent direction of the skeleton and closest to the normal vector of the back of the hand. The concept is illustrated in Figure 10.

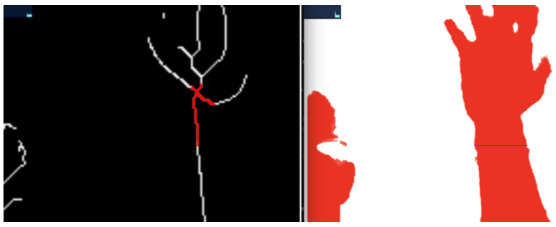


Figure 9. Locating the try-on point for watches from the hand's skeleton

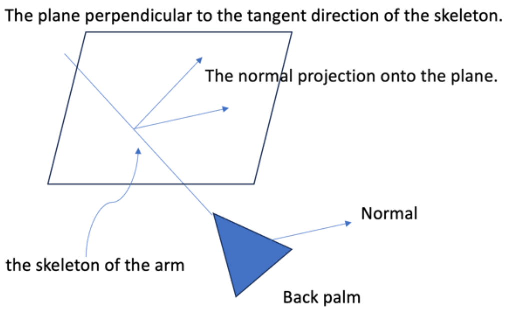

The plane perpendicular to the tangent direction of the skeleton.
The normal projection onto the plane.
Normal
the skeleton of the arm
Back palm

Figure 10. The normal vector of the back of the hand projected onto the plane perpendicular to the skeleton.

6. Result

To ensure consistency between the try-on size and the real size, we create virtual objects in proportion to their real-world counterparts. When trying on virtual objects, we can also wear real-world objects of known size to verify the size consistency between the real and virtual. As shown in Figure 11, we first try on virtual glasses and then wear real glasses on top. We can observe that the sizes of both are perfectly consistent.

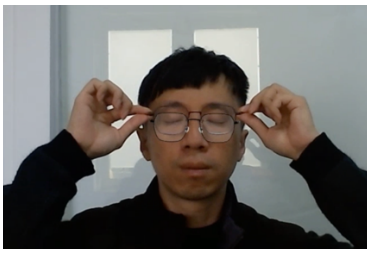


Figure 11. Comparing the size difference between real and virtual glasses

In Figure 12, we demonstrate trying on hats of different sizes. As shown in the display, we can observe the visual differences between wearing a smaller hat and a larger hat. By capturing two images at the same position, we can see that the lower edge of the larger hat appears larger than that of the smaller hat.

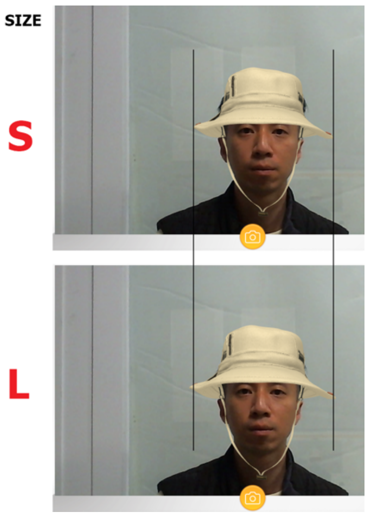


Figure 12. Comparing synthesized hats of different sizes

The current system still has some imperfections that can be improved. For example, when the user's head rotation angle is relatively large, as shown in Figure 13, the occlusion effect on the hair and nose is not well-executed, resulting in some flaws being visible. To address this issue, we plan to adopt recent advancements in image generation techniques. For instance, when generating portraits, Wang et al. [Wan23] faced challenges with hair, and they addressed this by employing UNet to refine the final results. However, UNet can be computationally intensive. While Jia et al. [Jia23]demonstrated its application on mobile devices, it might not be entirely suitable for real-time synthesis. We are currently working on how to achieve real-time performance for this solution.

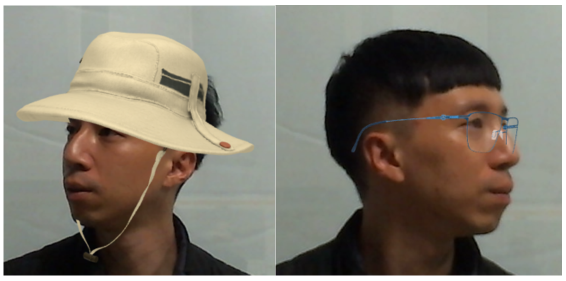


Figure 13. Imperfections during head rotation

Since our system is implemented in the web frontend, it is compatible with various operating systems such as iOS, Android, and Windows, and browsers like Chrome or Safari. Through testing on individuals with different body shapes, both males and females, our system consistently provides a realistic sense of size when interacting with the camera.

7. Conclusion

In the context of augmented reality applications, a better understanding of the real world leads to more accurate synthesis of virtual objects that match the real-world environment. In this paper, we propose a straightforward method for calibrating the user's camera, allowing for consistent results in terms of synthesized and real object sizes, regardless of their dimensions.

Furthermore, we demonstrate how to configure the virtual camera's projection matrix after obtaining the camera's intrinsic parameters. This ensures a consistent performance between the real and virtual cameras, enhancing the natural appearance of synthesized objects in the real environment.

When it comes to determining reference points for synthesized objects, we present an efficient method for reverse engineering 3D positions and orientations from 2D image reference points. Leveraging proximity relationships from the segmentation's skeleton, we locate the necessary points and find suitable solutions in 3D through vector projection.

Also, since virtual try-on will involve issues of portrait rights and privacy, to avoid controversy over user data being transmitted to the server, we also use MobileNet for training the AI model, allowing the AI model to run on the client-side, and it can even be used offline.

Lastly, we integrate a virtual try-on system, showcasing headwear trials for hats, glasses, and earrings, as well as hand trials for rings and watches. We demonstrate trial results of different sizes and compare them with real-size trials. Cross-testing on various devices and users validates the system's ability to provide a realistic sense of size in the try-on results.

8. ACKNOWLEDGMENTS

This work was supported by the National Science and Technology Council, Taiwan, R.O.C., under grant no. NSTC 112-2221-E-126-003.

9. REFERENCES

[Yan14] Yang, Y.-I., Yang, C.-K. and Chu, C.-H. A virtual try-on system in augmented reality using RGB-D cameras for footwear personalization. *Journal of Manufacturing Systems*, 33, 4 (2014), 690-698.

[Aze16] Azevedo, P., Dos Santos, T. O. and De Aguiar, E. *An augmented reality virtual glasses try-on system*. IEEE, City, 2016.

[Yu23] Yu, R., Wang, J., Ma, S., Huang, S. X., Krishnan, G. and Wu, Y. *Be Real in Scale: Swing for True Scale in Dual Camera Mode*. IEEE, City, 2023.

[Hei97] Heikkila, J. and Silvén, O. *A four-step camera calibration procedure with implicit image correction*. IEEE, City, 1997.

[Stu99] Sturm, P. F. and Maybank, S. J. *On plane-based camera calibration: A general algorithm, singularities, applications*. IEEE, City, 1999.

[Zha00] Zhang, Z. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22, 11 (2000), 1330-1334.

[Mei07] Mei, C. and Rives, P. *Single view point omnidirectional camera calibration from planar grids*. IEEE, City, 2007.

[Ken15] Kendall, A., Grimes, M. and Cipolla, R. *Posenet: A convolutional network for real-time 6-dof camera relocalization*. City, 2015.

[Bog18] Bogdan, O., Eckstein, V., Rameau, F. and Bazin, J.-C. *DeepCalib: A deep learning approach for automatic intrinsic calibration of wide field-of-view cameras*. City, 2018.

[Lee21] Lee, J., Go, H., Lee, H., Cho, S., Sung, M. and Kim, J. *Ctrl-c: Camera calibration transformer with line-classification*. City, 2021.

[Pon22] Ponimatkin, G., Labbé, Y., Russell, B., Aubry, M. and Sivic, J. *Focal length and object pose estimation via render and compare*. City, 2022.

[Jin23] Jin, L., Zhang, J., Hold-Geoffroy, Y., Wang, O., Blackburn-Matzen, K., Sticha, M. and Fouhey, D. F. *Perspective Fields for Single Image Camera Calibration*. City, 2023.

[Ber22] Bermejo, E., Fernandez-Blanco, E., Valsecchi, A., Mesejo, P., Ibáñez, O. and Imaizumi, K. FacialSCDnet: A deep learning approach for the estimation of subject-to-camera distance in facial photographs. *Expert Systems with Applications*, 210 (2022), 118457.

[Che21] Chen, X., Liu, Y., Ma, C., Chang, J., Wang, H., Chen, T., Guo, X., Wan, P. and Zheng, W. *Camera-space hand mesh recovery via semantic aggregation and adaptive 2d-1d registration*. City, 2021.

[Che22] Chen, X., Liu, Y., Dong, Y., Zhang, X., Ma, C., Xiong, Y., Zhang, Y. and Guo, X. *Mobrecon:*

Mobile-friendly hand mesh reconstruction from monocular image. City, 2022.

[Rom22] Romero, J., Tzionas, D. and Black, M. J. Embodied hands: Modeling and capturing hands and bodies together. *arXiv preprint arXiv:2201.02610* (2022).

[Cos19] Costa, C. M., Veiga, G., Sousa, A., Rocha, L., Sousa, A. A., Rodrigues, R. and Thomas, U. *Modeling of video projectors in OpenGL for implementing a spatial augmented reality teaching system for assembly operations*. IEEE, City, 2019.

[Wan23] Wang, L., Zhao, X., Sun, J., Zhang, Y., Zhang, H., Yu, T. and Liu, Y. StyleAvatar: Real-time Photo-realistic Portrait Avatar from a Single Video. *arXiv preprint arXiv:2305.00942* (2023).

[Jia23] Jia, H., Wang, Q., Tov, O., Zhao, Y., Deng, F., Wang, L., Chang, C.-L., Hou, T. and Grundmann, M. *BlazeStyleGAN: A Real-Time On-Device StyleGAN*. City, 2023.