Be Real in Scale: Swing for True Scale in Dual Camera Mode

Rui Yu[1]* Jian Wang[2] Sizhuo Ma[2] Sharon X. Huang[1] Gurunandan Krishnan[2] Yicheng Wu[2]

[1]The Pennsylvania State University [2]Snap Inc.

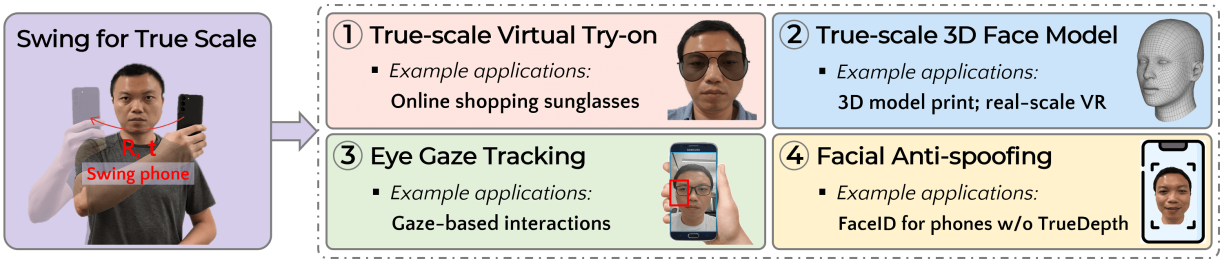


Figure 1: We introduce a novel approach for a user to estimate the metric scale of the face: simply swing the phone in front of the face. Our app uses the *Dual Camera mode* to capture selfies and scene images with the front and rear cameras simultaneously. The proposed algorithm leverages the scene images to resolve the scale ambiguity and then calculates the absolute scale of the face's geometry. Our method is easy to use and as accurate as TrueDepth-based methods, which enables a wide range of applications for smartphones without TrueDepth sensors, as shown in sub-figures 1-4.

Abstract

Many mobile AR apps that use the front-facing camera can benefit significantly from knowing the metric scale of the user's face. However, the true scale of the face is hard to measure because monocular vision suffers from a fundamental ambiguity in scale. The methods based on prior knowledge about the scene either have a large error or are not easily accessible. In this paper, we propose a new method to measure the face scale by a simple user interaction: the user only needs to swing the phone to capture two selfies while using the recently popular Dual Camera mode. This mode allows simultaneous streaming of the front camera and the rear cameras and has become a key feature in many social apps. A computer vision method is applied to first estimate the absolute motion of the phone from the images captured by two rear cameras, and then calculate the point cloud of the face by triangulation. We develop a prototype mobile app to validate the proposed method. Our user study shows that the proposed method is favored compared to existing methods because of its high accuracy and ease of use. Our method can be built into Dual Camera mode and can enable a wide range of applications (*e.g.*, virtual try-on for online shopping, true-scale 3D face modeling, gaze tracking, and face anti-spoofing) by introducing true scale to smartphone-based XR. The code is available at `https://github.com/ruiyu0/Swing-for-True-Scale`.

Index Terms: Computing methodologies—Computer vision; Human-centered computing—Interaction design

1 Introduction

Knowing the true scale of the user's face is crucial for many mobile apps. For example, in AR-based virtual try-on experiences during online shopping, having knowledge of the actual face scale enables customers to visually determine the correct size without the need for manual measurements. Moreover, the scale information aids in constructing an accurate 3D face model, essential for rendering real-scale avatars in Augmented Reality (AR) or Virtual Reality (VR). Other applications include eye gaze tracking, face anti-spoofing, etc.

However, it is well-known in computer vision that, with a single front-facing camera, there is an ambiguity to estimate the scale given the unknown distance between the camera and the face. To obtain the true scale, extra information or sensors are needed. The methods that rely on extra prior information are either not accurate (*e.g.*, assuming known iris size) or not easily accessible (*e.g.*, requiring extra tools such as a ruler or a bank card). In terms of extra sensors, latest Apple iPhone are equipped with a TrueDepth sensor that allows for facial scale estimation. In contrast, the majority of Android phones lack front-facing depth sensors. Another alternative involves utilizing inertial measurement units (IMUs) [14, 15, 26] to estimate accurate motion scale. But this method requires the phone to be in motion for a long duration to obtain a measurement with adequate precision.

In this work, we propose a simple approach to recover the true scale of the face utilizing the *Dual Camera mode*. Dual Camera Mode has been a popular feature in multimedia social media apps, *e.g.*, Snapchat [36], Instagram [24], and BeReal [3]. In this mode, images from both front-facing and rear-facing cameras are recorded simultaneously and combined together for creative AR effects. The popularity of Dual Camera mode is enabled by the recent development in both mobile hardware and software. On one hand, most recent smartphones have both front-facing and rear-facing cameras; on the other hand, system-level support of streaming multiple cameras simultaneously becomes available on both iOS and Android platforms for the first time.

The user operation of our method is simple: the user only needs to hold the phone on the left and take a selfie, swing their phone to the right, and take a second selfie (or swing from right to left depending on the user's preference). If the motion of the front camera between the two selfies is known, triangulation can be utilized to calculate the true scale of the face. Our key observation is that the front and rear cameras are rigidly attached together on the phone, which means that their motion will be the same. Notice that it is much easier to estimate the motion of the rear cameras because: (1) rear cameras capture the surrounding world with a larger field-of-view (FOV), most of which can be assumed static and provides a reference for motion estimation; (2) many mainstream smartphones have at least two back-facing cameras (*e.g.*, main camera, ultrawide camera,

*Rui Yu's work on this paper was partially done during his internship at Snap Research. E-mails: {rzy54, suh972}@psu.edu, {jwang4, sma, guru}@snap.com, wuyichengg@gmail.com.

telephoto camera), which can provide the true scale of the motion. Specifically, we use two rear cameras to reconstruct the true-scale 3D point cloud of the back scene based on stereo computer vision. We track the true-scale motion of the rear cameras by matching the 3D point cloud reconstructed before and after the swing. The estimated motion is then transferred to the front camera and used to recover the true scale of the face.

We developed a prototype to validate the proposed method. The prototype includes an app that can simultaneously capture three (one front-facing and two back-facing) images and an algorithm to estimate the face scale from two poses by swinging the phone. More specifically, we applied our method to measure pupillary distance (PD), which is essential for crafting prescription glasses and can be integrated into the process of true-scale virtual glasses try-on. We conduct a user study with 16 participants to evaluate the measurement accuracy and the user experience. The mean error of PD estimation is only 0.9 *mm*, showing the feasibility and applicability of the proposed method.

The main contributions of this paper are as follows:

- We propose a new method for estimating the true scale of objects (*e.g.*, face) in Dual Camera mode and demonstrate the potential applications in AR/VR, online shopping, gaze-based interactions, and biometric authentication.
- We implement a prototype to estimate face scale with a multi-camera smartphone, and further improve our algorithm's robustness to face motion by leveraging face prior.
- We conduct a user study and demonstrate the feasibility and applicability of the proposed method.

2 Related Work

2.1 Scale Estimation with Smartphone

Scale estimation is a fundamental problem in computer vision. 3D scenes can be reconstructed from a single image or multiple images using Structured from Motion (SfM), up to a scale ambiguity. However, the accurate scale is crucial for AR/VR experiences such as shopping for glasses with the actual size and immersive interaction with full-size avatars, as well as 3D printing with the precise size of the scanned object.

Due to the scale ambiguity, extra references or sensors are required to recover the scale. A reference object (e.g., a ruler, a credit card, or the iris of an eye) in the image has a known scale that can be used to infer the scale of the rest of the image. Recovering the scale using references typically requires extra items or statistical information. In terms of extra sensors, stereo cameras, inertial measurement units (IMUs), and depth sensors have been used. Ham *et al*. [14] and follow-up works [15, 26] use an IMU to estimate the phone true-scale motion. It requires a long period of time to get a sufficiently accurate measurement. For example, [14] needs 68s to estimate the pupil distance, which limited its use cases. Apple iPhone X and later models are equipped with a TrueDepth selfie camera (*i.e.*, FaceID). The face scale can be estimated based on this depth sensor. But the existence of the TrueDepth camera worsens the full-screen-displaying experience, and most Android phones do not have front depth sensors. Our method is proposed upon Dual Camera mode. The required cameras are already available in most popular Android and iPhone models.

2.2 Multi-Camera Applications

Over the last decade, we have seen a clear trend of equipping multiple cameras in smartphones, including both iOS and Android systems. This enables new applications that are not feasible using a single camera. Based on the orientation of cameras, related work can be grouped into two categories. The first group merges the images from two rear cameras. Abdelhamed *et al*. [1] propose to predict scene illumination based on the different spectral sensitivities of two cameras' sensors. Lai *et al*. [20] build a face deblurring algorithm relying on the different shutter speeds. The second group demonstrates new features with both front and rear cameras. By combining photos from the front and rear cameras, Cheng *et al*. [7] are able to reconstruct a better environment map, which is helpful for virtual object rendering. Nam *et al*. [27] showcase the application in health. The breathing rate is estimated from the front camera video, while the heart rate is measured from signals of a fingertip placed in contact with the rear camera simultaneously.

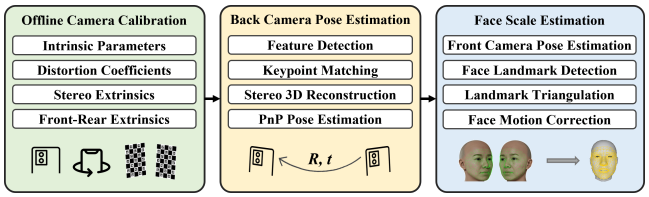


Figure 2: Overview of "Swing-for-True-Scale" method

To the best of our knowledge, we are the first paper utilizing three cameras (including two rear cameras and one front camera) to demonstrate an important application in metric scale estimation. On the one hand, the two rear cameras capture the same back scene, which enables recovery of the absolute camera pose. On the other hand, the rear cameras and the front camera have the same motion, which allows us to know the camera pose of the front camera.

3 Swing-for-True-Scale Algorithm

3.1 Overview

This study focuses on how to estimate the true scale of the face in a smartphone's selfie mode. Since a single camera suffers from scale ambiguity [16], additional sensors are needed. One simple method is to utilize a front-facing depth sensor. However, among all the models released in 2022 [12], only iPhone 14 series and Huawei Mate P50 Pro are equipped with a front-facing depth sensor. While other models do not have a front-facing depth camera, most of them are equipped with at least two rear cameras. Based on this observation, we propose a simple "Swing-for-True-Scale" algorithm to obtain the true-scale 3D face using front and back cameras simultaneously.

The user operation is simple: first take a selfie image in Pose 1, then slightly swing the phone to Pose 2 and take another selfie, as illustrated in Figure 1. During taking selfies, the two rear cameras are also capturing images of the back scene simultaneously. The key idea of the proposed method is to estimate the true-scale phone motion through the rear dual cameras, as the scale ambiguity can be resolved by the stereo cue. The real-size 3D face can then be reconstructed based on the phone's motion and the two selfies. For better 3D face reconstruction, the face keeps static during the "swing" operation. As shown in Figure 2, the method involves three components:

- **Offline Camera Calibration.** Camera calibration estimates the intrinsic parameters and distortion coefficients of each camera, the stereo extrinsics (*i.e.*, rotation and translation) of two rear cameras, and the front-rear extrinsics between the front and rear cameras.
- **Back Camera Pose Estimation.** In Pose 1, we reconstruct a real-scale 3D point cloud of back scene from the captured stereo rear images. Then in Pose 2, we match keypoints between the images of Pose 1 and Pose 2 to establish a correspondence between the keypoints of Pose 2 and the 3D point cloud. Based on this 2D-3D correspondence, we can estimate the relative pose between Pose 2 and Pose 1 by solving a Perspective-n-Point (PnP) problem.
- **Face Scale Estimation.** This component first transfers the transformation of back cameras to that of front camera with calibrated front-rear extrinsics. Then, we detect predefined face landmarks

from the two selfie images to reconstruct true-scale 3D face landmarks based on the true-scale camera transformation. In the case of face motion, we also proposed a correction algorithm to rectify the scale estimation.

3.2 Offline Camera Calibration

The camera on a smartphone can be represented by a pinhole camera model with lens distortions. We use $\mathbf{K}$ to represent the intrinsic parameters. The extrinsic parameters includes a 3×3 orthogonal rotation matrix $\mathbf{R} \in SO(3)$ and translation $\mathbf{t} \in \mathbb{R}^3$. In practice, the lens introduces image distortions, which consist of radial distortion and tangential distortion in Brown–Conrady's model [9] with five distortion parameters $\mathbf{k} = (k_1, k_2, k_3, p_1, p_2)$.

There are three cameras utilized in our method: a front camera C_f and two rear cameras C_{r_1}, C_{r_2}. For convenience, we regard C_{r_1} as the main camera. The parameters that need to be calibrated include the intrinsic parameters ($\mathbf{K}_f$, $\mathbf{K}_{r_1}$, $\mathbf{K}_{r_2}$) and distortion coefficients ($\mathbf{k}_f$, $\mathbf{k}_{r_1}$, $\mathbf{k}_{r_2}$) of each camera, the transformation $\mathbf{M}_{r_1 \to r_2}$ between C_{r_1} and C_{r_2} (stereo extrinsics), and the transformation $\mathbf{M}_{r_1 \to f}$ between C_{r_1} and C_f (front-rear extrinsics). For ease of presentation, we employ 4×4 homogenous transformation matrix representation $\mathbf{M}_* = \left[\begin{smallmatrix} \mathbf{R}_* & \mathbf{t}_* \\ 0 & 1 \end{smallmatrix}\right]$, where $*$ denotes any superscripts and/or subscripts.

3.2.1 Single Camera Calibration

We first adopt the classic single-camera calibration method [44] to obtain the intrinsic parameters and distortion coefficients of each camera. Specifically, we take a few images of a predefined pattern (*e.g.*, 9×6 chessboard with a grid size of 35 *mm*) and detect specific points on the pattern (*e.g.*, square corners on the chessboard). Given the real-world coordinates (in meter units) and the corresponding image coordinates (in pixel units) of these points, we can estimate the intrinsic camera parameters, distortion coefficients, and extrinsic parameters of each image by minimizing the reprojection error of the points using Levenberg-Marquardt optimization algorithm.

3.2.2 Cross-Camera Calibration

We fix the intrinsic parameters and distortion coefficients of the cameras and estimate $\mathbf{M}_{r_1 \to r_2}$ and $\mathbf{M}_{r_1 \to f}$ using a generic multi-camera calibration toolbox [30]. Specifically, we place two predefined patterns (*e.g.*, chessboards) in front of and behind the phone and take a few sets of synchronized images using the three cameras. Since there is a large overlapping FOV between the two rear cameras, the stereo calibration of $\mathbf{M}_{r_1 \to r_2}$ is a straightforward extension of single-camera calibration [5, 44]. The calibration of the transformation $\mathbf{M}_{r_1 \to f}$ between the front C_f and rear C_{r_1} cameras with non-overlapping FOV is based on a classic linear hand-eye calibration strategy [39]. The estimated $\mathbf{M}_{r_1 \to r_2}$ and $\mathbf{M}_{r_1 \to f}$ are then jointly refined via bundle adjustment. Once the calibration is complete, all parameters are fixed and we can reuse them at later stages.

3.3 Back Camera Pose Estimation

The goal of this component is to estimate the true-scale relative pose $\mathbf{M}_{r_1}^{t_1 \to t_2}$ of C_{r_1} swinging from Pose 1 (t_1) to Pose 2 (t_2) using the images $\{\mathbf{I}_{r_1}^{t_1}, \mathbf{I}_{r_2}^{t_1}, \mathbf{I}_{r_1}^{t_2}, \mathbf{I}_{r_2}^{t_2}\}$ captured by the dual rear cameras $\{C_{r_1}, C_{r_2}\}$ at Pose 1 and Pose 2, as well as the calibration parameters.

3.3.1 Feature Detection and Matching

First, we use the stereo images $\{\mathbf{I}_{r_1}^{t_1}, \mathbf{I}_{r_2}^{t_1}\}$ at Pose 1 to reconstruct a true-scale 3D point cloud of the back scene. To this end, we need to find some pairs of matched keypoints between the two images. Specifically, we adopt the SIFT algorithm [22] to detect keypoints on the two images separately. Each keypoint corresponds to a 128-dimensional descriptor. For each keypoint, we adopt FLANN algorithm [25] to find nearest neighbor keypoints on the other image with smallest Euclidean distances of the corresponding descriptors. For the initial matches, we apply Lowe's ratio test [22] to preserve good matches where the distance ratio between the two nearest matches of a keypoint is below a threshold (0.7 in our experiments). In this way, we can identify n pairs of matched keypoints $\{\mathbf{p}_i^{t_1,r_1} \leftrightarrow \mathbf{p}_i^{t_1,r_2} | i = 1, 2, \ldots, n\}$ on the stereo images.

3.3.2 Stereo 3D Reconstruction

Taking $\mathbf{I}_{r_1}^{t_1}$ as an example, the matched keypoints $\{\mathbf{p}_i^{t_1,r_1} | i = 1, 2, \ldots, n\}$ are 2D points in pixel units on the image. To reconstruct 3D points in physical world, we first convert each keypoint $\mathbf{p}_i^{t_1,r_1}$ to the normalized coordinate $\tilde{\mathbf{p}}_i^{t_1,r_1}$ in meter units by performing undistortion and reverse perspective transformation using intrinsic parameters $\mathbf{K}_{r_1}$ and distortion coefficients $\mathbf{k}_{r_1}$. Similarly, the keypoints on the other image $\mathbf{I}_{r_2}^{t_1}$ can also be converted to normalized coordinates, resulting in n pairs of matched points in world coordinate $\{\tilde{\mathbf{p}}_i^{t_1,r_1} \leftrightarrow \tilde{\mathbf{p}}_i^{t_1,r_2} | i = 1, 2, \ldots, n\}$. Then, we can triangulate the 3D point $\mathbf{P}_i$ for each correspondence $\{\tilde{\mathbf{p}}_i^{t_1,r_1} \leftrightarrow \tilde{\mathbf{p}}_i^{t_1,r_2}\}$ using calibrated $\mathbf{M}_{r_1 \to r_2}$. We adopt the direct linear transformation (DLT) algorithm [16] to triangulate the 3D points, resulting in a point cloud $\{\mathbf{P}_i | i = 1, 2, \ldots, n\}$. We can further identify the outliers by reprojecting the 3D points to the image planes. If the reprojection error is greater than a threshold (*e.g.*, 8 pixels), we treat it as an outlier. After removing outliers, the remaining n' 3D points are the point cloud $\{\mathbf{P}_i | i = 1, 2, \ldots, n'\}$ we reconstructed in Pose 1.

3.3.3 Camera Pose Estimation

In Pose 2, the dual rear cameras also capture a pair of images $\{\mathbf{I}_{r_1}^{t_2}, \mathbf{I}_{r_2}^{t_2}\}$. As in Pose 1, we extract the SIFT features on $\mathbf{I}_{r_1}^{t_2}$ and match them with the keypoints on $\mathbf{I}_{r_1}^{t_1}$. We only keep the matches where the keypoints on $\mathbf{I}_{r_1}^{t_1}$ are also used to reconstruct the point cloud $\{\mathbf{P}_i | i = 1, 2., \ldots, n'\}$, resulting in n'' pairs ($n'' \leqslant n'$) of 2D keypoints matches $\{\mathbf{p}_i^{t_1,r_1} \leftrightarrow \mathbf{p}_i^{t_2,r_1} | i = 1, 2, \ldots, n''\}$. They are then converted to normalized coordinates $\{\tilde{\mathbf{p}}_i^{t_1,r_1} \leftrightarrow \tilde{\mathbf{p}}_i^{t_2,r_1} | i = 1, 2, \ldots, n''\}$ using $\mathbf{K}_{r_1}$ and $\mathbf{k}_{r_1}$. Since $\tilde{\mathbf{p}}_i^{t_1,r_1}$ also corresponds to a 3D point $\mathbf{P}_i$ in the point cloud, we can establish the 2D-3D correspondence $\{\tilde{\mathbf{p}}_i^{t_2,r_1} \leftrightarrow \mathbf{P}_i | i = 1, 2, \ldots, n''\}$. Estimating $\mathbf{M}_{r_1}^{t_1 \to t_2}$ according to this 2D-3D correspondence is a Perspective-n-Point (PnP) problem. It can be formulated as finding the optimal rotation matrix $\tilde{\mathbf{R}}$ and translation vector $\tilde{\mathbf{t}}$ to minimize the summed squared projection errors of the n'' points [38]:

$$\min_{\mathbf{R},\mathbf{t}} \sum_{i=1}^{n''} \left\| \tilde{\mathbf{p}}_i^{t_2,r_1} - \frac{\mathbf{R}\mathbf{P}_i + \mathbf{t}}{\mathbf{1}_z^T (\mathbf{R}\mathbf{P}_i + \mathbf{t})} \right\|^2, \tag{1}$$

where $\mathbf{1}_z = [0, 0, 1]^T$. This minimization problem can be solved iteratively by Levenberg-Marquardt optimization.

According Eq. (1), the optimization is based on all n'' points. If the keypoints are highly unbalanced on the image (*e.g.*, much more points in a small region than others), the pose estimation would be biased to feature-dense regions and tend to introduce a large estimation error. We found in our experiments that feature binning (a popular trick in visual odometry [31]), which divides the image into many grids and limits a maximum number of keypoints in each grid before optimization, could make the pose estimation much more stable. The final estimated camera pose is $\mathbf{M}_{r_1}^{t_1 \to t_2} = \left[\begin{smallmatrix} \tilde{\mathbf{R}} & \tilde{\mathbf{t}} \\ 0 & 1 \end{smallmatrix}\right]$

3.4 Face Scale Estimation

3.4.1 Front Camera Pose Estimation

Based on the estimated relative pose $\mathbf{M}_{r_1}^{t_1 \to t_2}$ of rear camera C_{r_1} from Pose 1 to Pose 2 and the calibrated $\mathbf{M}_{r_1 \to f}$, we can obtain the relative pose $\mathbf{M}_f^{t_1 \to t_2}$ of front camera C_f from Pose 1 to Pose 2. Figure 3 demonstrates the relationships of the camera transformations. We can derive from it $\mathbf{M}_f^{t_1 \to t_2} \mathbf{M}_{r_1 \to f} = \mathbf{M}_{r_1 \to f} \mathbf{M}_{r_1}^{t_1 \to t_2}$. Since $\mathbf{M}_{r_1 \to f}$ is an invertible matrix, the relative pose $\mathbf{M}_f^{t_1 \to t_2}$ can be computed from:

$$\mathbf{M}_f^{t_1 \to t_2} = \mathbf{M}_{r_1 \to f} \mathbf{M}_{r_1}^{t_1 \to t_2} (\mathbf{M}_{r_1 \to f})^{-1}. \tag{2}$$

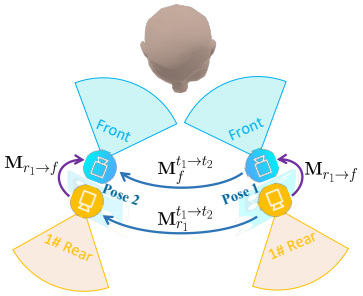
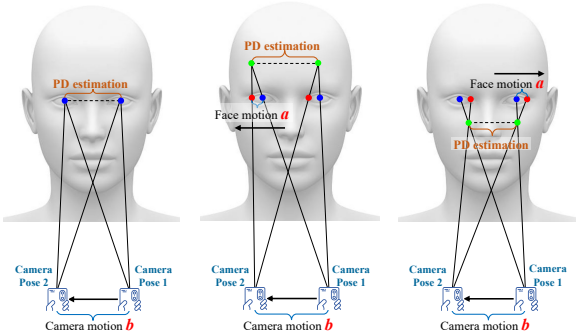
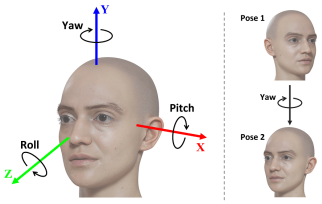


Figure 3: Transformation of rear-front cameras

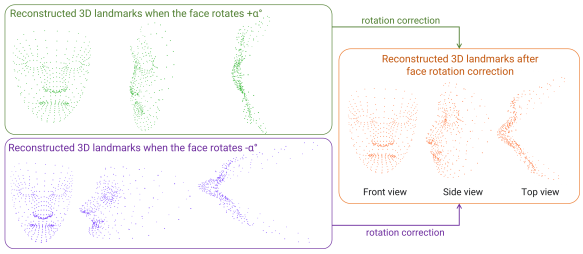


(a) No face motion (b) Same direction (c) Opposite direction

Figure 4: Impact analysis of face motion



(a) Face rotation directions



(b) Face motion correction

Figure 5: Face rotations and correction. (a) Pitch, roll, and yaw movements of a human head. To rectify scale estimation, we assume the head motion is limited to yaw rotation during the swing. (b) If there is a yaw rotation, the reconstructed 3D landmarks would be greatly deformed. The proposed correction algorithm estimates the rotation to rectify the reconstructed 3D face according to a face prior.

3.4.2 Face Landmark Detection

Assuming that the two selfie images $\{\mathbf{I}_f^{t_1}, \mathbf{I}_f^{t_2}\}$ contain two different views of the same face, we first detect the 2D face landmarks (in pixel units) on the two images. In this study, we adopt Google's MediaPipe Face Mesh solution [23] to detect $m = 468$ face landmarks on each image. The solution employs a lightweight model [19] that can run in real-time on mobile devices. Each landmark is matched with a corresponding landmark on the other image, resulting in m pairs of landmarks $\{\mathbf{p}_i^{t_1,f} \leftrightarrow \mathbf{p}_i^{t_2,f} | i = 1, 2, \ldots, m\}$.

3.4.3 3D Face Landmarks Reconstruction

Similar to Section 3.3.2, we first convert the 2D landmarks to normalized coordinates using $\mathbf{K}_f$ and $\mathbf{k}_f$ and obtain m pairs of landmarks in world coordinates $\{\tilde{\mathbf{p}}_i^{t_1,f} \leftrightarrow \tilde{\mathbf{p}}_i^{t_2,f} | i = 1, 2, \ldots, m\}$. Then, we adopt DLT algorithm [16] to triangulate the 3D face landmarks $\{\mathbf{P}_i^f | i = 1, 2, \ldots, m\}$ from 2D landmark pairs using camera transformation $\mathbf{M}_f^{t_1 \to t_2}$. Based on the 3D landmark reconstruction, we can obtain not only the real-scale 3D face mesh but also the distance and angle from the camera to the face. This feature can be employed by various applications, as exemplified in Figure 1.

3.4.4 Impact Analysis of Face Motion

In 3D face triangulation, there is an underlying assumption that the face remains still when the person swings the phone. If the face moves, the reconstructed 3D face mesh will not only be the wrong size, but may also be deformed. In this section, we will analyze the impact of face motion on face scale estimation. To measure the estimation accuracy, we first need to define appropriate evaluation metrics. In online shopping glasses, *pupillary distance* (*i.e.*, the distance between left and right pupils) is crucial for making prescription glasses. Since PD also reflects the accuracy of face scale estimation, we will use it as the evaluation metric in the experiments.

To simplify the analysis, we assume that the smartphone and the face only move along the X-axis direction. As shown in Figure 4, the pupils are marked with blue dots when the camera is in Pose 1. After the camera is moved for a distance b to Pose 2, the positions of the pupils are marked with red dots. If there is no face motion (Figure 4(a)), the pupils estimated from triangulation are true positions without errors. If the face moves in the same direction as the phone for a distance of a (Figure 4(b)), the pupils estimated from triangulation are marked with green dots. By using properties of similar triangles, we can derive the PD estimation is $\frac{a}{b-a} \times 100\%$ larger than the true value. Similarly, if the face moves in the opposite direction to the phone for a distance of a (Figure 4(c)), the PD estimation is $\frac{a}{b+a} \times 100\%$ smaller than the true value. For example, if a person's PD is 65 mm, and the smartphone moves for 100 mm. If the face moves 10 mm in the same direction, the error of PD estimation is $+\frac{10}{100-10} \times 65 \approx +7.22$ mm. If the face move 10 mm in the opposite direction, the error of is $-\frac{10}{100+10} \times 65 \approx -5.91$ mm. It can be seen that face motion has a non-negligible impact on PD estimation.

3.4.5 Algorithmic Correction for Face Motion

The user's head can undergo arbitrary 3D translation or rotation during the swing. There are three directions of human head rotation: pitch, roll, and yaw, as shown in Figure 5(a). We notice in the study that most of the head motion is negligible except the yaw. Therefore, we design a simple correction algorithm that models the unwanted head motion as a one degree-of-freedom yaw motion.

Our correction method is based on the observation that if the head rotates in the yaw direction during hand swing, the 3D face mesh reconstructed from the 2D landmarks of the two images $\{\mathbf{I}_f^{t_1}, \mathbf{I}_f^{t_2}\}$ will be severely deformed. Specifically, the 3D face will become very flat or sharp in the Z direction, which is different from the normal face geometry. Our idea is to rotate the landmarks $\{\mathbf{p}_i^{t_2,f} | i = 1, 2, \ldots, m\}$ on $\mathbf{I}_f^{t_2}$ by an angle so that the 3D face mesh reconstructed with $\{\mathbf{p}_i^{t_1,f} | i = 1, 2, \ldots, m\}$ is geometrically as close to the normal face as possible. In terms of implementation, we first define a 3D canonical face, and then estimate the 3D face landmarks $\{\mathbf{P}_i^{t_2,f} | i = 1, 2, \ldots, m\}$ from single image $\mathbf{I}_f^{t_2}$ and the transformation $\mathbf{M}_{3D \to 2D}^{t_2,f}$ from 3D to

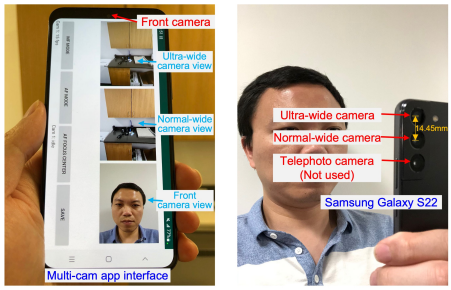


Figure 6: Multi-camera app for data collection

2D landmarks[1]. The 3D face landmarks are approximated from the 3D canonical face and do not reflect the real size of the face. Then, we can rotate the 3D face landmarks by an angle α in the yaw direction and project them to $\mathbf{I}_f^{t_2}$ using $\mathbf{M}_{3D\to 2D}^{t_2,f}$. The new 2D landmarks $\{\mathbf{p}_i^{t_2,f}(\alpha)|i=1,2,\ldots,m\}$ on $\mathbf{I}_f^{t_2}$ can be used to triangulate new 3D face landmarks $\{\mathbf{P}_i^f(\alpha)|i=1,2,\ldots,m\}$ with the landmarks $\{\mathbf{p}_i^{t_1,f}|i=1,2,\ldots,m\}$ on $\mathbf{I}_f^{t_1}$. We will obtain different triangulated 3D face candidates by changing the value of α.

Among these 3D face candidates, we choose the one that is closest to the real face geometry as the final result. In particular, scaled point cloud registration [8] is performed between the triangulated 3D face candidate $\{\mathbf{P}_i^f(\alpha)|i=1,2,\ldots,m\}$ and the 3D face landmarks $\{\mathbf{P}_i^{t_2,f}|i=1,2,\ldots,m\}$. The angle α^* that produces the smallest registration error (*i.e.*, root-mean-squared pairwise distance) is the estimated head rotation. Its corresponding 3D face candidate $\{\mathbf{P}_i^f(\alpha^*)|i=1,2,\ldots,m\}$ are the corrected landmarks. Figure 5(b) shows an example of the reconstructed 3D face landmarks before and after applying face motion correction.

4 Implementation

4.1 Prototype

We developed an Android app (namely "*Multi-camera* app") to collect data. As shown in Figure 6, *Multi-camera* app captures three images simultaneously when clicking the "SAVE" button on the touchscreen. We use the manual focus (MF) mode to fix the focal length of the cameras and disable optical stabilization in this app. We tested the app on Samsung Galaxy S22 and Google Pixel 5 with Android 12 system for data collection.

As shown in Figure 6, Galaxy S22 has a single front camera and triple rear cameras (*i.e.*, ultra-wide, normal-wide, and telephoto). In our experiments, we only employ two of the three rear cameras: ultra-wide as C_{r_1} and normal-wide as C_{r_2}. After calibration, the baseline of the dual cameras is 14.45 mm. The output resolutions of the three images from the *Multi-camera* app are 1920×1440 pixels.

In our prototype system, we implemented the "Swing-for-True-Scale" algorithm using Python on a laptop and leave the mobile version as future work. The computer vision modules (including camera calibration, feature detection, keypoint matching, 3D triangulation, PnP pose estimation) are developed based on `OpenCV` library. The matrix operations are realized with `NumPy` library. The face landmark detection module is implemented with the MediaPipe Python package. For PD estimation, we do not directly adopt the detected pupil but estimate the center of the eye as the pupil position. In this way, even if the pupils greatly move during the swing, it will not affect the PD estimation.

We transfer the captured 6 images (3 images in each pose) from the smartphone to an Apple Macbook Pro 2017 with a 2.8 GHz quad-core Intel i7 CPU and 16GB memory and run the Python program. The estimation can be completed within 1 second on average. It is worth noting that all of the `OpenCV` functions and MediaPipe solution in the current implementation have the corresponding Android versions, and thus can be ported to smartphones in future work.

[1]The implementation of this part is inspired by open source `https://github.com/Rassibassi/mediapipeDemos`.

(a) selfie image (b) small size (c) medium size (d) large size

Figure 7: Virtual try-on sunglasses with true scale

(a) selfie image (b) scene image (c) reflection (d) cutout effect

Figure 8: Virtual try-on sunglasses involving back-camera scene

4.2 Enhanced Virtual Try-On Effect

When shopping for glasses online, users can choose the color and style of glasses with AR-based virtual try-on apps. While traditional virtual try-on apps only utilize a single selfie camera, we show some examples of enhancing virtual try-on via Dual Camera mode.

4.2.1 True-Scale Effect

Since traditional virtual try-on only uses a single selfie camera, it does not know the true size of the face. When displaying AR effects, existing virtual try-on apps usually scale the glasses to fit the detected face. In contrast, the proposed "Swing-for-True-Scale" method can estimate the true scale of the face through multi-camera. Benefiting from it, we can display the real virtual try-on effect of a spectacle model of known size. As shown in Figure 7, we adopt the Sunglasses template of Snap AR Lens Studio [35] to simulate the true-scale virtual try-on effect of different sizes (Small, Medium, and Large) of sunglasses on the same face. This additional effect can help people choose sunglasses of the right size to purchase, which greatly improves online shopping experience.

4.2.2 Real Reflection Effect

A major advantage of multi-camera is that it can simultaneously capture the front selfie image (*e.g.*, Figure 8(a)) and back scene image (*e.g.*, Figure 8(b)). Thanks to this feature, during the online virtual try-on, the back scene can be rendered on the virtual sunglasses to simulate the real reflection effect, which makes the AR effects more realistic and engaging. As shown in Figure 8(c), we simulate this effect using the Sunglasses template of Snap AR Lens Studio.

4.2.3 Other Creative Effects

With multi-camera, it is also possible to combine virtual try-on with the creative effects in Dual Camera mode. For example, we can integrate the "cutout" effect in Snapchat into virtual try-on to replace the background of the selfie with the content of the back camera, resulting in a special virtual try-on experience. As shown in Figure 8(d), we simulate this effect using the Segmentation and Sunglasses templates of Snap AR Lens Studio.

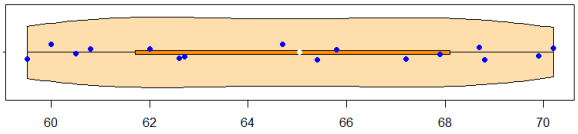


Figure 9: Distribution of the participants' PD (in millimeter) using violin plot and 1D scatter plot. Each data point is represented by a blue dot, and the white dot denotes the medium.

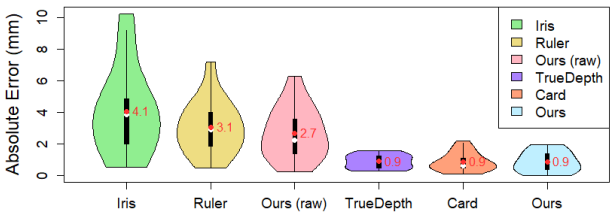


Figure 10: Violin plots for comparing PD estimation errors (in millimeter). The methods are sorted by the mean error (red dot) in descending order. "Ours (raw)" represents our "Swing-for-True-Scale" method without face motion correction.

5 Evaluation

5.1 Participants and Apparatus

We recruited 16 voluntary participants (12 males, 4 females). The participants' ages ranged from 22 to 38 (mean = 26.7). We used a digital caliper to obtain the ground truth PD value of each participant by averaging multiple independent measurements carefully. Figure 9 shows the PD distribution. The shape of the violin plot and the scatter plot show that the 16 participants are relatively evenly distributed in the interval of 59.5∼70.2 *mm* with the median of 65.05 *mm*.[2] The required images were captured by the participants with a Galaxy S22 and transferred to an Apple Macbook Pro 2017 for processing.

5.2 Research Method and Procedure

We evaluate the proposed method in terms of feasibility and applicability. The feasibility is demonstrated by evaluating the scale estimation accuracy. As previously discussed, we adopt PD as the evaluation metric. The participants were instructed to measure their PD using the following methods in the experiments:

- **Ruler.** The participants follow the PD measurement guide [43] to use a ruler to measure the PD in front of a mirror.
- **Card.** The participants place a standard-size card (*e.g.*, credit card with the size of $85.6mm \times 53.98mm$) on the forehead as a reference object. Then, the card edge and pupil positions (in pixel units) are detected on the selfie images with computer vision algorithms. Based on the ratio of PD to the card length, the actual PD can be estimated. In the experiments, the participants directly use an off-the-shelf PD measurement app[3] based on the standard-size card.
- **TrueDepth.** The iPhones with TrueDepth sensor can obtain the depth of the face in 3D space and thus the true scale of the face. Although this study focuses on the vast majority of smartphones that are not equipped with front depth sensors, we adopt TrueDepth as a reference for comparing accuracy. The participants measure the PD using an off-the-shelf app[4] based on TrueDepth.
- **Iris.** This approach is based on the fact that the iris diameter of human eyes is roughly constant, falling in $11.7 \pm 0.5mm$ across a wide range of population [4, 33]. With the participants' selfie images, we can estimate the PD by detecting the iris and pupil landmarks in pixels and computing the ratio between PD and iris diameter. We implemented the iris landmark estimation using Google's MediaPipe Iris solution [23].
- **Ours (raw).** The proposed method *without* face motion correction described in Section 3.4.5.
- **Ours.** The proposed method including face motion correction.

Each participant used all six methods to measure the PD. We evaluate them in various indoor scenes for different participants. The first three (*i.e.*, ruler, card, and TrueDepth) are off-the-shelf solutions for measuring PD, and the participants can see the results instantly. The other three methods are developed by us and run on a laptop with Python. The participants were asked to capture images in two different poses with the *Multi-camera* app. Then, the images are transferred to the laptop to compute the estimation results.

On the other hand, the applicability is demonstrated through the questionnaires completed by the participants after experiencing all the PD measurement methods. Since the proposed method is mainly designed for smartphones without front depth sensors, we do not consider the TrueDepth method in the applicability analysis. The iris-based method is excluded because it has low accuracy from the feasibility analysis (see Section 5.3.1). Two other existing methods (*i.e.*, ruler and card) are compared with our method. In the questionnaire, the participants were first asked to rate the ease of use and the accessibility to the apparatus of the three methods (ruler, card, ours) in 5-point Likert scale scores. Then, we asked the participants to order the three methods based on the question "*If the measurement accuracies are the same, which method will you prefer to use?*" and provide comments on the proposed method. We also asked them to answer the question "*Will you use the multi-camera method to measure PD?*" and explain the reason (note the multi-camera method is our method). The study lasts about 15 minutes for each participant.

5.3 Results

5.3.1 Feasibility Analysis

We compute the absolute error of the PD estimation of each method, by comparing it to the ground truth, for every participant, as shown in Figure 10. The violin plot demonstrates the distribution of the errors from all participants. The red and white dots represent the mean and medium values, respectively. Among the six methods, the iris-based method has the largest error with an average of 4.1 *mm*. The errors of the iris-based method mainly come from two aspects: (1) the iris diameter may deviate from 11.7 *mm* due to individual differences; (2) the error from iris landmark detection. The errors of the ruler measurement are also not small, with an average of 3.1 *mm*. The participants report that they felt it difficult to align the pupil to the ruler's lines in the mirror.

A one-way repeated-measures ANOVA reveals significant differences in the absolute error of the six measurement methods ($F_{5,90} = 13$, $p < .0001$). Post-hoc tests with Bonferroni correction show that our method ($M = 0.88mm$, $SD = 0.65mm$) yields significantly smaller absolute errors than the iris-based method ($M = 4.07mm$, $SD = 2.75mm$; $p < .0001$), the ruler measurement ($M = 3.06mm$, $SD = 1.69mm$; $p < .0017$), and our method without face motion correction ($M = 2.68mm$, $SD = 1.63mm$; $p < .016$), which verified the effectiveness of the face motion correction algorithm. Meanwhile, the post-hoc tests indicate the absolute error of our method ($M = 0.88mm$, $SD = 0.65mm$) is at the same level as that of the TrueDepth method ($M = 0.89mm$, $SD = 0.45mm$; $p = 1.0$) and the card-based method ($M = 0.89mm$, $SD = 0.59mm$; $p = 1.0$).

[2] The normal range of adult PD is 54∼74 *mm*.

[3] GlassesOn (`https://play.google.com/store/apps/details?id=com.sixoversix.copyglass`) with a score of 4.8 out of 5 in 17.5 thousand reviews.

[4] EyeMeasure (`https://apps.apple.com/us/app/eyemeasure/id1417435049`) with a score of 4.8 out of 5 rated by 14.3 thousand users.

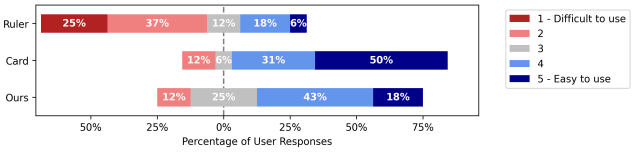


Figure 11: Participants' rating on the ease of use

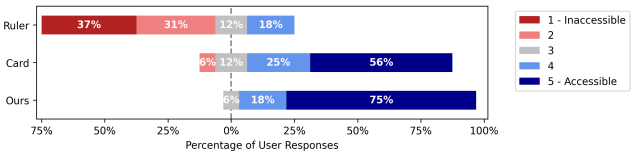


Figure 12: Participants' rating on the accessibility to the apparatus

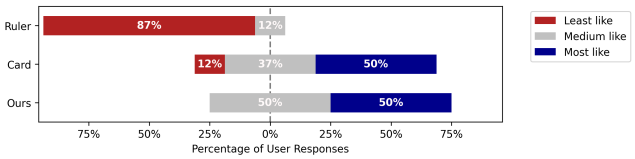
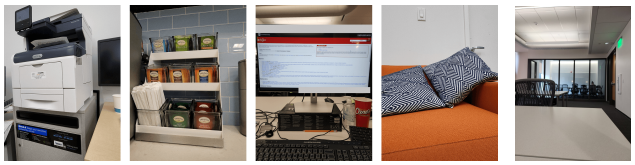


Figure 13: Participants' preference to use the methods



(a) Scenes tested with accurate estimations (b) failure case

Figure 14: Example scenes tested in the experiments

5.3.2 Applicability Analysis

The results from the questionnaire are reported in this section. Figure 11 summarizes the participants' rating on the ease of use of the three methods (ruler, card, ours). About 62% of the participants rated score 1 or 2 for the ruler-based method, indicating it was difficult to use. In contrast, there are only 12% of the participants feeling the other two methods difficult to use (*i.e.*, a score less than 3). Compared with our method, more participants gave score 5 (easy to use) to the card-based method. Some comments on our method in the questionnaires indicate the reason, *e.g.*, "*keeping head stable is not easy*" and "*need detailed instructions*".

Figure 12 demonstrates the participants' rating on the accessibility to the apparatus of the three methods. The apparatus in the ruler-based method consists of a ruler and a mirror. About 68% of the participants rated score 1 or 2 to this method, implying the ruler or mirror is inaccessible to them. The apparatus in the card-based method include a smartphone with front camera and a standard-size card. Only 6% of the participants gave a score less than 3 (*i.e.*, inaccessible) to it, indicating that most of them felt it easy to find a card (*e.g.*, credit card, student card) around. Our method only needs a smartphone with multiple cameras, which can be satisfied by almost all the latest models. The participants also agreed with this fact that about 93% of them rated a score more than 3 (*i.e.*, accessible) to our method.

The participants' preference for the three methods is summarized in Figure 13. The ruler-based method is the least liked, which is in line with the results of the participants' ratings on the ease of use (Figure 11) and the accessibility to the apparatus (Figure 12). Two participants ranked the card-based method the least like because "*I do not like the card to attach to my face because it might be dirty*". It is interesting to see from Figure 13 that half of the participants' favorite method is the card-based method, while the other half voted for our method. The result is consistent with that in Figure 11 and Figure 12. The participants' comments on our method also explained their preference, *e.g.*, "*quick and convenient*", "*fewest items required*", and "*comparably easy to use*".

For the responses to the question "*Will you use the multi-camera method to measure PD?*", about 87.5% of the participants would be willing to use our multi-camera method, while the other 12.5% were not sure. No one answered "*No*". According to the collected reasons, the participants would use our method because "*it is accurate*" and "*easy to use anywhere*"; the main concern of the two participants who answered "*Not sure*" is the head movement issue – they thought our method is "*not easy to keep head static*" and "*keeping head static can be a challenge*". But as shown in Figure 10, our algorithm for face motion correction can effectively alleviate this issue.

6 Limitations

6.1 Dependence on Background Scenes

The accurate scale estimation of the proposed method relies on reliable camera pose estimation. The pose estimation is based on feature detection, keypoint matching, and 2D-3D correspondence. Thus, the effectiveness of our method is affected by the quality of back scenes. As described in Section 3.3, we adopt some good practices (*e.g.*, Lowe's ratio test for keypoint matching, feature binning for pose estimation) to make our method robust to various background scenes. To verify the robustness, we assigned the participants to different scenes that are common in life. Figure 14(a) shows four examples of scene images capture by the participants, including printer, tea bags, desktop, and sofa. Our method achieved accurate estimation results in these scenes. However, in some challenging scenarios, our method failed. Figure 14(b) shows a failure case where the scene is far away from the camera and with few textures. It could also present difficulties when used in outdoor settings where the background scene undergoes frequent changes.

6.2 Vulnerability to Large Face Motion

A basic assumption of the proposed method is the face should be as static as possible when swinging the phone. In practical applications, it is impossible for the users to keep the head completely still. To address this issue, we proposed a correction algorithm for rectifying the head rotation around the yaw axis. The effectiveness of the algorithm was verified in our user study. As shown in Fig 10, the mean error of PD estimation was decreased from 4.1 mm to 0.9 mm with the face motion correction. However, the proposed method will suffer from large estimation errors if there is a big face motion. In this case, the motion correction algorithm is unable to predict the non-negligible face translation.

6.3 Limitations in Implementations

Our prototype system employs an Android app to collect data, which is then transmitted to a laptop for algorithm calculation. The current implementation demonstrates the feasibility of robustly estimating metric face scale from images captured by a smartphone. However, we have yet to examine the real-time performance of the proposed algorithm on smartphones. Since the proposed approach is computationally lightweight, we believe a real-time on-device implementation is possible with careful development and optimization, which we leave for future work.

7 Discussions

7.1 Interpretation of Results

The human PD has a reasonable range, such as the 45∼82 *mm* measuring range of a pupilometer [2], which very few people exceed.

Therefore, any PD estimation beyond this range will be considered a failure, and the measurement will need to be retested. Only the data measured within this range will be included in the results of Section 5.3.1. Combining the factors of scene quality and face motion, the overall measurement success rate is approximately 80%.

7.2 Out-of-the-Box VIO/SLAM Systems

In this study, we did not choose the out-of-the-box VIO/SLAM systems (*e.g.*, Apple ARKit, Google ARCore) as baseline for the following reasons. 1) It is well-known that monocular VIO/SLAM systems need a specific initialization step to get initial depth estimates to bootstrap pose estimation [29], which can take up to a few seconds. 2) VIO/SLAM systems focus on real-time estimation of a continuous camera trajectory, which is not necessary for our purpose and often trades accuracy for computing efficiency. We propose a quick, responsive way of getting pose estimates for sparse views. 3) Evaluating the contributions of sensors (IMU and cameras) in pose estimation is challenging due to the closed-source nature of ARKit and ARCore. Our paper focuses on developing an open-source camera-based solution.

7.3 Privacy Concerns

Compared with normal selfie mode, our method is built upon Dual Camera mode which may increase the risk of the leakage of private information. With the front camera on, the users tend to focus on the content of the selfie and ignore the objects in the back scene. Sensitive information that appears in the rear camera (*e.g.*, the screen content on the third image in Figure 14(a)) may be leaked when using Dual Camera mode. In order to reduce this risk, we can use computer vision methods [37] to automatically detect private information on the image and hide them. In comparison, the card-based method has even greater privacy concerns. Since the standard-size cards accessible to most users are usually credit cards or debit cards, it is easy to reveal sensitive information on the card (*e.g.*, names and numbers) when measuring PD using the front camera.

7.4 Other Applications

True-Scale 3D Face Reconstruction. A straightforward extension of our work is to reconstruct the 3D face mesh with the metric scale, which is crucial for applications in AR/VR. Most single-view 3D face reconstruction [10, 11, 13, 32, 46] relies on learned statistical prior of the face model. Multi-view-based methods [6, 28] are able to explore the geometry constraint and show better performance. However, almost all those methods are not able to reconstruct the actual face dimensions. In our system, we have selfie images and their corresponding 6-DOF camera poses. It allows us to recover the 3D point cloud of the face landmarks. Combining with the selfie images, we can get the 3D face mesh with the scale.

Eye Gaze Tracking. The proposed method can also be used for eye gaze tracking on mobile devices [17, 41]. For image-based gaze tracking, it has been demonstrated that an extra depth camera improves tracking accuracy as they can directly measure head pose and eye position in 3D [21]. So far, few works have explored gaze tracking using depth information on mobile devices. The proposed method can help geometric-based methods [18, 34, 40, 45] by providing true-scale depth estimates at detected face landmarks, or by feeding a reconstructed full-depth map as input to appearance-based methods. This enables accurate gaze tracking on devices even without a front-facing depth camera, which greatly broadens the potential application of mobile gaze tracking.

Face Anti-Spoofing. The mobile apps using face authentication usually require the user to open the front camera and capture a selfie or record a video duration for anti-spoofing. But these methods could be easily attacked by impersonation of 2D methods, like using a printed face or playing a video on a display. We come up with a new interaction method for anti-spoofing; the app asks the user to keep their head still and swing their phone while recording a short video to capture the front, left, and right sides of the face. From the video, the true-scale 3D face can be reconstructed, which can be compared to the same person in the database so that 2D face spoofing attacks can be easily detected. 3D attacking methods [42] like using a paper mask or resin mask lack the true scale and expose the boundary between the mask and the skin of the attacker in the left/right side views. Another 3D attacking method using a mannequin lacks natural eye and talking movement. Thus, our method could severely increase the face spoofing's cost.

8 Conclusion

In this work, we propose a novel approach to the true-scale recovery of users' faces for smartphone apps. The proposed approach operates in Dual Camera mode, which involves streaming images with both the front and rear cameras simultaneously. The approach only requires minimal interaction from the user: the user only needs to swing the phone in front of their face. We conduct a user study to evaluate the feasibility and applicability of the approach as compared to several existing methods. According to the questionnaires collected from the participants, it is also easy to use and accessible. These merits make the proposed approach a competitive alternative for face scale estimation on mobile phones, especially considering that most of smartphone models do not have a front depth camera.

As our first attempt to face scale estimation by swinging the phone, this work shows promising results of the proposed approach. Nevertheless, it does have a few limitations including less robustness to textureless or dynamic scenes, and difficulty in correcting large face motion, which we believe can be resolved in future work. In addition to showing how the approach can improve user experiences in true-size virtual try-on for online shopping, we also give preliminary roadmaps of how the approach can assist a variety of other applications, including true-scale 3D face reconstruction, eye faze tracking, and face anti-spoofing. We hope this work will spur further studies in the Dual Camera mode and we are excited to see more applications enabled by this Swing-for-True-Scale approach.

Acknowledgments

The authors would like to thank Vu An Tran for supporting Android app, and Rajan Vaish, Riku Arakawa, Fannie Liu, Zekun Cai, and Jingyi Xie, for the helpful discussions.

References

[1] A. Abdelhamed, A. Punnappurath, and M. S. Brown. Leveraging the availability of two cameras for illuminant estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6637–6646, 2021.

[2] Amcon. *Amcon Digital Pupilometer User's Manual EQ-6310*, 2023.

[3] BeReal SAS. BeReal, 2023. Accessed May 1, 2023. `https://bere.al/en`.

[4] J. P. Bergmanson and J. G. Martinez. Size does matter: what is the corneo-limbal diameter? *Clinical and Experimental Optometry*, 100(5):522–528, 2017.

[5] J.-Y. Bouguet. Camera calibration toolbox for matlab. *http://robots.stanford.edu/cs223b04/JeanYvesCalib*, 2004.

[6] C. Cao, T. Simon, J. K. Kim, G. Schwartz, M. Zollhoefer, S.-S. Saito, S. Lombardi, S.-E. Wei, D. Belko, S.-I. Yu, et al. Authentic volumetric avatars from a phone scan. *ACM Transactions on Graphics (TOG)*, 41(4):1–19, 2022.

[7] D. Cheng, J. Shi, Y. Chen, X. Deng, and X. Zhang. Learning scene illumination by pairwise photos from rear and front mobile cameras. In *Computer Graphics Forum*, vol. 37, pp. 213–221, 2018.

[8] R. Diamond. A note on the rotational superposition problem. *Acta Crystallographica Section A: Foundations of Crystallography*, 44(2):211–216, 1988.

[9] C. B. Duane. Close-range camera calibration. *Photogramm. Eng*, 37(8):855–866, 1971.

[10] G. Fanelli, M. Dantone, J. Gall, A. Fossati, and L. Van Gool. Random forests for real time 3d face analysis. *International Journal of Computer Vision (IJCV)*, 101(3):437–458, 2013.

[11] Y. Feng, F. Wu, X. Shao, Y. Wang, and X. Zhou. Joint 3d face reconstruction and dense alignment with position map regression network. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 534–551, 2018.

[12] GSMArena.com. Gsmarena phone finder, 2023. Accessed May 1, 2023. `https://www.gsmarena.com/results.php3?nYearMin=2022`.

[13] J. Guo, X. Zhu, Y. Yang, F. Yang, Z. Lei, and S. Z. Li. Towards fast, accurate and stable 3d dense face alignment. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 152–168. Springer, 2020.

[14] C. Ham, S. Lucey, and S. Singh. Hand waving away scale. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 279–293. Springer, 2014.

[15] C. Ham, S. Lucey, and S. Singh. Absolute scale estimation of 3d monocular vision on smart devices. In *Mobile Cloud Visual Media Computing*, pp. 329–353. Springer, 2015.

[16] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[17] Q. Huang. *TabletGaze: Dataset and Algorithm for Unconstrained Appearance-based Gaze Estimation in Mobile Tablets*. PhD thesis, Rice University, 2015.

[18] L. Jianfeng and L. Shigang. Eye-model-based gaze estimation by rgb-d camera. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 592–596, 2014.

[19] Y. Kartynnik, A. Ablavatski, I. Grishchenko, and M. Grundmann. Real-time facial surface geometry from monocular video on mobile gpus. *arXiv:1907.06724*, 2019.

[20] W.-S. Lai, Y. Shih, L.-C. Chu, X. Wu, S.-F. Tsai, M. Krainin, D. Sun, and C.-K. Liang. Face deblurring using dual camera fusion on mobile phones. *ACM Transactions on Graphics (TOG)*, 41(4):1–16, 2022.

[21] D. Lian, Z. Zhang, W. Luo, L. Hu, M. Wu, Z. Li, J. Yu, and S. Gao. Rgbd based gaze estimation via multi-task cnn. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, vol. 33, pp. 2488–2495, 2019.

[22] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision (IJCV)*, 60(2):91–110, 2004.

[23] MediaPipe Solutions. Face landmark detection guide, 2023. Accessed May 1, 2023. `https://developers.google.com/mediapipe/solutions/vision/face_landmarker`.

[24] Meta Platforms Inc. Instagram, 2023. Accessed May 1, 2023. `https://www.instagram.com`.

[25] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. *VISAPP (1)*, 2(331-340):2, 2009.

[26] J. Mustaniemi, J. Kannala, S. Särkkä, J. Matas, and J. Heikkilä. Inertial-based scale estimation for structure from motion on mobile devices. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4394–4401. IEEE, 2017.

[27] Y. Nam, Y. Kong, B. Reyes, N. Reljin, and K. H. Chon. Monitoring of heart and breathing rates using dual cameras on a smartphone. *PloS one*, 11(3):e0151013, 2016.

[28] K. Park, U. Sinha, J. T. Barron, S. Bouaziz, D. B. Goldman, S. M. Seitz, and R. Martin-Brualla. Nerfies: Deformable neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5865–5874, 2021.

[29] J.-C. Piao and S.-D. Kim. Adaptive monocular visual–inertial slam for real-time augmented reality applications in mobile devices. *Sensors*, 17(11):2567, 2017.

[30] F. Rameau, J. Park, O. Bailo, and I. S. Kweon. Mc-calib: A generic and robust calibration toolbox for multi-camera systems. *Computer Vision and Image Understanding*, 217:103353, 2022.

[31] Cvišić, Igor and Marković, Ivan and Petrović, Ivan. SOFT2: Stereo Visual Odometry for Road Vehicles Based on a Point-to-Epipolar-Line Metric. *IEEE Transactions on Robotics*, 2022.

[32] S. Romdhani and T. Vetter. Estimating 3d shape and texture using pixel intensity, edges, specular highlights, texture constraints and a prior. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, pp. 986–993. IEEE, 2005.

[33] F. Rüfer, A. Schröder, and C. Erb. White-to-white corneal diameter: normal values in healthy humans obtained with the orbscan ii topography system. *Cornea*, 24(3):259–261, 2005.

[34] S.-W. Shih and J. Liu. A novel approach to 3-d gaze tracking using stereo cameras. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 34(1):234–245, 2004.

[35] Snap Inc. Snap AR Lens Studio, 2023. Accessed May 1, 2023. `https://ar.snap.com/en-US/lens-studio`.

[36] Snap Inc. Snapchat, 2023. Accessed May 1, 2023. `https://www.snapchat.com`.

[37] P. Speciale, J. L. Schonberger, S. N. Sinha, and M. Pollefeys. Privacy preserving image queries for camera localization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 1486–1496, 2019.

[38] G. Terzakis and M. Lourakis. A consistently fast and globally optimal solution to the perspective-n-point problem. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 478–494. Springer, 2020.

[39] R. Y. Tsai, R. K. Lenz, et al. A new technique for fully autonomous and efficient 3d robotics hand/eye calibration. *IEEE Transactions on Robotics and Automation*, 5(3):345–358, 1989.

[40] K. Wang and Q. Ji. Real time eye gaze tracking with kinect. In *Proceedings of the International Conference on Pattern Recognition (ICPR)*, pp. 2752–2757. IEEE, 2016.

[41] E. Wood and A. Bulling. Eyetab: Model-based gaze estimation on unmodified tablet computers. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, pp. 207–210, 2014.

[42] Z. Yu, Y. Qin, X. Li, C. Zhao, Z. Lei, and G. Zhao. Deep learning for face anti-spoofing: A survey. *arXiv preprint arXiv:2106.14948*, 2021.

[43] Zenni Optical Inc. How to measure your pupillary distance, 2023. Accessed May 1, 2023. `https://www.zennioptical.com/measuring-pd-infographic`.

[44] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 22(11):1330–1334, 2000.

[45] X. Zhou, H. Cai, Y. Li, and H. Liu. Two-eye model-based gaze estimation from a kinect sensor. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1646–1653. IEEE, 2017.

[46] X. Zhu, X. Liu, Z. Lei, and S. Z. Li. Face alignment in full pose range: A 3d total solution. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 41(1):78–92, 2017.