3D Curve Creation on and around Physical Objects with Mobile AR

Hui Ye, Kin Chung Kwan, and Hongbo Fu*

Abstract—The recent advance in motion tracking (e.g., Visual Inertial Odometry) allows the use of a mobile phone as a 3D pen, thus significantly benefiting various mobile Augmented Reality (AR) applications based on 3D curve creation. However, when creating 3D curves on and around physical objects with mobile AR, tracking might be less robust or even lost due to camera occlusion or textureless scenes. This motivates us to study how to achieve natural interaction with minimum tracking errors during close interaction between a mobile phone and physical objects. To this end, we contribute an elicitation study on input point and phone grip, and a quantitative study on tracking errors. Based on the results, we present a system for direct 3D drawing with an AR-enabled mobile phone as a 3D pen, and interactive correction of 3D curves with tracking errors in mobile AR. We demonstrate the usefulness and effectiveness of our system for two applications: in-situ 3D drawing, and direct 3D measurement.

Index Terms—Mobile Augmented Reality; In-situ 3D Drawing; Interactive System; Curve Refinement; Optimization

1 Introduction

3D virtual curve creation is a fundamental operation in many graphics applications, including 3D sketching [1], [2], [3], 3D modeling [4], [5], 3D animation [6], [7], [8], and 3D measurement [9], [10], [11]. Many existing techniques such as single-view sketching [12] and multi-view sketching [13] focus on 3D curve creation from 2D inputs. However, requiring users to concentrate on 2D displays, such techniques only support indirect authoring and are often not intuitive to create 3D curves with complex shape. While they might be adapted for drawing *on* physical objects, it is difficult to use them for creating 3D curves *around* objects, since 3D interpretation of 2D input strokes is a difficult problem on its own.

In contrast, mid-air 3D drawing enabled by for example outside-in motion capture (MoCap) systems (e.g., Motion Analysis [14], OptiTrack [15]) allows easy creation of complex 3D curves. However, such systems are not only expensive, thus being limited to a small group of professional users, but also available only in the laboratory setup.

In recent years, inside-out motion tracking algorithms such as Visual Inertial Odometry (VIO) [16] and Concurrent Odometry and Mapping (COM) [17] have been robustly implemented (e.g., in ARKit by Apple and ARCore by Google) to estimate a mobile device's 3D pose, i.e., 3D position and 3D orientation, for mobile Augmented Reality (AR). The ability of freely creating 3D curves using a single AR-enabled mobile device as a 3D pen has been shown in various mobile apps such as Paint Space AR [18] and Just a Line [19], and a recent research prototype called Mobi3DSketch [20], making 3D content creation more accessible by everyday users. However, mobile devices are different from real pens in shape, weight and size. How to naturally use a mobile device to create 3D curves similar to the experience of using a pen has not been explored before. In addition, for curve creation in mobile AR, real-world objects are often used as physical references, requiring a mobile device to move on and around physical objects (Figure 1 (a)). In such tasks, motion tracking might be less robust or lost due to insufficient visual features caused by partial or complete camera occlusion. A similar problem occurs for interacting near textureless regions.

In this work we address the above issues when novice users use a single AR-enabled mobile phone as a 3D pen. We first conduct a user study on *input point* (i.e., which part of the phone is preferred to be used as the tip of a pen) and *phone grip* when using the phone to create 3D curves on and around physical objects with mobile AR (Figure 2). The results show that there are 7 types of input point, and 6 types of phone grip. We also find that users usually pair input point and phone grip, and there are in total 21 types of pairs, some of which are more preferred to others. We select top three pairs, and for each of them analyze four types of motion tracking errors through a quantitative experiment for controlled in-situ 3D drawing. To the best of our knowledge, there is no existing work in exploring the input point, phone grip, and tracking error of a mobile device for creating 3D curves. Our work is the first one to study these issues, which are significant in curve creation with mobile AR.

Based on the results of the studies, we design and develop an interactive system for in-situ 3D curve creation using a single mobile phone in an eyes-free (i.e., without looking at its screen) manner. The study results also show that motion tracking errors are often difficult to avoid. To correct created 3D curves with tracking errors, we thus design a touch-based interface through which users can interactively provide hints (e.g., 2D sketching, relocating key points and/or edges), which are used together the IMU data to refine the curves through first initial correction and

* *Corresponding author*

- *H. Ye and H. Fu are with the School of Creative Media, City University of Hong Kong, and K.C. Kwan is with the University of Konstanz and the School of Creative Media, City University of Hong Kong.E-mail: huiye4-c@my.cityu.edu.hk, {hongbofu, kinckwan}@cityu.edu.hk.*

.

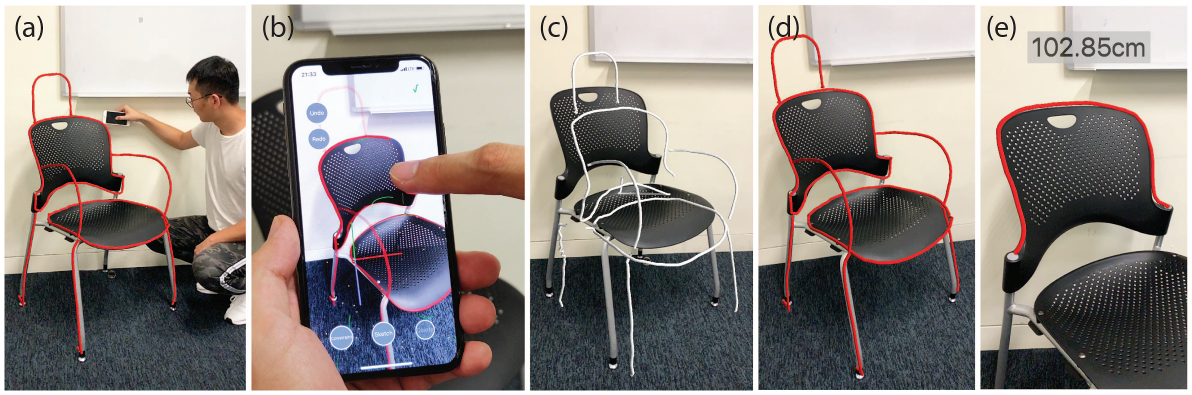


Fig. 1. We present a system for novice users to facilitate 3D curve creation on and around physical objects using an AR-enabled mobile device as a 3D pen (a), followed by interactive correction of curves with tracking errors (b). (c) and (d) show the results after initial creation (using 0.7 mins) and after interactive correction (using 5.6 mins), respectively. We also demonstrate our system with a 3D measurement application (e). The ground truth length is 106.1 cm for this example.

then global curve optimization. Figure 1 shows an example before and after interactive refinement. Compared with the existing relative drawing tools [20], our proposed system is more intuitive and accurate to create curves that have close interaction with real-world objects. Compared with the existing absolute drawing tools [1], [21], [22], which usually rely on specialized hardware setup to obtain accurate results tailored for experts, our proposed system for absolute drawing with interactive refinement is more accessible to ordinary mobile users. Thus, it is a portable, intuitive, and reasonably accurate 3D curve creation tool.

We demonstrate the usefulness and effectiveness of our system for two applications: in-situ 3D drawing, and direct 3D measurement. We conduct a user study where novice users create and refine 3D curves in various scenarios that require close interaction between a mobile phone and real-world environments. Such curves are useful for creating artistic 3D sketches, 3D models, 3D animations or measurements that closely interact with the environment, but are often challenging to create for inside-out motion tracking systems due to the previously mentioned reasons. The user feedback shows our system is intuitive and powerful, and benefits various applications.

Our main contributions are summarized as follows: (1) an elicitation study on the input point and phone grip when using a mobile phone to create 3D curves on and around physical objects, (2) a quantitative experiment on tracking errors of the paired input point and phone grip for controlled in-situ 3D drawing, (3) an interactive system for direct in-situ 3D curve creation using mobile AR, (4) the demonstration of two applications: in-situ 3D drawing and direct 3D measurement, and user studies for novice users to create and refine freeform 3D curves with a single AR-enabled mobile phone as a 3D pen.

2 Related Works

The state-of-the-art motion tracking techniques such as COM [17] and VIO [16], [23], [24], [25] often achieve robust tracking by fusing the visual information and the IMU data. To achieve real-time tracking, such techniques often take the past visual and IMU data in a temporal window for local motion estimation through forward integration, causing accumulated errors, which might be reduced by bundle adjustment [26] and loop closure [27], [28]. Instead, since we post-process a 3D curve returned by an automatic VIO-like tracking algorithm, we perform initial correction using both forward and backward integration, and global curve optimization by considering an entire initially corrected curve and user-specified hints. Given our focus on interactive curve refinement, which can be used together with many existing motion tracking algorithms, a full review on existing motion tracking techniques is beyond the scope of this paper. Interested readers might refer to an insightful survey on visual odometry and visual SLAM [29].

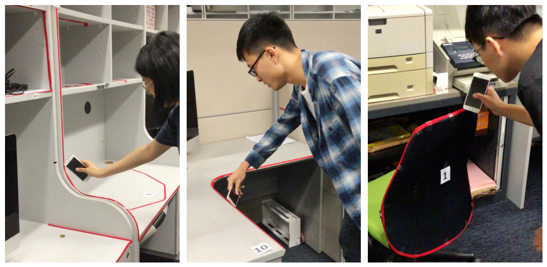
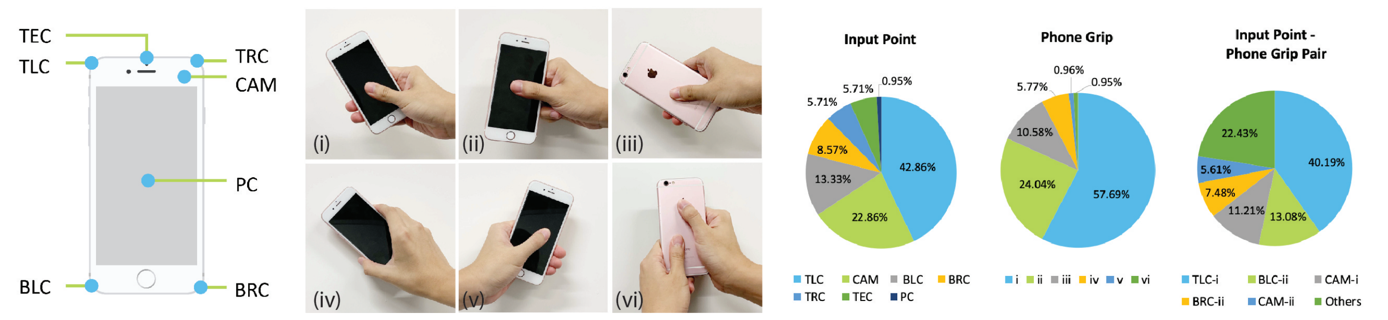


Fig. 2. Elicitation study of input point and phone grip.

3D curve creation has been well studied due to its wide applications. Existing techniques can be roughly categorized into three groups based on their interfaces, namely 2D interfaces [3], [30], [31], [32], [33], [34], [35], [36], [37], 3D interfaces [2], [38], [39], [40], [41], and hybrid 2D-and-3D interfaces [1], [5], [42]. 2D interfaces take as input 2D curves often from graphics tablets or touch screens, and rely on inference algorithms to predict the missing depth. One of the commonly used methods to lift 2D curves to 3D is to project 2D curves to 3D planar surfaces, which can be obtained through environment understanding for mobile AR applications. The planar requirement limits the expressiveness of such techniques, and thus their applications. For example, AR-based measuring apps [9], [10], [11] work only for physical objects with easily detectable planar surfaces. In addition, due to the inherent 2D input nature, 2D interfaces and some of the hybrid 2D-and-3D



(a) Input point layout (b) Phone grip types (c) Percentages of input point, phone grip and pair

Fig. 3. Results of input point and phone grip study.

interfaces [5], [42] do not support direct interaction with physical 3D objects. Compared with these 2D or hybrid 2D-and-3D interfaces for relative curve creation, our direct curve creation interface does not require planar surfaces in the camera view to support the projection of 2D curves, and thus allows users to create freeform 3D curves on and around any real-world objects from multiple views.

Meanwhile, most of the existing techniques have focused on creating 3D curves from scratch, since they do not need to handle tracking errors during in-situ drawing. Instead, we focus on the whole curve creation process, including both curve drawing and curve refinement. Still, some of the techniques based on 2D inputs could be potentially adapted to our problem by using the touch screen of a mobile device as an input device. For example, our multi-view curve refinement feature is inspired by existing multi-view curve creation methods (e.g., [13], [31]). However, in our case we need to deal with unique challenges to respect the IMU data and curve segments with high confidence of tracking.

We are interested more in creating 3D curves directly from 3D inputs. Such techniques (e.g., [2], [21], [22], [43]) have been mainly explored in an immerse virtual reality (VR) environment, where 3D curves are typically created with a tracked stylus and viewed through a head-mounted display (HMD). Due to the adoption of robust outside-in MoCap systems and no need to deal with the interaction between 3D controllers and real-world objects, their resulting 3D curves are often almost free of tracking errors. A similar hardware setup is adopted by SymbiosisSketch [1] but for 3D sketching in HMD-based AR, where users can first create 3D canvases using a stylus with a real object as reference and then draw on the canvases through a 2D interface on a tablet. Since their stylus is tracked by an outside-in MoCap system, its tracking is rather robust even when the stylus is in direct contact with a physical surface. However, due to the dependence on an external MoCap system, SymbiosisSketch has been demonstrated in a laboratory environment only, and is accessible to a small group of users due to high cost. It is a general problem of all the HMD-based systems. In contrast, we would like to extend 3D curve creation to more pervasive usage, which only requires a single mobile device.

The modern mobile AR platforms such as ARKit and ARCore, with robust implementation of VIO-like motion tracking algorithms, enable 3D curve creation in mobile AR. Some related works such as ARPen [41] and Pocket6 [44] based on mobile AR interaction have been proposed recently. Since these works do not consider tracking errors, they restrict interaction to a limited-size control space with sufficient visual features or the space in front of a mobile camera. Different from their systems, our system aims to offer users more freedom for 3D curve creation, especially on and around real-world objects.

As the first powerful 3D sketching system in mobile AR, Mobi3DSketch [20] summarizes the challenges such as the limited field of view (FOV) and the coupling of input and output devices during absolute mid-air drawing, but indirectly addresses them by introducing relative drawing enabled by proxy surfaces as canvases and various snapping mechanisms. In contrast, our work focuses on addressing the fundamental issues (e.g., input point, phone grip, tracking errors) for absolute drawing on and around physical objects. Unlike Mobi3DSketch, which aims for easy creation of well-connected 3D drawings, our work focuses on the creation of individual curves closely interacting with the environment. Our findings can potentially benefit existing works such as SymbiosisSketch and Mobie3DSketch, since it has been shown that absolute drawing and relative drawing have their own unique advantages, and can co-exist in a single system [1], [20].

Researchers have explored to use a mobile phone like a pen for text entry [45], [46], [47]. They use accelerometer or magnetometer data to reconstruct the moving trajectory of the phone. However, these works are only applicable to the creation of 2D strokes or very simple 3D sketching recovery. Compared with these works, our work focuses on a more general and fundamental 3D curve creation task using a single AR-enabled mobile device.

3 Problems of In-situ 3D curve creation

Everybody knows how to naturally hold a pen, whose tip is pre-defined. However, mobile phones are different from pens in terms of the size, shape, weight and other features, and they do not have pre-defined points to be used as pen tips. We thus first conduct an elicitation study to understand how users hold a mobile phone (i.e., phone grip) and define the input point on the phone when they use a mobile device as a 3D pen closely interacting with real environments. In addition, we are also interested in studying motion tracking errors under different choices of input point and phone grip to suggest the most natural way to use a mobile device as a 3D pen with the minimum tracking errors.

3.1 Elicitation Study on Input point and Phone Grip

Setup and Procedure. We considered 3D curves along edges and on surfaces of physical objects, which allow us to add physical references as the same tracing targets for different participants. As the changes of edge or surface might influence user performance, five different conditions were included: *curve along edge*, *on surface*, *from edge to edge*, *from surface to surface*, and *from edge to surface*. We selected ten scenarios involving the above conditions in our office, and stuck red tapes for visual reference (Figure 2).

We invited 10 participants (including 3 female and 7 male with M: 26.1 years old and SD: 2.2 years old) in this study. All of them used mobile phones everyday but had no experience in 3D drawing in mobile AR. They were all right-handed. At the beginning of the study, the participants were shown the ten scenarios. Each participant was asked to use a mobile phone (iPhone 6S in our study) as a 3D pen to trace along red tapes in all the ten scenarios to simulate 3D curve creation on and around physical objects in practice. The participants might use either a single hand or both hands. No information was shown on the screen to avoid influencing user performance. For further analysis, we recorded a video for each tracing task. The order of scenarios was given by a Latin Square. In each scenario, each participant needed to decide how to hold the mobile phone and define which point of the device to be used as an input point. They were given a few minutes to think about possible input points and grips before tracing. In the process of tracing, they could change the input point and/or grip.

Results. From all the results of the 10 participants, 104 input points and 103 grasps were collected since some participants changed their choices for input point and/or phone grip in the tracing process. As illustrated in Figure 3 (a), we categorized input points into 7 types, namely, TLC: Top left corner; CAM: Camera back; BLC: Bottom left corner; BRC: Bottom right corner; TRC: Top right corner; TEC: Top edge center; PC: Phone center. We also divided the types of phone grasp into 6 categories, as illustrated in Figure 3 (b) i–vi. Among these types, the most preferred (42.86%) input point was at the top left corner because most of these participants paired this input point with phone grip (i), and this input point was the farthest point from their right hands, just like they held a pen during drawing. 22.86% of the participants defined the input point at the position of the back camera since they believed it would be more natural when moving the phone on the surface.

For the types of phone grip, most of the participants (57.69%) held the phone from its bottom left corner using their right hands, because they claimed that they usually hold a mobile phone using their right hands and this grip type is the most similar way to they do in daily lives. 24.04% of the participants held the phone from the right side using their right hands since it was similar to holding a pen. From Figure 3(b), girp types (i) and (v) are symmetric, but (i) was much more popular than (v) since the participants were all right-handed and used to hold pens using their right hands. Grip (i) and (iii) are also similar except for the facing orientation of screen. Most of the participants with grip (iii) argued that it had no difference with grip (i). Figure 3 (c) shows the percentages of input point and phone grip. We suspect that for users with left-handedness, their preference for input point and phone grip would be somewhat similar to our findings but mirrored.

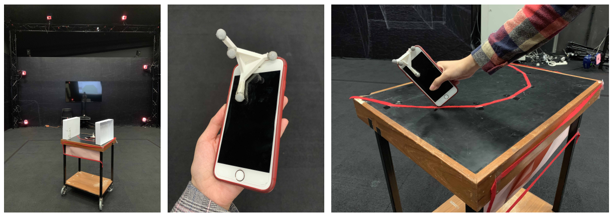


Fig. 4. Quantitative experiment on tracking errors. Left: MoCap system and setup. Middle: Tracking markers on the phone. Right: One test case of curves on surface.

We also found that the participants usually paired the choices of input point and grasp. We categorized the collected pairs into 21 types. Figure 3 (c-right) shows the percentage of each pair. Pair TLC-i was most preferred (40.19%), where the participants defined the input point at the top left corner of the device in the front side, and held the phone at its bottom-right corner. Similarly, BLC-ii (13.08%) and CAM-i (11.21%) were the second and third favorable pairs.

3.2 Quantitative Experiment on Tracking Errors

To understand how different choices of input point and phone grip affect tracking errors during in-situ 3D curve creation, we ran a quantitative experiment. Since we already found through the previous elicitation study that pairs TLC-i, BLC-ii, and CAM-i are the most preferred combinations of input point and phone grip, we focused on studying the tracking errors for them.

We implement a 3D curve creation system based on ARKit by Apple. ARKit provides reasonably accurate and real-time estimation of the device's 3D pose using VIO [1], when there are sufficient visual features in the camera view and the device is moving smoothly. Thus we extract the position data of the device in real-time to form a 3D curve (e.g., red curves in Figure 5), which approximates the moving trajectory of the mobile phone.

Setup and Procedure. To calculate motion tracking errors, we need to have the ground truth data for the trajectory of a mobile phone for the comparison with the corresponding trajectory returned by ARKit. To achieve this, we used an accurate outside-in MoCap system, Motion Analysis [14] with 24 infrared cameras, as shown in Figure 4 (Left). We attached four tracking markers on an iPhone 6S with iOS 12.1.4 (Figure 4 (Middle)) to accurately track the phone. To improve the tracking quality of ARKit, we placed multiple objects on the ground to generate more visual features. Since the purpose of this study was rather objective, one of the authors, instead of invited participants, performed this experiment. In total, five representative test cases similar to those used in the elicitation study on input point and

1. https://developer.apple.com/documentation/arkit/world_tracking/understanding_world_tracking

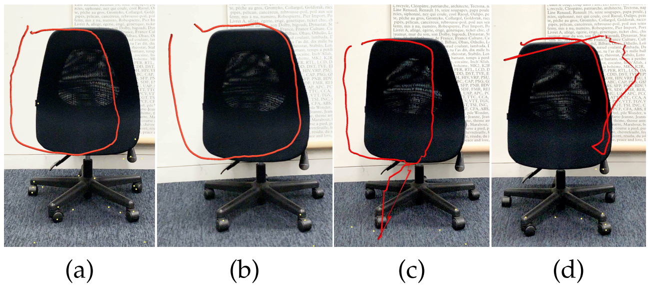


Fig. 5. Examples of error types when creating a 3D curve with the back of a chair as reference. (a) Global world drift, (b) Local accumulated drift, (c) Local sudden track loss, (d) Local short-term track loss.

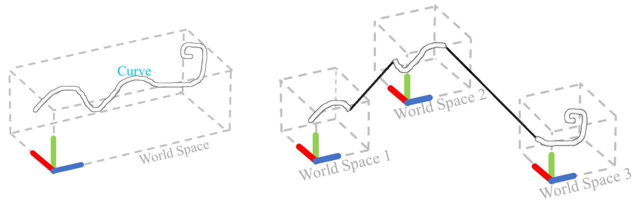


Fig. 6. A world coordinate system might be re-estimated due to the sudden track loss, which breaks a desired curve.

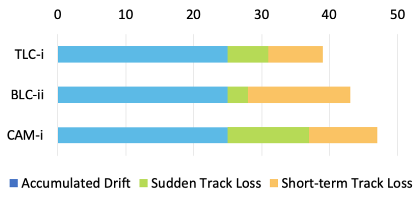


Fig. 7. Frequency of local tracking errors for each pair of input point and phone grip.

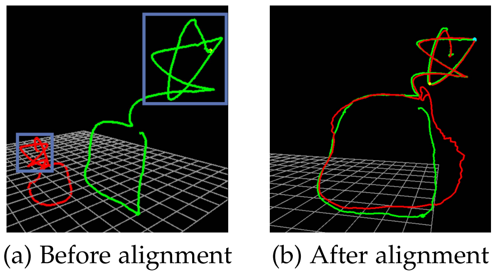


Fig. 8. (a) The curves generated by ARKit and MoCap were not aligned. (b) We used the corners of a specific shape (in purple box) to align the curves.

phone grip were performed, including curve creation along edge, on surface (Figure 4 (Right)), from edge to edge, from surface to surface, and from edge to surface. We set up these cases with a portable desk, a chair and some cardboards (Figure 4 (Left)). We again attached red tapes on the setup to represent tracing targets. We also controlled the orientation of the phone the same as the participants performed in the previous study. In each case, we performed each tracing task with the chosen top-ranked pairs of input point and phone grip for fives times, respectively. This resulted in $5 \times 5 \times 3 = 75$ pairs of the ARKit tracking results and ground truth data.

Results. We discuss the motion tracking errors in 75 ARKit results from the aspects of type, frequency, and measurement. First, we manually categorize tracking errors into global errors (Figure 5 (a)) and local errors (Figure 5 (b - d)). Global errors are due to the world coordinate global drift, resulting in the drift of an entire curve. In contrast, local errors occur in part of the curve. We further divide them into three categories. The first one is *accumulated drift* (Figure 5 (b)). Due to the forward estimation strategy of local motion, tracking errors might be accumulated gradually and make the estimated camera pose drift from its actual location. The second one is *sudden track loss* (Figure 5 (c)). Motion tracking estimates a camera's pose relative to the real world. Possibly due to insufficient visual features, motion tracking algorithms might suddenly re-estimate a virtual world coordinate system. This results in curve segments drawn in different world coordinate systems, and breaks a desired curve into multiple segments with undesired gaps between them, as illustrated in Figure 6. We found that ARKit often re-estimates only the origin of the coordinate system and keeps the axes largely not changed. The third one is *short-term track loss* (Figure 5 (d)). In this case, the tracking algorithm works poorly for a short period. When tracking continues to be robust, the desired curve segment drawn during track loss has to be coarsely estimated based on the IMU data (Figure 1). This type of error might also lead to the re-estimation of the world coordinate system.

Second, we observed and compared the curves from ARKit and ground truth. If a curve by ARKit contains the errors like Figure 5 (b)-(d), we determined this curve had such errors. We manually counted (by carefully examining visualized curve results) the number of curves where each local tracking error occurred for each pair of input point and phone grip, and as summarized in Figure 7. For each pair, we had conducted tracing tasks for 25 times. It was found that accumulated drift occurred every time, meaning it was the most common type of error of ARKit. Sudden track loss had the lowest frequency. Pair TLC-i had the fewest error times among the three pairs. Pair BLC-ii has the fewest sudden track loss times, but had the most short-term track loss times. Pair CAM-i had a high frequency of sudden track loss, causing large shift of some curve segments. Since the coordinate systems of ARKit and MoCap are independent, we could not directly count the occurence times of global errors. But we observed that in this experiment global errors occurred almost every time.

We quantitatively measured the error of each 3D curve drawn by ARKit against the corresponding curve by MoCap using the following method. Since the coordinate systems and the time stamps of the MoCap system and ARKit are independent, their generated curves were initially not aligned (Figure 8(a)). To address this issue, we performed a calibration process. To do so, we simply drew a specific shape, "star" shape in our case (purple box in Figure 8), in a good tracking condition before drawing a test curve. We manually annotated the corners of the shape, and these corners provided the correspondence between the space of MoCap and ARKit. We manually look for the best alignment to register the curves (Figure 8 (b)). Once the two curves were well aligned, we manually trimmed the curves for alignment and the unwanted extra line segments before

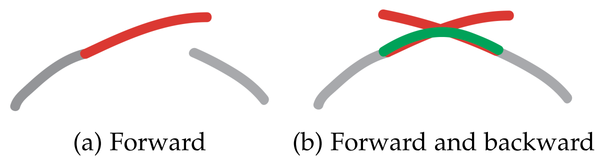
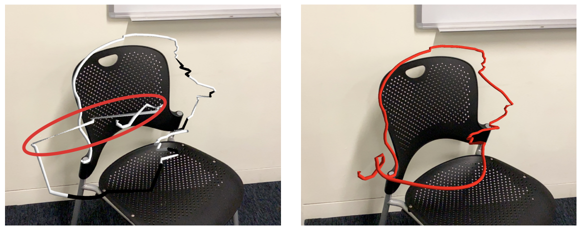


Fig. 9. A curve segment (in red) due to short-term track loss can be estimated between adjacent curve segments (in gray) based on the acceleration data from IMU. A fused curve (in green) from both forward and backward integration of acceleration reduces accumulation errors and better connects the curve segments in gray than using forward integration alone.



(a) Original result (b) Initial corrected result

Fig. 10. Initial correction removes sudden track loss (red circle area) and refines short-term track loss (black segment) in (a) automatically.

(after) reaching (exiting) the cardboards, and then calculated the errors. The error $E(P)$ between a curve P and ground truth G is defined below:

$$E(P) = \max\left(\frac{1}{N_G}\sum_{i}^{N_G} D(G_i, P), \frac{1}{N_P}\sum_{i}^{N_P} D(P_i, G)\right), \quad (1)$$

where N_G and N_P are the numbers of points in curve G and P. G_i and P_i are the i-th points in the corresponding curves. $D(.)$ is the shortest distance between a point and its nearest correspondence on the other curve. We took the maximum on each direction to suppress the errors caused by data sampling.

We conducted a one-way ANOVA to compare the effect of pairs of input point and phone grip on the errors: a significant effect was found for the three conditions [$F_{(2,72)}$ = 3.372, p = 0.040]. Post hoc comparisons using Fisher's Least Significant Difference indicated that the mean error of TLC-i (M: 0.097, SD: 0.066) was significantly lower (p = 0.016) than CAM-i (M: 0.350, SD: 0.603). BLC-ii (M: 0.150, SD: 0.167) also resulted an almost significantly lower error (p = 0.056) than CAM-i. However, there is no significant difference (p = 0.606) between TLC-i and BLC-ii. The results here suggest that the choices of input point and phone grip did have an effect on the errors of in-situ curve construction in mobile AR. The unit of all the errors here is meter.

Summary. Among the top three preferred pairs of input point and phone grip, TLC-i had the smallest errors considering types, frequency and measurement. BLC-ii was ranked only second to TLC-i and worked well in some cases. CAM-i had the largest errors, mainly due to severe camera occlusion during curve construction. Therefore, for users holding a mobile phone in their right hands, we suggest to use TLC-i for 3D curve creation tasks in mobile AR for both natural interaction and low initial tracking errors. If users prefer to hold the device with their left hands, we suggest a mirrored input point and phone grip. Nevertheless, the tracking errors cannot be ignored, since they lead to undesired curves. Next we will present a system supporting both in-situ 3D curve construction and interactive curve refinement to reduce tracking errors.

4 In-situ Curve Construction and Refinement

As shown in Figure 1, our system allows a user to first create 3D curves on and around an object using a single mobile phone as a 3D pen, followed by optional curve refinement through a specially designed touch-based interface. Following the results of the previous studies, by default we set the input point to the top-left corner of the phone and suggest the user to hold the device similar to Figure 3 (b-i) when the user decides to use his/her right hand to hold the device. The user can change the desired input point and phone grip (e.g., the top-right corner of the device for left-handed users) before starting creating curves. During absolute drawing, due to the coupling of the input and output devices, the user often cannot see the phone screen. We thus use a special pan gesture to start and finish drawing, to support absolute drawing in an eyes-free manner (without looking at the mobile screen) so that the user can focus more on the close interaction between the device and physical references. To use this gesture, the user moves a finger (typically their thumb) on the screen towards any direction for a certain distance (100 pixels in our implementation), with vibration feedback for activation. After it is activated, the user can release the finger from the screen for free drawing. Once a curve is created, the user may examine the result by ARKit and interactively refine it through a touch-based interface.

Resulting 3D curves in our system are represented as 3D polylines, with the parameterization based on the nearly constant sampling rate of ARKit. A new point is added to a curve being drawn if its distance to its previous point is larger than a threshold. For every curve point we also store its corresponding IMU data and a tracking state. For example, a point might be marked as "track-loss" by ARKit due to short-term track loss (black curve segments in Figure 10).

Our curve refinement method has two phases. In the first automatic phase, it performs initial correction by automatically identifying and reducing the errors due to the sudden and short-term track losses using the IMU data. In contrast the accumulated drift is difficult to detect automatically and thus handled interactively. In the second interaction phase, the initially corrected curve is refined with the help of user-specified hints for instance for relocating the world coordinate, specifying key points the curve must pass through, 2D sketching and 2D edge snapping. Every time a new user-specified hint is given, the system performs global curve optimization, which incorporates the information from the initially refined curve and user-specified hints.

4.1 Initial Correction

As discussed previously, the sudden track loss would result in sudden gaps between adjacent points (see Figure 6 for an illustration and Figure 10 (a) for a real-world example).

Such a sudden track loss case often occurs when there is insufficient visual feature, and usually lasts several seconds. During these short periods, the world coordinate system including the previously created curve segments drifts away from its current positions. When the tracking recovers, there will be a large distance (often over three times larger than the average distance of adjacent points) between points before and after sudden track loss. Following this observation, we thus check every pair of adjacent points and confirm the sudden track loss if the distance between the adjacent points is above a threshold (three times larger than the average distance of adjacent points, in our implementation). To correct the caused error, we remove the original edge between the adjacent discrete sampled points, and estimate the velocity (i.e., the difference of the discrete positions) between two points using a weighted average of the velocity values in a local neighborhood to connect two curve segments more smoothly.

The curve segments caused by the short-term track loss are marked as "track-loss" by ARKit (black curve segment in Figure 10 (a)). ARKit employs the acceleration data from IMU to estimate such segments. However, for the purpose of real-time tracking, ARKit makes use of the past data and performs forward integration of acceleration from a robustly tracked position as an initial point using explicit Euler, leading to increasingly accumulated errors, as illustrated in Figure 9 (a). Since our curve refinement is done in a post-processing step, we are able to perform both forward and backward integration of acceleration using explicit Euler, and then fuse the two integrated curve segments by simply using normalized weighted sum based on their distance to their starting point to suppress accumulated errors and get a better refined curve (see Figure 9 for an illustration). The position of the i-th point on the fused curve segment (e.g., the middle curve segment between two gray curve segments in Figure 9) can be formulated as follows:

$$p_i = w_f p_f + w_b p_b, \tag{2}$$

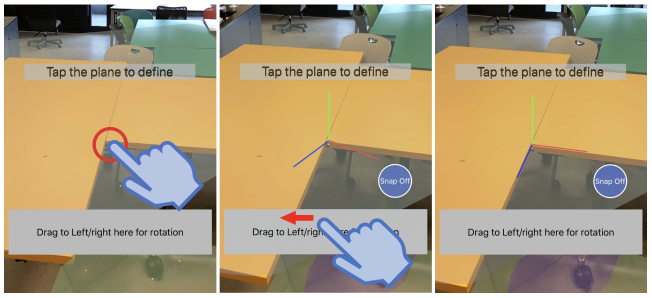
where p_f and p_b are the respective positions of i-th point using forward and backward integration, w_f and w_b are normalized weights as follows:

$$w_f = \frac{n-i}{n}, w_b = \frac{i}{n},$$

with n being the total point number of fused curve segment.

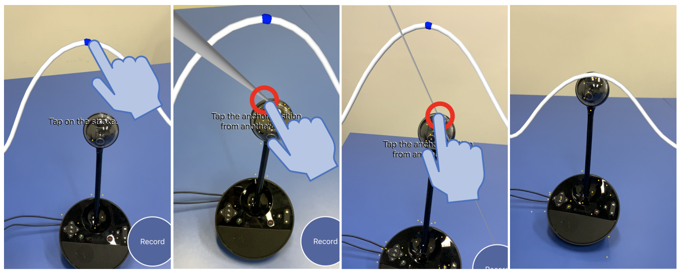
4.2 User Interface for Curve Correction

3D Point Localization. A fundamental operation needed in our system is to allow users to easily specify a target 3D position, which can be used to relocate the origin of a world coordinate system or key points of a curve. To achieve this we provide two techniques: *one-ray-plane intersection* and *two-ray intersection*. Since our curves often have close interaction with the environment, it is intuitive to interactively specify the 3D point on physical planes. For the first technique, users tap a key point on the horizontal or vertical plane detected by ARKit, to specify a 3D point on the plane (Figure 11 (a)). If users wish to locate a 3D point on a non-planar surface, they can use the second technique: tap a same image feature point (e.g., a corner) from two



(a) (b) (c)

Fig. 11. World anchor re-localization. (a) Our system detects planes (semi-transparent gray planes in the figure). The user taps on a plane (red circle) to locate the origin of the world coordinate system. (b) In this example, the origin of the world coordinate system is located at the corner on the table. The user swipes left or right to change the world system's orientation. (c) The world coordinate system is located accurately with the desired orientation at the corner of the table. Edge snapping is used in (a) to help locate along physical edges in the image. The blue hand icon, the red arrow, and the red circle are added to the figure to illustrate the gesture interaction.



(a) (b) (c) (d)

Fig. 12. Point constraint. (a) The user creates a curve around a rounded camera, but it is deviated from the top of the camera. To make the curve go through the top of the camera, the user first taps to select a point on the curve (blue point) to change. (b)(c) Then the user uses *two-ray intersection* to locate a target point (red circle) in two views. (d) The blue point will move to the target point and the whole curve is optimized to go through the target point. The blue hand icon and the red circle are added to the figure to illustrate the gesture interaction.

views (by rotating the mobile device) to specify a 3D point at the intersection of two rays originated from the two tapped positions (Figure 12 (b)(c)).

World Anchor Re-localization. As shown in Figure 11, to handle world global drift, we allow users to relocate the world coordinate system by re-localizing both its origin and orientation. If the drift happens, users utilize one of the two 3D point localization techniques to specify the original world origin, followed by optional manipulation of its orientation using a swipe gesture.

Point Constraint. When users create curves with the reference of real-world objects, they can easily specify points of real-world objects to constrain created curves for curve refinement. To do so, as shown in Figure 12, users first select a point (a blue point in Figure 12 (a)) on an already-created curve by tapping. This point is found by searching for the nearest point on the curve to the ray casting from the tapped location on the screen, since precisely selecting a small point on a phone touch screen is not easy mainly due to the fat-finger problem [48] . Then it is followed by swiping along the curve to refine its position, and then use the two 3D point localization techniques to locate a

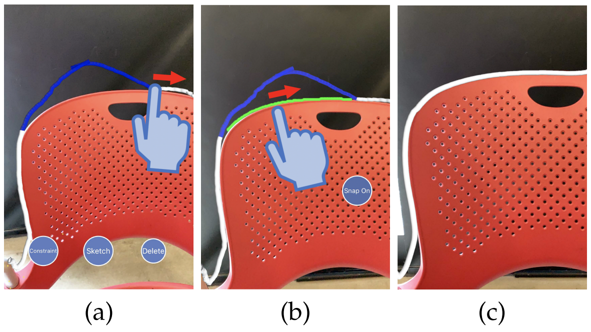


Fig. 13. 2D sketching. The user selects a curve segment (in blue in (a)), and draws a desired 2D curve (in green in (b)) in a frozen view. Edge snapping is used in (b) to help create accurate curves. The blue hand icon and the red arrow are added to the figure to illustrate the gesture interaction.

desired 3D position. Finally the entire curve is optimized to go through the specified point. By specifying several key points, users may quickly reduce accumulated drift, which is the most frequently tracking error during curve creation.

2D Sketching. To get more elaborate results after removing short-term track loss, we provide a 2D sketching interface to change the shape of a selected curve segment (Figure 13). The user needs to sketch 2D curves of the selected curve segment in one or more views. Our system then updates the 3D shape of the selected segment with respect to the 2D curves. To do so, we sample each 2D curve with the same number of vertices as the selected 3D segment to build their correspondence. We then calculate the rays that pass through the center point of projection (i.e., camera) and the sampled points on the projection plane. For single-view sketching, we simply determine the depth of the sampled points based on the depth value of their correspondence points on the original 3D curve. For multi-view sketching, each vertex has several rays. We calculate the middle points on the shortest line segment between every pair of rays, and calculate the average point of these middle points as the new position.

2D Edge Snapping. For 3D curves highly interacting with physical edges, we offer a 2D edge snapping feature (Figure 13 (b)) to change the curve shape more intelligently. When the user sketches 2D curves of a target 3D curve from single or multiple views, our system automatically detects the main edges of the image in each view. Our system then snaps the sketched curve(s) to the closest edge(s) on the 2D screen [49], and thus snaps the 3D curve to the image edges. This feature is also helpful for 3D point localization (Figure 11).

4.3 Global Curve Optimization

After initial correction and user interaction, we get a coarsely refined curve, which is then further optimized. Our problem is essentially a special case of a well-studied constrained feature-preserving shape deformation problem [50]. Our objective function consists of a position term, a shape term and an optional smoothness term.

Position Term. During interactive refinement, the user specifies some target 3D positions that the curve must pass through. Let C denote the index set of curve points with the corresponding target positions. We design the position term to let the curve go through user-specified constraint points, and keep the remaining points of the optimized curve as similar to the coarsely refined curve as possible. Specifically, it is formulated as follows:

$$T_p = \sum_{i \in C} w_c \|p_i - \hat{p_{c_i}}\| + \sum_{i \notin C} w_i \|p_i - \hat{p}_i\|, \tag{3}$$

with $w_c = 1000$ and $w_i = min(1.0, N_i/T)$. Here, p_i is the i-th unknown 3D position of the curve to be optimized. $\hat{p_{c_i}}$ and $\hat{p}_i$ are the corresponding user-specified 3D position and the latest 3D position of the point after the initial correction and editing stages. To prioritize the curve going through the user-specified positions, we set the weight w_c as a high value. We set the weight w_i according to its confidence level, which is defined as a function of the number of visual features N_i during 3D drawing of the i-th point in mobile AR. We set a threshold $T = 40$ as the upper bound of the confidence value. Although the relative weight for the second term is not large compared to the first term, there are often more points contributing to the second term.

Shape Term. This term is to preserve the shape of the optimized curve as similarly to the coarsely refined curve as possible. It is defined as follows:

$$T_s = \sum \|\delta(p_i) - \delta(\hat{p}_i)\|, \tag{4}$$

where $\delta(p_i) = p_i - \frac{1}{2}(p_{i-1} + p_{i+1})$ is the distance between the point p_i and the center formed by its previous and following generated points, and is a differential representation of local shape.

Smoothness Term (optional). Finally, if the user feels the original curve suffers from noise for example due to tracking errors and/or hand shaking during curve creation, he/she may enforce an *optional* smoothness term. Specifically, it is formulated as:

$$T_m = \sum \|\delta(p_i)\|. \tag{5}$$

It assumes the distances between the point p_i, and its adjacent points should be almost the same.

Objective Function. Our final objective function is defined as the weighted sum of all the above terms:

$$F_a(V) = T_p + W_s T_s + W_m T_m, \tag{6}$$

where W_s and W_m are the weights for the shape and smoothness terms, respectively. We always set $W_s = 100.0$ and $W_m = 30.0$, respectively, in our implementation. $V = \{p_i\}$ is a set of 3D unknown positions of the curve to be optimized. Note that minimizing the above objective function is equivalent to solving a sparse linear system. It thus can be efficiently solved by existing linear solvers, e.g., the QR factorization using the sparse solver in iOS in our implementation. It achieves interactive curve correction in real time, as shown in the supplementary video.

5 Evaluation

5.1 Qualitative Study

To evaluate the effectiveness of our system, we first employed our system to create 3D curves on and around real-world objects such as chair and shelf, as shown in

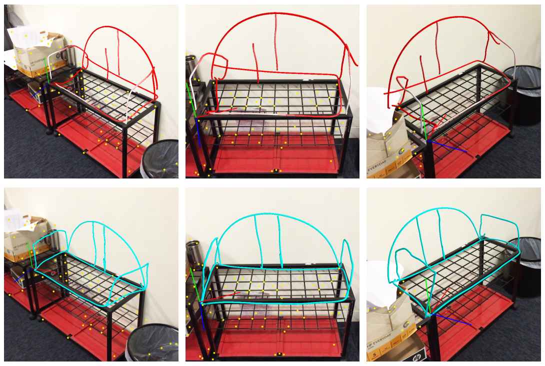


Fig. 14. 3D curve creation on and around a shelf (6 strokes; 4.0 mins including 0.4 mins for drawing and 3.6 mins for correction). The top row is the initial result by ARKit. The bottom row is our result after interactive correction.

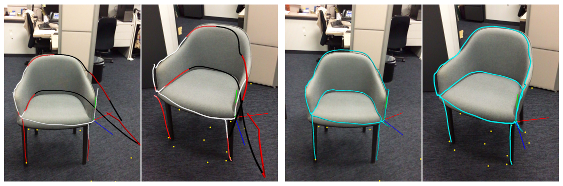


Fig. 15. Drawing multiple curves along a chair (5 strokes; 2.4 mins including 0.3 mins for drawing and 2.1 mins for correction.). The left two are the results by ARKit in two views. The right two are our interactive editing results.

Figures 1, 14, and 15. We present both initial curves by ARKit and our refined results to show the effectiveness of interactive correction. We found that although ARKit sometimes produced very poor results due to very close interaction between the mobile phone and real-world objects, our system still helped refine the curves effectively. The completion time and the number of operations for drawing and refinement were collected. For the examples in Figures 1, 14, and 15, on average 6.7 strokes (SD: 2.1) were created, and 2.7 world anchor localization operations (SD: 0.6), 10.3 point constraints (SD: 0.6) and 2.3 2D sketching operations (SD: 0.6) were used. The average time was 4.2 minutes (SD: 2.0), including 0.5 minutes (SD: 0.2) for drawing and 3.7 minutes (SD: 1.8) for correction. The total time required to produce each example was shown in the captions of the corresponding figures.

We also recruited four university students (including 2 female and 2 male with M: 25.5 years old and SD: 0.58 years old) to perform in-situ curve creation tasks. One participant (P4) had tried mobile 3D sketching apps for a few times. The other participants had no experience in 3D drawing in mobile AR. All of them had not used our proposed system before. They were given a 10-minute tutorial on our system, and asked to create 3D curves on and around real-world objects. Note that since initial curves needed to be created with respect to real-world objects often in an eyes-free manner, they were suggested to focus on the interaction between curves and reference objects, instead of the connectivity between curves. Figure 16 shows representative results created by the participants.

We get various use cases such as tracing curves on objects potentially for 3D modeling, 3D printing and fabrication (Figure 16 (a)-(d)), sketching curves on and around objects for decoration or artistic creation (Figure 16 (e)-(m)), and direct measurement of complex curves (Figure 16 (n)-(s)).

For drawing results (Figure 16 (a)-(m)), on average 6.1 strokes (SD: 4.8) were created, and 3.2 world anchor localization operations (SD: 2.2), 11.7 point constraints (SD: 11.7), and 2.8 2D sketching operations (SD: 4.0) were used for each result. The average time was 10.3 minutes (SD: 10.5), including 0.5 minutes (SD: 0.4) for drawing and 9.8 minutes (SD: 10.1) for correction. For the measuring results (Figure 16(n)-(s)), the mean error (absolute error) and error rate (relative error) between our results and ground truth were 3.87 centimeters (SD: 4.53) and 2.27% (SD: 2.37%) respectively.

Generally, the feedback collected from four users shows that our system provides an effective tool for in-situ 3D curve creation:

Usefulness. All the users thought our system was very useful to create 3D curves. *P4: "In my limited previous experience of using AR drawing tool, it was only applicable to draw far away from physical objects. I am happy to use this system to put the phone very close to desks, chairs and so on." P1: "I think I can correct the wrong curves well, though the original results sometimes have very big errors."*

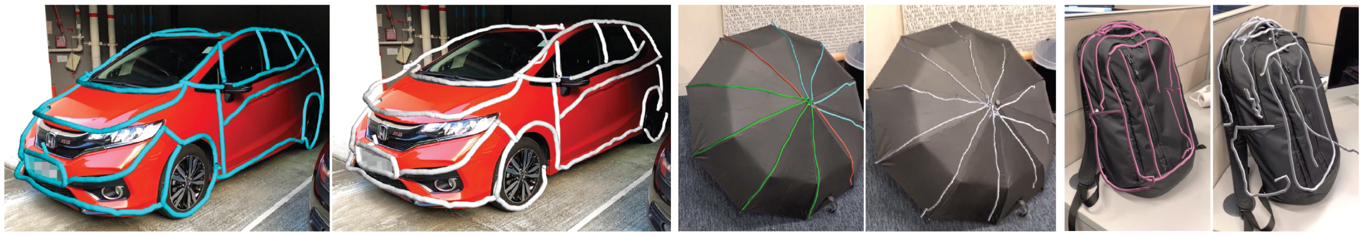
Interaction. We also got some positive comments on our interface. *P1: "I think the most powerful function is point constraint, allowing me to drag the whole curve to my desired positions quickly. When I sometimes draw on planes, the functions of edge snapping and plane tapping let me locate the world origin quickly."* However, users also pointed out the world coordinate drift, caused by unreliable tracking of ARKit on textureless surfaces, took them a lot of time to relocate it. *P2: "The world often drifts when I snap the phone to objects. In fact I spent most of the time on relocating the origin."*

Applications. All the users thought our system has various promising applications. *P3: "I can use it for 3D printing. For example, I can use the phone to draw a hat along the edges of a real hat. Then I will 3D print the hat."* The advantage of our system is that the created hat has a physical size property. *P2: "It is very useful for fast-modeling physical objects. I can use it to scan complex objects as virtual sketching models." P1: "I like the measurement function in the system. I can measure some curves quickly."*

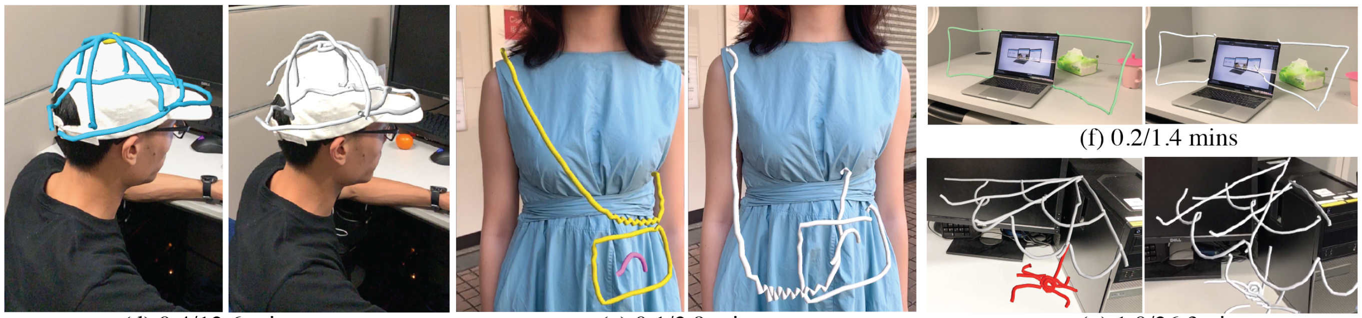
5.2 Applications

As observed by our user study participants, our system can be potentially applied to 3D sketching, 3D modeling, 3D animation, and 3D measurement, etc.

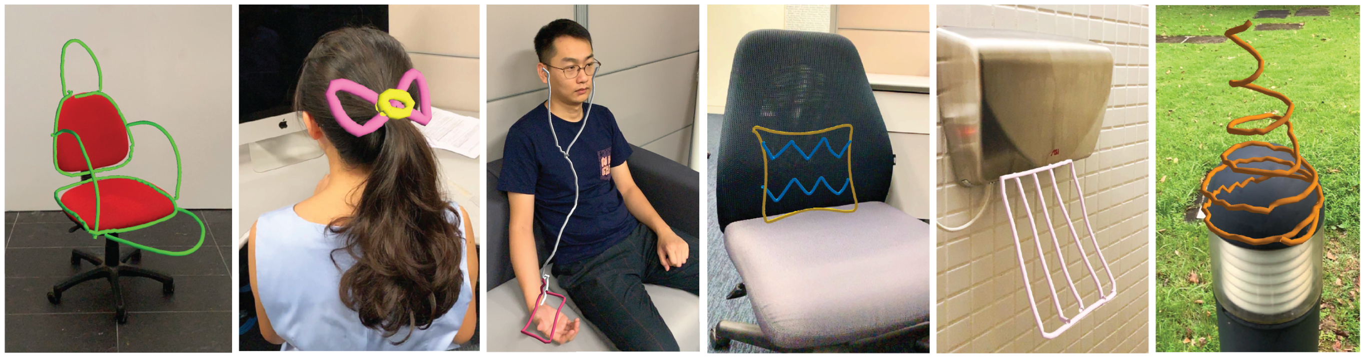
Due to our focus on absolute drawing in an eyes-free manner, our current sketching results are not complex. However, our work demonstrates unique use cases, which are difficult to achieve with existing mobile AR sketching applications. Thus, our system can be utilized in 3D sketching applications like artistic creation [51], [52], and our tool can provide artists with an intuitive interface for drawing desired 3D curves. To support the creation of more complex 3D sketches for example for the application of 3D conceptual design, our tool can be used for creating 3D canvases on and around physical objects [1] and then



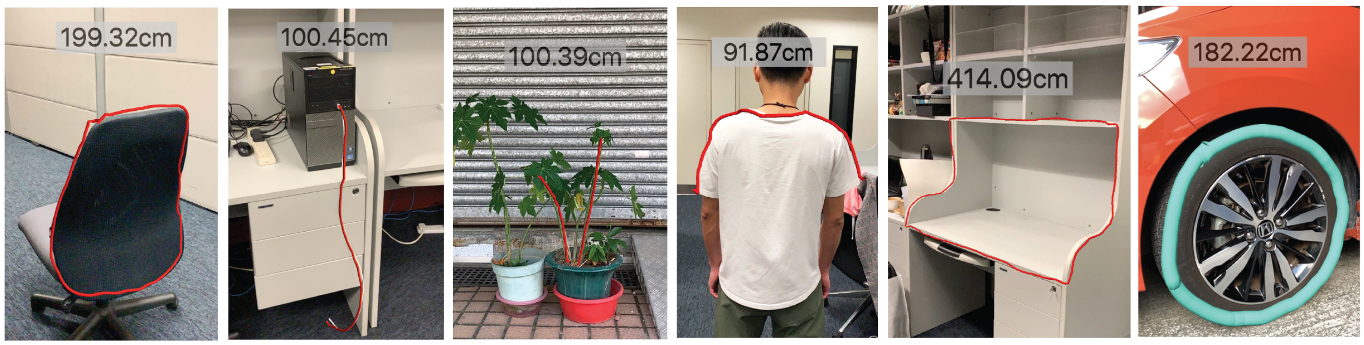
(a) 1.3/20.2 mins (b) 0.8/19.7mins (c) 0.6/27.6 mins



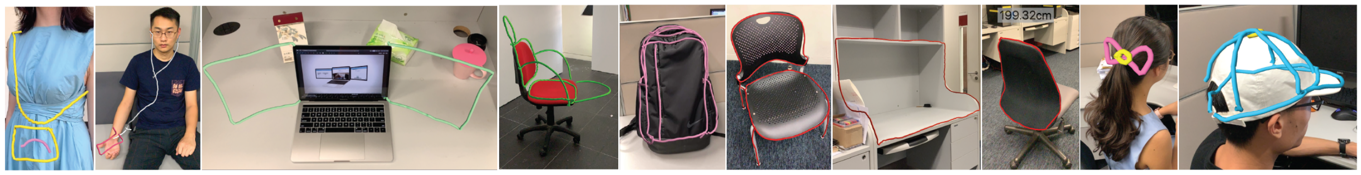
(d) 0.4/12.6 mins (e) 0.1/2.9 mins (f) 0.2/1.4 mins (g) 1.0/26.3mins



(h) 0.5/5.0 mins (i) 0.2/0.9 mins (j) 0.2/3.8 mins (k) 0.2/2.3 mins (l) 0.5/4.1 mins (m) 0.5/1.0 mins



(n) gt: 197.2 cm (o) gt: 100.2 cm (p) gt: 104.0 cm (q) gt: 90.5 cm (r) gt: 411.0 cm (s) gt: 195.0 cm



Other views

Fig. 16. A gallery of 3D curve creation results produced by the four test users of our system (from (a) to (m)). The time (t1/t2) below each result refers to curve drawing time (t1) and curve correction time (t2). Due to the limited space, we only include the selected examples (a) to (g)) for the comparisons between the results after (Left) and before (Right) refinement, and the selected examples for other views. From (n) to (s) we show representative 3D measuring results supported by our system. Here "gt" corresponds to the ground-truth length.

be integrated with relative drawing supported by various snapping mechanisms [20]. It is interesting to investigate the choices of input point and phone grip during mid-air drawing without any reference object.

3D modeling is another possible application of our system. Our tool is especially useful for modeling 3D parts that are used to augment existing physical objects. With our tool, users can directly create 3D models on and around physical objects, without digitizing physical objects in advance.

An AR-enabled mobile device can be used as a 6-DoF controller [53] to achieve in-situ 3D character animation in mobile AR. Our system allows the creation of a 3D moving trajectory of a virtual character interacting with a physical environment. For example, our technique can be integrated into ARAnimator [53], our recent work for in-situ character animation creation, since a mobile device is used to directly control a virtual character and closely interact with real-world environments.

We have already shown that our technique can be used as a direct 3D measuring tool, as shown in Figure 16 (n)-(s). Unlike existing AR measuring apps [9], [10], [11], which only support the measurement on planar surfaces detected by environment understanding, our tool allows users to directly use a mobile phone as a measuring wheel, supporting reasonably accurate measurement even on complicated surfaces.

6 Discussions

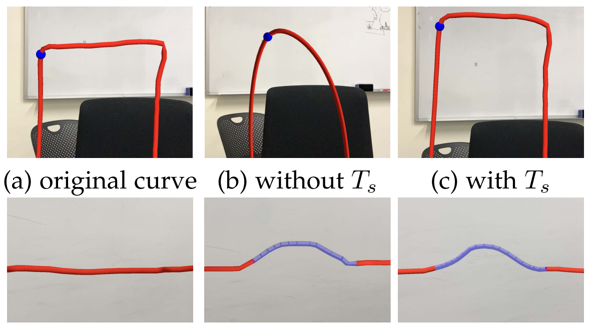
Here we discuss different aspects of our work, including comparison with relative drawing systems, comparison with vision-based approaches, objective function, limitations, and future works.

6.1 Comparison with Relative Drawing Systems

Our system is not a complete 3D drawing tool. Instead, our system provides an absolute drawing solution, which is good at creating individual curves of complicated shape. Thus, the demonstration of the results (Figure 16) using our system focuses on comparing the curves before and after refinement, and the individual curve creation on and around physical objects, instead of presenting compelling 3D sketching results. We expect its integration with relative drawing features from Mobi3DSketch [20] for a more complete and powerful drawing system. Such systems like SymbiosisSketch [1] and Mobi3DSketch [20] are useful for applications like 3D conceptual design. Similarly, we would like to clarify that absolute drawing can potentially benefit but itself is not sufficient for 3D modeling and fabrication.

6.2 Comparison with Vision-based Approaches

In this work, we present a direct manipulation interface on a mobile device for in-situ curve creation. Vision-based approaches such as 2D edge detection and snapping are utilized in our current implementation to facilitate user interaction. With the current advances in computer vision technologies, it is possible to extract 3D curves along physical edges of real-world objects using the approaches of edge detection and depth estimation. But for curves around physical objects (e.g., those in Figure 16 (e)(g)(i)(j)), it is


(a) original curve (b) without T_s (c) with T_s

(a) original curve (b) without T_m (c) with T_m

Fig. 17. The effects of shape term T_s and smoothness term T_m.

very challenging to create them only using the vision-based methods, since these curves do not exist in the real world. Thus, computer-vision-based approaches cannot work individually as an alternative to our system, but they can work as a supplementary for more accurately recovering 3D curves existing in physical environments. In the future, we are considering to involve more advanced vision-based methods in our system to improve the curve quality and reduce the amount of user interaction. For example, we may recover the object geometry to facilitate more effective snapping of individual points or curve segments to physical edges, planes or surfaces.

6.3 Objective Function

As mentioned in our system design, a global optimization, involving three energy terms, is utilized to find user-desired results. Here, we discuss the impact of individual terms. The position term describes that the corrected curve should go through some target 3D positions, and the remaining points should be near to the position of original ones. The goal of this term is trivial and easy to understand, and our discussions below focus more on the shape and smoothness terms.

The shape term encourages a similar shape between the corrected and original curves. This prevents the optimization dramatically deforming the shape of the optimized curve. Figure 17 (b & c) shows the results of optimization without and with the shape term, respectively. It is obvious that the global shape of the curve is better preserved with the shape term.

The smoothness term encourages the smoothness and uniform distribution of points in the corrected curve. The comparison in Figure 17 (b & c) shows the usefulness of this term. Without the smoothness term, there are sudden changes at the connection of a modified part and the original stroke.

Although our current optimization can provide reasonably good results, we can exploit more optimization strategies including adding other terms to improve the curve quality and the user experience, by considering the uncertainty of SLAM, considering the errors caused by manual input, and using the frames from the camera to re-evaluate the optimization results. For example, we can calculate the distance of each point on the curve to the nearest planes detected by ARKit, and then infer whether we can snap such points to the nearest planes according to the calculated distance and the state of the neighboring points. Besides, we can leverage the movement of real-world objects from camera frames to enhance or re-evaluate the optimized results by comparing such movement with the curve trend.

6.4 Limitations and Future Work

Our current system suffers from several limitations. First, we have requirements on starting views for 3D drawing. Users are asked to start to draw a curve from a view with sufficient visual features. This is necessary because the subsequent correction is based on previously drawn curve segments and thus it is important to guarantee the starting segment with high reliability. But it also restricts user interaction from more freedoms. In our future work, we plan to provide more efficient interaction techniques to allow users to start from any locations with any viewpoints.

Second, during the interaction phase, if there are very limited visual features in the environment, the world coordinate system drifts very often, requiring users to frequently relocate the origin of the world coordinate. This is very time consuming. We will make the world coordinate system locate more robustly using previous sufficient visual features.

Acknowledgments

We thank the anonymous reviewers for the constructive comments. This work was supported by an unrestricted gift from Adobe and grants from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. CityU 11212119), City University of Hong Kong (No. 7005590 and No. 7005176), and the Centre for Applied Computing and Interactive Media (ACIM) of School of Creative Media, CityU.

References

[1] R. Arora, R. Habib Kazi, T. Grossman, G. Fitzmaurice, and K. Singh, "Symbiosissketch: Combining 2d & 3d sketching for designing detailed 3d objects in situ," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, 2018, pp. 1–15.

[2] B. Jackson and D. F. Keefe, "Lift-off: Using reference imagery and freehand sketching to create 3d models in vr," *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, no. 4, pp. 1442–1451, 2016.

[3] C. Grimm and P. Joshi, "Just drawit: a 3d sketching system," in *Proceedings of the International Symposium on Sketch-Based Interfaces and Modeling*, 2012, pp. 121–130.

[4] V. Krs, E. Yumer, N. Carr, B. Benes, and R. Měch, "Skippy: Single view 3d curve interactive modeling," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, pp. 1–12, 2017.

[5] K. Huo and K. Ramani, "Window-shaping: 3d design ideation by creating on, borrowing from, and looking at the physical world," in *Proceedings of the Eleventh International Conference on Tangible, Embedded, and Embodied Interaction*, 2017, pp. 37–45.

[6] M. Thorne, D. Burke, and M. van de Panne, "Motion doodles: an interface for sketching character motion," *ACM Transactions on Graphics (TOG)*, vol. 23, no. 3, pp. 424–431, 2004.

[7] B. Choi, R. B. i Ribera, J. P. Lewis, Y. Seol, S. Hong, H. Eom, S. Jung, and J. Noh, "Sketchimo: sketch-based motion editing for articulated characters," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 4, pp. 1–12, 2016.

[8] M. Guay, R. Ronfard, M. Gleicher, and M.-P. Cani, "Space-time sketching of character animation," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 4, pp. 1–10, 2015.

[9] GRYMALA, "Ar ruler," https://apps.apple.com/us/app/ar-ruler-app-tape-measure/id1326773975, 2017.

[10] L. Labs, "Airmeasure," https://apps.apple.com/us/app/airmeasure-ar-tape-ruler/id1251282152, 2017.

[11] GRYMALA, "Ar plan 3d," https://apps.apple.com/us/app/ar-plan-3d-camera-to-plan/id1459846158, 2019.

[12] J. M. Cohen, L. Markosian, R. C. Zeleznik, J. F. Hughes, and R. Barzel, "An interface for sketching 3d curves," in *Proceedings of the 1999 symposium on Interactive 3D graphics*, 1999, pp. 17–21.

[13] O. A. Karpenko, J. F. Hughes, and R. Raskar, "Epipolar methods for multi-view sketching." in *SBM*, 2004, pp. 167–173.

[14] M. A. Corporation, "Motion analysis," http://motionanalysis.com/, 1982.

[15] I. NaturalPoint, "Optitrack," https://optitrack.com/, 1996.

[16] M. Li and A. I. Mourikis, "High-precision, consistent ekf-based visual-inertial odometry," *The International Journal of Robotics Research*, vol. 32, no. 6, pp. 690–711, 2013.

[17] E. Nerurkar, S. Lynen, and S. Zhao, "System and method for concurrent odometry and mapping," Oct. 13 2020, uS Patent 10,802,147.

[18] Z. Assets, "Paint space ar," https://www.paintspacear.com/, 2018.

[19] Google, "Just a line," https://justaline.withgoogle.com/, 2018.

[20] K. C. Kwan and H. Fu, "Mobi3dsketch: 3d sketching in mobile ar," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, 2019, pp. 1–11.

[21] D. F. Keefe, D. A. Feliz, T. Moscovich, D. H. Laidlaw, and J. J. LaViola Jr, "Cavepainting: a fully immersive 3d artistic medium and interactive experience," in *Proceedings of the 2001 symposium on Interactive 3D graphics*, 2001, pp. 85–93.

[22] S. Schkolne, M. Pruett, and P. Schröder, "Surface drawing: creating organic 3d shapes with the hand and tangible tools," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, 2001, pp. 261–268.

[23] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual–inertial odometry using nonlinear optimization," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, 2015.

[24] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-manifold preintegration for real-time visual–inertial odometry," *IEEE Transactions on Robotics*, vol. 33, no. 1, pp. 1–21, 2016.

[25] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[26] J. Ventura, C. Arth, G. Reitmayr, and D. Schmalstieg, "Global localization from monocular slam on a mobile phone," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 4, pp. 531–539, 2014.

[27] A. Angeli, D. Filliat, S. Doncieux, and J.-A. Meyer, "Fast and incremental method for loop-closure detection using bags of visual words," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 1027–1037, 2008.

[28] Z. Yan, M. Ye, and L. Ren, "Dense visual slam with probabilistic surfel map," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 11, pp. 2389–2398, 2017.

[29] T. Taketomi, H. Uchiyama, and S. Ikeda, "Visual slam algorithms: a survey from 2010 to 2016," *IPSJ Transactions on Computer Vision and Applications*, vol. 9, no. 1, p. 16, 2017.

[30] S. Ryan, K. Azam, S. Karan, and K. Gord, "Analytic drawing of 3d scaffolds," *ACM Transactions on Graphics (TOG)*, vol. 28, no. 5, p. 149, 2009.

[31] S.-H. Bae, R. Balakrishnan, and K. Singh, "Ilovesketch: as-natural-as-possible sketching system for creating 3d curve models," in *Proceedings of the 21st annual ACM symposium on User interface software and technology*, 2008, pp. 151–160.

[32] B. Xu, W. Chang, A. Sheffer, A. Bousseau, J. McCrae, and K. Singh, "True2form: 3d curve networks from 2d sketches via selective regularization," *ACM Transactions on Graphics (TOG)*, vol. 33, no. 4, pp. 131:1–131:13, 2014.

[33] T. Grossman, R. Balakrishnan, G. Kurtenbach, G. Fitzmaurice, A. Khan, and B. Buxton, "Creating principal 3d curves with digital tape drawing," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2002, pp. 121–128.

[34] Y. Kim, S.-G. An, J. H. Lee, and S.-H. Bae, "Agile 3d sketching with air scaffolding," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, 2018, pp. 1–12.

[35] Y. Kim and S.-H. Bae, "Sketchingwithhands: 3d sketching handheld products with first-person hand posture," in *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, 2016, pp. 797–808.

[36] M. Lau, G. Saul, J. Mitani, and T. Igarashi, "Modeling-in-context: user design of complementary objects with a single photo," in *Proceedings of the Seventh Sketch-Based Interfaces and Modeling Symposium*, 2010, pp. 17–24.

[37] P. Xu, H. Fu, Y. Zheng, K. Singh, H. Huang, and C.-L. Tai, "Model-guided 3d sketching," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 10, pp. 2927–2939, 2018.

[38] D. Keefe, R. Zeleznik, and D. Laidlaw, "Drawing on air: Input techniques for controlled 3d line illustration," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 5, pp. 1067–1081, 2007.

[39] R. R. Mohanty, U. H. Bohari, and E. Ragan, "Kinesthetically augmented mid-air sketching of multi-planar 3d curve-soups," in *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, vol. 51739, 2018, p. V01BT02A031.

[40] Y.-T. Yue, X. Zhang, Y. Yang, G. Ren, Y.-K. Choi, and W. Wang, "Wiredraw: 3d wire sculpturing guided with mixed reality," in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, 2017, pp. 3693–3704.

[41] P. Wacker, O. Nowak, S. Voelker, and J. Borchers, "Arpen: Mid-air object manipulation techniques for a bimanual ar system with pen & smartphone," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, 2019, pp. 1–12.

[42] M. Xin, E. Sharlin, and M. C. Sousa, "Napkin sketch: handheld mixed reality 3d sketching," in *Proceedings of the 2008 ACM symposium on Virtual reality software and technology*, 2008, pp. 223–226.

[43] M. F. Deering, "Holosketch: a virtual reality sketching/animation tool," *ACM Transactions on Computer-Human Interaction (TOCHI)*, vol. 2, no. 3, pp. 220–238, 1995.

[44] T. Babic, H. Reiterer, and M. Haller, "Pocket6: A 6dof controller based on a simple smartphone application," in *Proceedings of the Symposium on Spatial User Interaction*, 2018, pp. 2–10.

[45] T. Deselaers, D. Keysers, J. Hosang, and H. A. Rowley, "Gyropen: Gyroscopes for pen-input with mobile phones," *IEEE Transactions on Human-Machine Systems*, vol. 45, no. 2, pp. 263–271, 2014.

[46] R. Darbar and D. Samanta, "Magitext: Around device magnetic interaction for 3d space text entry in smartphone," in *2015 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, 2015, pp. 1–4.

[47] S. Agrawal, I. Constandache, S. Gaonkar, R. Roy Choudhury, K. Caves, and F. DeRuyter, "Using mobile phones to write in air," in *Proceedings of the 9th international conference on Mobile systems, applications, and services*, 2011, pp. 15–28.

[48] K. A. Siek, Y. Rogers, and K. H. Connelly, "Fat finger worries: how older and younger users physically interact with pdas," in *IFIP Conference on Human-Computer Interaction*, 2005, pp. 267–280.

[49] Q. Su, W. H. A. Li, J. Wang, and H. Fu, "Ez-sketching: three-level optimization for error-tolerant image tracing." *ACM Transactions on Graphics (TOG)*, vol. 33, no. 4, pp. 54–1, 2014.

[50] M. Botsch and O. Sorkine, "On linear variational surface deformation methods," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 1, pp. 213–230, 2007.

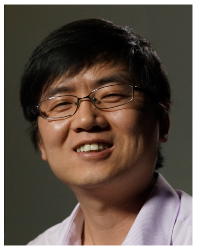
[51] J. Grey, "Badwater, hotlicks," https://digitalartarchive.siggraph.org/artwork/jen-zen-jen-grey-badwater-hotlicks/, 2002.

[52] J. Grey and S. K.-S. Burnham, "Centaur," https://digitalartarchive.siggraph.org/artwork/jen-zen-jen-grey-centaur/, 2003.

[53] H. Ye, K. C. Kwan, W. Su, and H. Fu, "Aranimator: in-situ character animation in mobile ar with user-defined motion gestures," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 4, pp. 83–1, 2020.

Hui Ye received a BA degree in communication from the University of Science and Technology of China in 2016. She is currently working towards the PhD degree in the School of Creative Media, City University of Hong Kong. Her research interests lie in the intersection between human computer interaction and computer graphics.

Kin Chung Kwan received the BSc and PhD degrees from the Department of Computer Science and Engineering, The Chinese University of Hong Kong, in 2009 and 2015, respectively. He is now a senior research associate at the School of Creative Media, City University of Hong Kong. His research interests include computer graphics, human-computer interaction, non-photorealistic rendering, GPGPU, and shape analysis.



Hongbo Fu received a BS degree in information sciences in 2002 from Peking University, China and a PhD degree in computer science from the Hong Kong University of Science and Technology in 2007. He is a Professor at the School of Creative Media, City University of Hong Kong. His primary research interests fall in the fields of computer graphics and human computer interaction. He has served as an Associate Editor of The Visual Computer, Computers & Graphics, and Computer Graphics Forum.