Real-time Hand Interaction for Augmented Reality on Mobile Phones

Wendy H. Chun
Unaffiliated*
Los Angeles, CA
wendyhchun@gmail.com

Tobias Höllerer
Department of Computer Science
University of California, Santa Barbara
holl@cs.ucsb.edu

ABSTRACT

Over the past few years, Augmented Reality has become widely popular in the form of smart phone applications, however most smart phone-based AR applications are limited in user interaction and do not support gesture-based direct manipulation of the augmented scene. In this paper, we introduce a new AR interaction methodology, employing users' hands and fingers to interact with the virtual (and possibly physical) objects that appear on the mobile phone screen. The goal of this project was to support different types of interaction (selection, transformation, and fine-grain control of an input value) while keeping the methodology for hand detection as simple as possible to maintain good performance on smart phones. We evaluated our methods in user studies, collecting task performance data and user impressions about this direct way of interacting with augmented scenes through mobile phones.

Author Keywords
Augmented Reality; Hand-based Interaction; Mobile Phone; Gesture Recognition

ACM Classification Keywords
H.5.2 Information Interfaces and Presentation: User Interfaces: Input devices and strategies.

INTRODUCTION

Mobile Augmented Reality (AR) applications have historically been deployed on experimental laboratory systems, using bulky setups of mini-computers, sensors, and large head-mounted displays. These instrumentation constraints posed major obstacles to widespread acceptance of AR technology. With the arrival of embedded AR devices in the form of smart phones, people can now experience AR applications more easily, without setup restrictions, anywhere, and at any time.

*major portions of this work were performed while this author completed her MS degree in Computer Science at UC Santa Barbara

Augmented Reality has been demonstrated widely on mobile phones, with many different applications such as games, navigation and references. In the beginning, many of these applications were less about interaction and focused more on information display on the top of our real world [1]. As AR applications became increasingly interactive, e.g. through gestures on a phone's touch screen, the possible scope of applications was extended. Our motivation stems from the successful demonstration of direct free-hand gestures in AR [2, 3, 8, 13, 19], and our aim is to enable such direct interaction in the "magic lens" of smart-phone based AR.

In this project, we implement hand-based Augmented Reality interaction in real-time on a mobile phone. We design several different interaction types, allowing a user to move and scale an AR object, as well as to control a continuous value (mapped for demonstration purposes to the transparency of a virtual object). We believe that the ensuing direct interaction is very applicable in many smart-phone based AR applications. We evaluate the accuracy and robustness we achieved with our methods through user evaluations. The results indicate that our prototype is functional and engaging. We use a very simple computer vision methodology in order to maximize responsiveness of the interface even on less powerful computational platforms.

We organize this paper into four parts: first, we put our work in perspective of related work, discussing how hand interaction in AR environments has previously been implemented, and comparing our approach. Next, we present the interaction techniques we designed, and explain the implementation of the system. Then we report on several user studies that we conducted, detailing procedure, results, and discussion. Finally, we conclude with an overall discussion and an outlook for future work.

RELATED WORK

Previous work [3, 13, 19] on hand-based interaction between a user and the augmented reality environment has been demonstrated using multiple fiducial markers on both moving and stationary objects. FingARTips by Buchmann et al. [3] implemented hand interaction employing multiple stationary markers on the ground, and others attached to the user's fingers. They utilized ARToolKit [10] to find the relationship between the ground and fingers to detect the gestures. The paper addressed potential occlusion problems by placing markers on three finger joints, resulting in a somewhat encumbered experience. Three gestures (grabbing, dragging, and releasing a virtual object) were distinguished by taking into account pose estimation differences among fingertips as well as finger poses relative to the object and ground (stationary markers). While the method produced promising results in public demonstrations and informal user studies, high setup costs and user encumbrance restricted broad adoption.

Other works, such as [8, 17] partially addressed encumbrance issues due to multi-marker use by attaching color stickers on fingertips. Recently, Hürst et al. [8] implemented various interaction capabilities on mobile phones by employing the system's sensor data, and color markers on fingertips. They showcased single- and dual-finger interactions and conducted user studies to evaluate their techniques on the mobile phone. Finger-based 2D translation tasks were compared against gestures utilizing the phone's touch screen and orientation sensors, and the results showed that in terms of task time, touch-screen interaction outperformed the other approaches. Although the finger-based interaction scored the lowest performance, a high score of fun-engagement level indicated the potential for gaming applications. The authors adopted this feedback, prototyping dual-finger interaction on a virtual board game, including translation, scaling, rotation, and incorporating visualization. Gestures were recognized based on the location, distance, and center point between two fingers. The results showed that translation using two fingers performed comparably to single-finger interaction, but in a post-study survey users rated dual-finger interaction slightly higher. Even though the paper presented the potential of dual-finger interaction on the mobile phone, one shortcoming was the lack of full-fledged 3D interaction. The authors mentioned that full 3D localization was difficult and noisy due to unintended camera motion; thus, they restricted their practical evaluations to 2D interactions. They also applied color stickers that prepared only certain fingers for the interaction. Our work addresses the problem of unintended camera motion, and lets users express their gestures with unmarked fingers of their choice.

Many prototypes implemented on higher-performance systems [5, 15, 16] integrated sophisticated computer vision algorithms (e.g. [21]) to find the hand and detect gestures more robustly in less-restricted scenes, demonstrating that gesture recognition can be rather complex and computationally expensive, depending on the working environment. On the lower-performance hardware afforded by mobile phones, simpler approaches have to be devised to run at practical speed. Baldauf et al. [2] suggested the use of fixed skin-color segmentation and morphological filtering to find fingertips in the image plane. This work focused mainly on the tracking technique, and it has not been demonstrated and evaluated as part of a complete interaction solution featuring different types of interaction and comparative performance evaluations. Seo et al. [20] estimated the palm pose by filtering potential hand regions based on skin color and subsequently applying a distance transformation to find the biggest area. The virtual object appeared on top of this detected region and could move along with the palm. The work however did not address sequential gestures, such as flicking or scaling. Both of these last two projects also suffered from relying solely on skin color when segmenting the hand, because it would be difficult to continuously recognize the hand when the working environment included big portions of skin-color materials such as a wooden desk.

INTERACTION TECHNIQUES

We implemented three different user input gestures to be recognized by our system. We restricted our system to three gestures that are commonly used on touch screen: swiping, scaling, and value adjustment (as by a slider) to experiment with the possibilities of hand interaction techniques on mobile phones both quantitatively and qualitatively. To optimize learnability, we designed our gestures to be similar to existing touch screen gestures.

To demonstrate and test the gestures, we apply transformations to an AR object: a virtual object superimposed on top of an ARToolkitPlus [23] marker. While the world reference frame is defined by the marker, our gestures are all recognized using fast and simple markerless computer vision algorithms.

- Translate (move left, right, up, or down): When the hand moves from outside of the scene toward the marker, the virtual object moves based on the hand direction (pushing the object in a certain direction).

- Scale (increase and decrease size): Similar to a touch screen zoom gesture, when a user makes a pinch or unpinch (spread) motion, the virtual object scales down or up.

- Adjust value: A continuous value is controlled as the hand comes closer to or further away from the marker. We illustrate this by adjusting the transparency of a virtual object: more opaque as the hand comes closer to the marker, and more transparent when the hand gets retracted further. In order to lock in the desired transparency value, the user removes the hand quickly.

IMPLEMENTATION

Without loss of generality, we built our prototype system on top of ARToolKitPlus [23], ported to a Nokia N900 smart phone, running Maemo 5.

After marker detection, we detect and track the hand in the scene using color and motion cues. We assume that a user holds the phone in one hand and uses the other for interaction with the scene. The first approach we pursued aimed to identify frame-to-frame optical flow and classify it by velocities (to distinguish camera motion from motion of the hand in the scene). While Lucas-Kanade optical flow [14], which ranges among the faster computer vision techniques, has good applicability for hand-tracking on laptops or workstations, we soon experienced the computational constraints of our mobile phone platform. Calculating optical flow on each frame, or even just every 5-10 frames, slowed down the overall performance considerably. In our initial implementation, the frame-rate dropped to about 1-5 fps when utilizing this method. Instead, as a fast alternative, we first applied background subtraction and then used simple motion segmentation algorithms to detect the hand gestures.

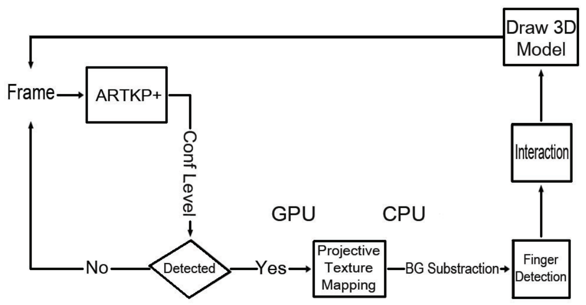

Figure 1. Flow diagram of our application

Background Subtraction

Background subtraction works well when the camera stays in a static position, but in our use case, in which a user holds the phone in one hand, we have to cope with intermittent camera motion. As shown in Figure 1, after the marker is detected, we employ a Projective Texture Mapping [6] step, projecting the reference camera image onto the scene geometry, which in our simple demo application is simply the plane defined by the recognized marker, and capturing a virtual image of that scene from the current camera position. Assuming a close-to-planar scene, we can thus remove the background even in presence of camera motion, and classify all remaining image differences as candidate interaction motion (a second classification step will filter out only skin-colored motion pixels). To improve performance, we compute this inside the GPU in parallel, modifying the framebuffer directly. This method works fast, causing little performance loss (we observed frame rates dropping from 30fps to 27 fps). Once the GPU returns the framebuffer that obtains the projected reference background frame from the current camera viewpoint, it is compared against the current camera frame via subtraction.

The question arises when to capture a new reference background frame. Simply re-capturing the background every n frames is not an option because when a hand is present in the scene, it would be taken as part of the background. Instead, we make use of the fact that this style of interaction (one hand performing a gesture while the other is framing the view) is usually performed while the camera stays close to stationary; We tend to move a camera around to see the virtual object from different angles when we do not interact with the scene. Based on this assumption, the background update will be made when $\Delta x \geq 20.0mm$ or $\Delta y \geq 20.0mm$ or $\Delta z \geq 20.0mm$ where Δx, Δy, and Δz are the absolute camera position differences since the last background frame was taken. Otherwise, when the background update is on hold, our finger detection algorithm is applied to recognize an interaction.

Interaction

After the background subtraction step, the hand can be detected much faster and more accurately than trying to detect it from a raw input frame. There are many algorithms to detect the hand in various environments, e.g. based on color [12, 22], motion flow [14], shape detection [18], and statistical information [9]. Combining multiple hand detection methods will improve the overall accuracy but it will also slow down the overall performance, especially in mobile phone applications. We take a very simple approach, applying color segmentation to detect the hand from the scene. Different color spaces have been suggested for the skin color range; to avoid the cost of color space conversion, RGB color was chosen for this project.

Normalizing RGB is crucial because it will remove any intensity changes and thus, it will address illumination problems and increase accuracy. After normalizing the RGB image, we performed a skin color filter to the region around the AR marker. To save a computational cost and since we are mostly interested in interaction in the region around the virtual object, we restricted the search window to three times the area of the marker (pose-estimation). We used an RGB skin color range as suggested by Kovac et al. [12].

We split the search window into a 4 by 4 grid (Figure 2 (d)). The size of this grid was determined experimentally as a good compromise between speed and necessary accuracy for the gestures we chose to implement and evaluate (see also Section on User Studies). For each grid cell, we compute the percentage of skin-color pixels, $P(skin_n)$, the number of skin-colored pixels divided by total number of pixels in $grid_n$. We implemented two different modes for the interaction: discrete event detection and continuous value adjustment. For the discrete mode, we manually set the threshold value for each cell, that is if $P(skin_n)$ is greater than the threshold value, $grid_n$ is hit. For the continuous value adjustment, we record how much the hand occludes that grid cell. In this case, each cell is not governed by a threshold, but the amount of hand occlusion in each cell, $P(skin_n)$, is used to change a value dynamically, for instance opacity. Figure 2 illustrates the three different interaction techniques: discrete (A and B), and continuous value adjustment (C) mode.

We use the discrete mode when we need to make a sequential gesture, such as for translation or scaling. For instance, scale-down or up uses a pinch motion and we split the detection into two parts: When $P(skin_{n1})$ is greater than the threshold value in a certain starting grid cell and next (within 10 frames from first detection) the corresponding neighbor grid cells obtain $P(skin_{n2})$ that is greater than the threshold value, then we trigger the interaction. On the other hand, we use the continuous mode for an interaction that continuously changes a value, similar to a slider. In our case, the virtual object remains transparent until the hand goes closer to the marker, which increases the opacity. The farther the user retracts the finger, the more transparent the object becomes. To lock in a certain degree of transparency, the user quickly removes the finger.

SPEED

Our application renders the augmented video stream at 640×480 resolution on the phone's display, but we internally use 320×240 resolution for all image processing steps to reduce computation time. Table 1 shows time measurements for each step, revealing that GPU to CPU read-back and image filtering took the most time. Even though the GPU processes

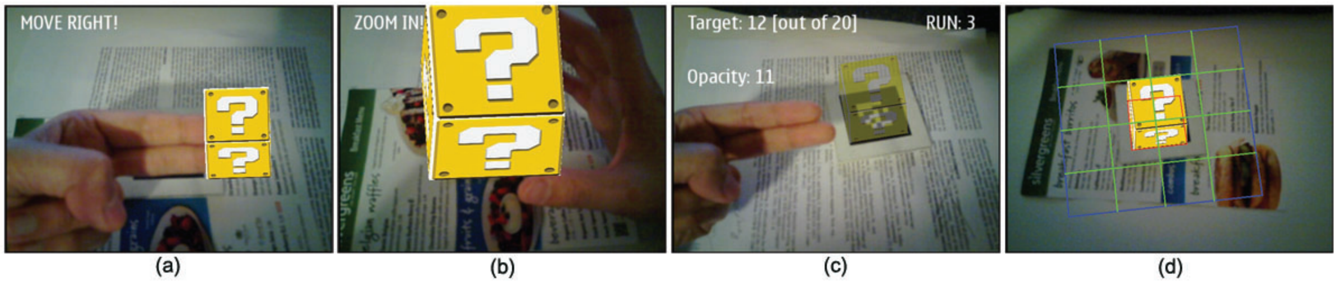

Figure 2. (a) and (b) show translation and scaling interaction. (c) shows continuous value adjustment, applied to box transparency. In this experiment, the user attempts to match a target opacity value. (d) shows (as debug information) the 4×4 grid board we use to track the hand.

Task	Processing Time (msec)
Marker Detection	8.72
Background Subtraction	9.27
Transfer GPU to CPU*	22.60
Image Filtering*	82.63
Skin Region Detection*	12.31
Interaction Detection	0.00

Table 1. Performance timings for each task (in ms)
* tasks are called every 5 frames; all others are called every frame

the projective texture mapping and background subtraction quickly, a challenge arises when we need to access this output from the CPU. We resolved this issue by only requiring a direct pixel access every 5 frames, which experimentation revealed to be sufficient for robust performance. Image filtering is applied after background subtraction, filling any holes that resulted, and smoothing out noise. After prolonged informal experimentation, we determined that applying the erosion and dilation steps three times would result in a sharp reliable segmentation, but this turned out to be a large workload. These expensive tasks (marked with * in Table 1) are thus called only every 5 frames whereas all others are called every frame. This all adds up to a total performance time of approximately 41.5 ms per frame. Since we initially started out with a rendering rate of 30 fps for simple AR marker detection, we now end up with an 18-20 fps overall frame rate, which is still a "reasonable" speed for interactive use.

USER STUDIES

We conducted two different studies to measure the accuracy and robustness of the application, and to observe if the hand interaction using the mobile phone is usable. We conducted both studies back-to-back, the two studies being independent of each other. The total length of the two studies was a maximum of 45 minutes. We recruited 30 participants, who were from different backgrounds, with a mean age of 24.6 years (23 females with a mean age of 24.1 years, and 7 males with mean age 26.4). The test environment was not constrained to a single place, but each test was conducted in a different place, using flat working surfaces which varied in color and texture. Before we started each trial, we determined how familiar a user was with smart phones and Augmented Reality, and established the participant's dominant hand. Participants received standardized instruction and training with our methods, all part of the 45 minutes. We asked users to hold the phone in their non-dominant hand and to use their dominant hand for the interaction.

Goals

The first study measured the time for continuous value control. We speculated that as the range from which to choose the target value becomes wider (e.g. from [1,10] to [1,30]), selecting the target value would become more difficult to control. We were interested in locating any range of drastic performance drop-off to determine the best suitability of our 4×4 grid-board method. A random number was chosen from three different ranges, [1,10], [1,20], or [1,30], and displayed on the screen (Figure 2 (c)). The users' task was to move their hands/fingers in or out the AR scene to control the value to match the displayed number. Every 10 seconds, if users could not finish the task, the system would time out, recorded 10 seconds as the maximum time, and move on to the next random number; otherwise, the completion time was recorded internally and the next trial started. We asked users to do this task 75 times, with 25 trials from [1,10], 25 from [1,20], and 25 from [1,30], in random order.

Our second study tested the performance of hand-held versus tripod setups (Figure 5 (b) depicts a typical tripod scene). One major point of interest with our proposed method was the intended system use which differed slightly from related work: users hold the phone in one hand and use the other for the interaction while enabling 6-degree-of-freedom interaction. We hypothesized that since our method addressed the issue of (slight) camera motion, we would get about the same results from non-stationary (hand-held) camera setups as with stationary (tripod) camera setups. We randomized the order of conditions. For hand-held setups, we asked users to hold the phone any which way they preferred and also let them switch hands if they deemed it more natural. We had four different gestures: Move a box left, right, up, and down in random order, 25 times for each direction. For each task, we displayed the requested direction on the screen and we asked users to complete the trial in 3 seconds. We simply recorded whether users completed the task within that time frame or not. We asked users to do 100 trials, 50 with the camera on a tripod and 50 with the camera in hand.

Results

Our pre-study survey showed that 83% of the participants own smart-phones, and that among these participants the predominant average usage of smart phone was between every

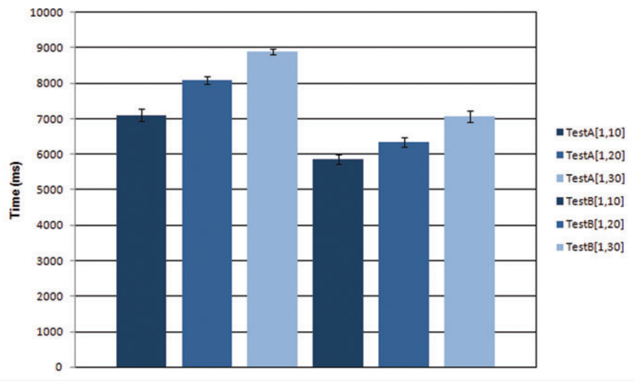

Figure 3. Results of Continuous Value Experiment based on *TestA:* whole data, and *TestB:* successfully completed trials only.

hour to every few hours. 27% of the participant knew or had heard of Augmented Reality before, generally from the web or through applications such as games, but they reported their familiarity level with AR as low (average of 1.3, with median of 1, on a 1-5 scale).

Task	Completed	Full data		Completed data	
		Mean	SD	Mean	SD
[1,10]	69.2%	7102.2	946.5	5866.8	718.6
[1,20]	53.6%	8076.7	590.0	6338.7	776.6
[1,30]	39.3%	8886.8	410.4	7068.9	837.9

Table 2. Results of continuous value adjustment

Table 2 reports the results of our first (continuous value adjustment) test, with the collected dataset sliced two ways: The full dataset represents all 2250 runs (25 runs $\times$ 3 ranges $\times$ 30 participants) we collected during the study. The completed dataset on the other hand represents only those tasks that participants were able to finish within 10 seconds. Table 2 lists the completion rate of each range (after removing three outlier participants, who did not follow instructions correctly, the probabilities improved to 75.3%, 57.1% and 42.4%).

For each dataset, we measured the average time (ms) to complete the number for each of the three different ranges. An ANOVA for the full dataset revealed significant differences ($F(2, 87) = 51.4$, and p = $1.8 \times 10^{-15} < 0.05$). Moreover, post-hoc analysis using Tukey's HSD test demonstrated that since all possible differences were greater than HSD, all groups are significantly different from each other. An ANOVA for the completed dataset also revealed significant differences ($F(2, 87) = 18.1$ and p = $2.63 \times 10^{-7} < 0.05$). Here, a post-hoc analysis using Tukey's HSD test showed that HSD = 479.4, with $|M_1 - M_2| = 471.9$, $|M_2 - M_3| = 730.2$, and $|M_3 - M_1| = 1202.1$, and thus no significant difference between the [1,10] and [1,20] groups, but differences among the other groups ([1,20], [1,30] and [1,10], [1,30]). Figure 3 illustrates these results.

We also tested the collected data for a learning effect, i.e. do later trials tend to exhibit increased performance? Over all participants, the average time in ms for the first 25 runs was 8008.9 (SD=676.3); for the next 25 runs, it was 8096.8 (SD=649.7); and for the last 25 runs, it was 7997.0 (SD=830.1). In this case, an ANOVA test revealed no significant differences ($F(2, 87) = 0.17$ and p = $0.84 > 0.05$). Similarly, a t-test between the first 25 runs and last 25 runs showed $t(29) = 0.081$ and p = $0.94 > 0.05$, implying that there was no significant difference. We did not observe a learning effect over time with this task.

Our results confirmed that the value adjustment task became more difficult with the interval to pick the target value from widening. We did not observe a drastic timing change between any two of the three different ranges. Instead, the difficulty level appears to rise nearly linearly from range [1,10] to range [1,30]. The average time for range [1,10] was 7102.2ms (5866.8ms for completed trials), which appears to be surprisingly long for the gesture, and the completion rate was 69.2%, which is fairly low. These quantitative results support that it is difficult to make fine-grin adjustments using our 4×4 grid approach, and this puts a limit on the range of hand gestures we can gainfully implement. As a result, we need to re-consider the use of the 4×4 region for gestures demanding high precision.

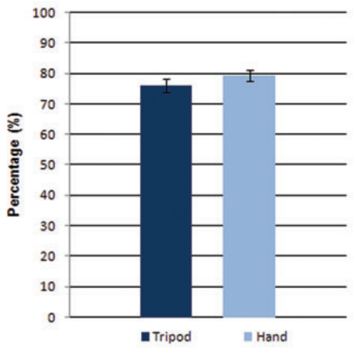

Figure 4. Completion rates for Experiment 2 (movement task) for camera on tripod (*left*) and camera in hand (*right*)

The second experiment measured the success rates of triggering discrete translation events for camera-in-hand versus camera-on-tripod setups (Figure 5). The results displayed in Figure 4 show that the average rate of completing the task within 3 seconds with the camera on a tripod was 0.76 (SD=0.13) and for camera-in-hand it was 0.79 (SD=0.10). A t-test ($t(29) = -2.08$ and p =$0.047 < 0.05$) revealed that there was a significant difference between the tripod and hand setups, with the camera-in-hand setup performing better. From the data, we also checked if there were any ordering effects, i.e. did it matter if a user first did the tripod-based test or the camera-in-hand test? T-tests revealed no significant difference from what modality was performed first.

Our analysis contradicted our initial hypothesis that the two setups would perform equally well. We obtained the unexpected result that our camera-in-hand setup performed slightly better than the tripod setup. It is difficult to find exact quantitative reasons of why the camera-in-hand mode might have outperformed the tripod mode. After all, the differences

	Experiment 1		Experiment 2		
	Fun to use	Easy to control	Easy to control	Move physically	Tripod-Hand
Strongly disagree (1)	0%	6.7%	0%	0%	16.7%
Disagree (2)	0%	26.7%	0%	0%	13.3%
Somewhat disagree (3)	0%	33.3%	6.7%	3.3%	6.7%
Neutral (4)	3.3%	23.3%	3.3%	10%	6.7%
Somewhat agree (5)	26.7%	10%	20%	26.7%	3.3%
Agree (6)	26.7%	0%	30%	13.3%	13.3%
Strongly agree (7)	43.3%	0%	40%	46.7%	40%
Mean (Likert scale)	6.1	3.0	5.9	5.9	4.6

Table 3. Post-study questionnaire results using a 7-point Likert scale; data from 30 participants. For the last column (Tripod-Hand), 1 indicated an absolute preference for the tripod setup and 7 for the camera-in-hand setup

were not huge. What can be concluded is that our method of dealing with slight camera movements was successful enough as not to lead to performance handicaps in this study. Post-study survey comments indicated an alignment of user performance with user preferences: Some users who preferred the tripod setup scored high on the tripod due to "easiness" because they liked having "both hands free" and thus, they did not have to "worry about switching hands for the interaction", such as between move gestures to the left and to the right. On the other hand, some users preferred the camera-in-hand, because for the tripod setup it was "harder" to see the virtual object "from a certain angle", and "tripod legs were an obstacle for the interaction".

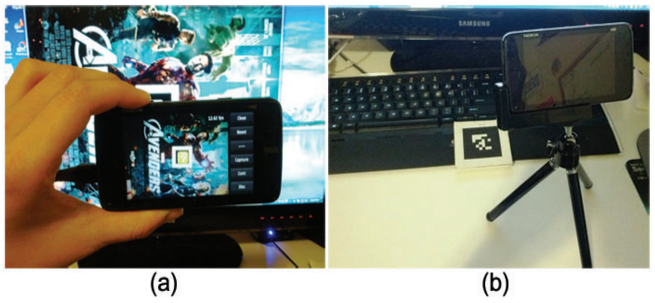

Figure 5. Two different setups: (a) camera-in-hand and (b) camera-on-tripod

Discussion

After the first test, we asked users about how fun the application was and on a scale from 1 (not fun) to 7 (very fun), our average score was 6.1 (median = 6). In exit interviews participants mentioned that it was "like a game application" and that they derived a "must-win" feeling during the study. We also think that the low familiarity level with AR (1.3 according to our pre-study survey) might have been a contributing factor, in that experiencing a method that users were not familiar with was "entertaining" to them. However, the score for the easiness of locking in the correct target number in experiment 1, from 1 (very hard) to 7 (very easy) scale, was only 3.0 (median = 3). As our results on completion rate also demonstrated (Table 2 shows 69.2%, 53.6%, and 39.3% for the different ranges), it was possible that frustration with fine-grain adjustments brought down this score. Table 3 shows more detail about two experiments' post-survey result.

On the second test, the average score on easiness of control in 1 (hard) to 7 (easy) scale was 5.9 (median 6). The discreteness of this task contributed to better completion rates. We also asked if users felt like they were moving the box physically, and on a scale from 1 (not really) to 7 (absolutely) scale, the result was 5.9 (median 6). This is a high value, considering that there was no force feedback. A shortcoming of the experience was that the box always moved with the same velocity in only four directions.

Despite many users' positive feedbacks about our proposed method, we also received some complaints, such as "tiring shoulder". Particularly, some critical comments came in response to the first test, when users had to hold the phone continuously for each task that took up to 10 seconds. This might mean that gestures that requires longer interaction times could face acceptance problems. For instance, one of the possible solutions that Hürst et al. [8] suggested was to focus on applications that requires gestures less frequently.

A question arises if direct hand interaction is a better solution compared with current state-of-the-art touch-screen gestures. As the results from our first experiment indicate, based on the completion rate, time, and post-survey feedback, controlling a small value using our proposed method was not easy. However, interacting with a 3D AR object and environment through gestures on a 2D screen observing the 3D scene also has its potential problems. The fixed and potentially very small size of a smart-phone display screen doesn't lend itself well to larger-motion gestures, especially when the distance between the camera and the AR object increases, making the interaction footprint on the display very small (through perspective projection). Further experiments are needed to make quantitative comparisons for various scenarios. Although touch screens offer a very reliable and thoroughly established mode of interaction, for controlling 3D AR scenes the jury is still out on what interface metaphor might work best.

Quantitative results from our second experiment revealed a success rate of 79% for users being able to complete a flicking/pushing task within 3 seconds. We conducted a brief analysis, applying keystroke-level model (KLM-GOMS) [4, 7, 11] theory to predict the time of a related swiping gestures on the touch screen, so that we can informally compare it against our method. We split our discrete method into two parts: the mental act, through which users cognitively prepare for the task that they need to achieve (flicking in a certain direction), and the pointing event, for which users move a finger around the touch screen to achieve the goal (just like moving a mouse cursor). We predict the time of using the discrete method on the touch screen to amount as approximately 2.3 seconds. This time can vary depending on the distance between the screen and finger and other factors. It is likely that touch-screen input will yield higher success rates and altogether faster response times than our through-the-camera-phone hand gestures. However, the benefits of direct-manipulation for AR object manipulation is a potential plus for the proposed method, as evidenced by the positive post-study survey feedback and specifically the question on the realism of interaction.

Conclusions

In this paper, we introduced a simple yet novel approach for interacting with virtual content in AR environments observed through camera smart phones, using our unencumbered hands for gestural input. Our results demonstrate that we can successfully address the problem of potential camera motion during gesture detection using projective texture mapping and background subtraction, obtaining acceptable correctness rates for simple scenes while maintaining good frame rates.

Our first evaluation experiment showed that this type of hand interaction might not (yet) be a good fit for subtle adjustment motions, as evidenced by users' difficulties in controlling small interval values; we think that button, keypad or other physical sources of input might still be a better fit in this case. Our second study demonstrated acceptable performance (correctness rate) of using a discrete sequential gesture detection for a translation event, at a reasonable speed. Our post-study survey comments highlight some potential promise of direct-manipulation gesture interaction with AR scenes (high engagement and physical realism). The hand-based interaction was perceived as "fun" and "easy to control" (for the discrete method). There is a distinct difference in features and affordances between a touch screen and our proposed method. A touch-screen interface is a common and accepted gesture tool in mobile phones, which users are thoroughly familiar with and which works robustly and, in conjunction with the right interface elements, very advantageously. We are not arguing or even targeting a complete replacement of this interaction modality. We do think, however, that our work, as well as other recent work in the area of hand gesture recognition highlights specific potential of 3D interaction using finger gestures to directly manipulate an AR scene. We have demonstrated the feasibility and potential uses of such technologies using camera smart phones. Based on reasonable accuracy, robustness, and positive feedbacks we received, we conclude that direct hand interaction in phone-based AR is a modality that should be considered to complement others.

We realize that improvements are desirable in terms of long-term-use ergonomics, robustness and reliability. We will consider the design of infrequent gestures as Hürst et al. suggested, and we also plan to collect more data on user behaviors to optimize gesture design. Our current 4×4 grid restricts possible gestures, as detailed adjustments are difficult to achieve. Future work will consist in tracking individual fingers more efficiently and robustly, for instance by applying multiple hand detection algorithms in parallel while keeping performance high. We would like to expand our method to more and more fine-grained gestures, such as freely rotating virtual objects or moving objects in any direction with different velocities. User studies will be required to investigate if 3D interaction 'in the air' has advantages over existing and more established display-based interaction methods, both qualitatively and quantitatively. Once we make progress towards these goals, one of the possible applications in which we can utilize and showcase our new direct manipulation gestures would be a virtual storybook, in which users can interact with characters and scenery through life-like 3D gestures, hopefully enhancing the Flow and Presence afforded by such an environment.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their helpful comments and suggestions. This work was partially supported by NSF CAREER grant IIS-0747520.

REFERENCES

1. Azuma, R., Baillot, Y., Behringer, R., Feiner, S., Julier, S., and MacIntyre, B. Recent Advances in Augmented Reality. *IEEE Computer Graphics and Applications 21*, 6 (Nov. 2001), 34–47.

2. Baldauf, M., Zambanini, S., Fröhlich, P., and Reichl, P. Markerless Visual Fingertip Detection for Natural Mobile Device Interaction. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*, MobileHCI '11, ACM (New York, NY, USA, 2011), 539–544.

3. Buchmann, V., Violich, S., Billinghurst, M., and Cockburn, A. FingARtips: Gesture Based Direct Manipulation in Augmented Reality. In *Proceedings of the 2nd International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia*, GRAPHITE '04, ACM (New York, NY, USA, 2004), 212–221.

4. Card, S. K., Moran, T. P., and Newell, A. The Keystroke-Level Model for User Performance Time with Interactive Systems. *Communications of the ACM 23*, 7 (July 1980), 396–410.

5. Chan, L.-W., Kao, H.-S., Chen, M. Y., Lee, M.-S., Hsu, J., and Hung, Y.-P. Touching the Void: Direct-Touch Interaction for Intangible Displays. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, ACM (New York, NY, USA, 2010), 2625–2634.

6. Everitt, C. Projective Texture Mapping. Tech. rep., NVDIA, 2000. `https://developer.nvidia.com/content/projective-texture-mapping`.

7. Holleis, P., Otto, F., Hussmann, H., and Schmidt, A. Keystroke-Level Model for Advanced Mobile Phone

Interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '07, ACM (New York, NY, USA, 2007), 1505–1514.

8. Hürst, W., and van Wezel, C. Gesture-based Interaction via Finger Tracking for Mobile Augmented Reality. *Multimedia Tools and Applications* (2012), 1–26. 10.1007/s11042-011-0983-y.
9. Jones, M. J., and Rehg, J. M. Statistical Color Models with Application to Skin Detection. *International Journal of Computer Vision 46*, 1 (Jan. 2002), 81–96.
10. Kato, H., and Billinghurst, M. ARToolKit, 2003. `http://www.hitl.washington.edu/artoolkit/`.
11. Kieras, D. Using the Keystroke-Level Model to Estimate Execution Times. Unpublished Report, `http://www.pitt.edu/~cmlewis/KSM.pdf`, 1993.
12. Kovac, J., Peer, P., and Solina, F. Human Skin Color Clustering for Face Detection. In *EUROCON 2003. Computer as a Tool. The IEEE Region 8*, vol. 2 (Sept. 2003), 144 – 148 vol.2.
13. Looser, J. *AR Magic Lenses: Addressing the Challenge of Focus and Context in Augmented Reality*. PhD thesis, University of Canterbury, New Zealand, 2007.
14. Lucas, B. D., and Kanade, T. An Iterative Image Registration Technique with an Application to Stereo Vision. In *Proceedings of the 7th international joint conference on Artificial intelligence - Volume 2*, IJCAI'81, Morgan Kaufmann Publishers Inc. (San Francisco, CA, USA, 1981), 674–679.
15. Malik, S., Ranjan, A., and Balakrishnan, R. Interacting with Large Displays from a Distance with Vision-Tracked Multi-finger Gestural Input. In *Proceedings of the 18th annual ACM symposium on User Interface Software and Technology*, UIST '05, ACM (New York, NY, USA, 2005), 43–52.
16. McDonald, C., Malik, S., and Roth, G. Hand-Based Interaction in Augmented Reality. In *Haptic Virtual Environments and Their Applications, IEEE International Workshop 2002 HAVE* (2002), 55–59.
17. Mistry, P., Maes, P., and Chang, L. WUW - Wear Ur World: A Wearable Gestural Interface. In *CHI '09 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '09, ACM (New York, NY, USA, 2009), 4111–4116.
18. Potamias, M., and Athitsos, V. Nearest Neighbor Search Methods for Handshape Recognition. In *Proceedings of the 1st international conference on Pervasive Technologies Related to Assistive Environments*, PETRA '08, ACM (New York, NY, USA, 2008), 30:1–30:8.
19. Reitmayr, G., and Schmalstieg, D. Mobile Collaborative Augmented Reality. In *Proceedings of IEEE and ACM International Symposium on Augmented Reality*, IEEE Computer Society (2001), 114–123.
20. Seo, B.-K., Choi, J., Han, J.-H., Park, H., and Park, J.-I. One-Handed Interaction with Augmented Virtual Objects on Mobile Devices. In *Proceedings of The 7th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry*, VRCAI '08, ACM (New York, NY, USA, 2008), 8:1–8:6.
21. Vatavu, R. D., Pentiuc, Ş., Chaillou, C., Grisoni, L., and Degrande, S. Visual Recognition of Hand Postures for Interacting with Virtual Environments. *Advances in Electrical and Computer Engineering 6*, 13 (2006), 55–58.
22. Vezhnevets, V., Sazonov, V., and Andreeva, A. A survey on Pixel-Based Skin Color Detection Techniques. In *Proceedings of the 13th International Conference on the Computer Graphics and Vision* (Sept. 2003), 85–92.
23. Wagner, D., and Schmalstieg, D. ARToolKitPlus for Pose Tracking on Mobile Devices. In *Proceedings of the 12th Computer Vision Winter Workshop*, CVWW '07 (2007), 139–146.