Experiments in 3D Interaction for Mobile Phone AR

Anders Henrysson* — VITA — Linköping University

Joe Marshall† — Mixed Reality Lab — University of Nottingham

Mark Billinghurst‡ — HIT Lab NZ — University of Canterbury

Abstract

In this paper we present an evaluation of several different techniques for virtual object positioning and rotation on a mobile phone. We compare gesture input captured by the phone's front camera, to tangible input, keypad interaction and phone tilting in increasingly complex positioning and rotation tasks in an AR context. Usability experiments found that tangible input techniques are best for translation tasks, while keypad input is best for rotation tasks. Implications for the design of mobile phone 3D interfaces are presented as well as directions for future research.

CR Categories: H.5.1 [Information Interfaces and Presentation]: Multimedia Information Systems—Artificial, augmented, and virtual realities

Keywords: Mobile Graphics, Augmented Reality, 3D interaction

1 Introduction

Mobile phones have become increasingly capable of interactive 3D graphics. Featuring a CPU, full color display and in recent models; a GPU, a mobile phone is conceptually similar to a PC. This has lead to rendering algorithms and APIs developed for graphics workstations to be ported, though modified to compensate memory and CPU limitations. 3D application development for mobile platforms is thus fairly similar to development for stationary ones. While rendering algorithms have migrated, most interaction techniques developed for stationary computers are not applicable for mobile devices. For example, with mobile phones at least one of the users hands are busy; there is reduced keyboard input capability and the screen size is very small, limiting input and output options. Using external input devices such as a computer mouse is not an option; hence, we need to look at other input techniques.

In this paper we focus on the use of the keypad and phone camera for 3D interaction. The main point of difference from previous work is that for the first time we consider how a front facing camera on the phone can be used for 3D AR interaction. We describe several interaction methods using the front camera on the phone and also provide rigorous user study results of how effective these methods are. This work is important because it provides usability results and design guidelines that will help others develop more effective mobile AR and VR interfaces.

In the remainder of this paper we first review related work in the field and describe lessons learned from previous user studies. Next we give an overview of the interaction techniques we have implemented. Sections 4 and 5 present formal user studies comparing these techniques in translation and rotation tasks. Section 6 contains a discussion of these results and the design guidelines that can be drawn from them. Next we describe a sample application we have developed based on using camera input. Finally we present some conclusions and future directions.

2 Related Work

There has been a long history of evaluating different interaction techniques for desktop 3D graphics and immersive VR applications. For example, Beaton et al. [1987] present an early example of 3D positioning and orienting tasks on a desktop 3D user interface, Hinckley et al. [1997] compare the usability of mouse and six degree of freedom input devices for 3D graphics rotation, while [Mine 1995] describes several interaction techniques and usability study results for immersive VR environments. Hand has produced an extensive summary of the literature for 3D interaction in desktop and VR applications and usability study results [1997]. These papers are useful because they provide examples of well designed usability studies for object interaction, particularly translation and rotation tasks.

Interaction studies have also been conducted with AR interfaces, although there are not many that explore object manipulation. For example, Ellis [1997] conducted an experiment to explore user's ability move a virtual ring over a virtual wire in an AR display with different rendering latencies. More relevant is the work of Wither and Höllerer [2004] which describes a user-study in an outdoor wearable AR interface that evaluates four techniques for controlling a distant 3D cursor and annotating real objects. There is a need for further AR manipulation studies, especially because AR interfaces are different enough from VR interfaces that usability results from a desktop or immersive VR experiment may not apply.

Although AR interfaces have migrated to mobile phones there has been little research on interaction techniques for mobile phone AR, and almost no formal usability studies have been conducted. Few AR mobile phone applications support more than simple object selection and manipulation. One of the only handheld AR interaction studies is an earlier experiment [Henrysson et al. 2005b] which compared techniques for 3D virtual object translation and rotation. The task involved moving or rotating a selected object to align with a wireframe target object. The techniques compared included keypad input and tangible phone input. The tangible input condition was where the virtual object was at a fixed position relative to the phone and so moved when the user moved the phone. In the translation experiment they found that people were able to move objects more quickly in the tangible input condition than with keypad input. However for a 3D object rotation task the tangible input technique was significantly slower than using keypad input, partly due to the difficulty of constraining rotation about a specific axis with the tangible technique. This suggests that tangible input is effective for translation tasks but not for rotation.

In our current work we wanted to build on this and explore the new AR interaction possibilities offered by the forward facing cameras mounted on the front of the mobile phone. This paper is the first

*e-mail:andhe@itn.liu.se
†e-mail:jqm@cs.nott.ac.uk
‡e-mail:mark.billinghurst@canterbury.ac.nz

that describes the use of the front camera of a mobile phone for AR interaction, and it is unique in that it presents comparative results from a formal user study where two of the conditions are based on the forward facing camera.

3 Mobile Interaction Techniques

Mobile phones are interesting for AR applications because unlike many other AR interfaces the display and the input hardware are connected together. So a user could provide keypad or computer vision based input while viewing the output on the phone screen. In our research we wanted to consider four input options:

- Keypad input
- Tangible input
- Phone Tilting detected through computer vision
- Gesture input from a camera on the front of the phone

Before describing the various implementation techniques we will give an overview of our AR platform. It is based on an earlier custom port of the ARToolKit computer vision tracking library to the Symbian operating system [Henrysson et al. 2005a], which is able to run on current Symbian based mobile phones at up to 10 frames per second. Creating the ARToolKit port involved building an optimized fixed point library. The performance was further enhanced by adding frame-to-frame coherency to reduce computation load. Filtering tracking data using double exponential smoothing (DESP) reduced jitter inherent to ARToolKit. As a 3D graphics API we use OpenGL ES, which is a reduced subset of OpenGL, suitable for low-power, handheld devices.

The phone used in this paper is the Nokia 6680 (see Figure 1) which has a 220 Mhz processor and runs a software implementation of OpenGL ES. It has two cameras: a main camera on the back, and a second forward facing front camera. The screen size is 178 x 208 pixels and in our application we used a video capture resolution of 160x120 pixels.

3.1 Keypad Input

In the keypad/joypad method virtual objects continuously rotate or translate a fixed amount for each fraction of a second while the buttons are pressed. For each degree of freedom (DOF) we use two buttons to increment or decrement the transformation. The translation speed is 4 mm/frame yielding a speed of about 30 mm per second given the current framerate. The speed of rotation is 4 degrees per update i.e. around 30 degrees per second. Figure 1 shows how the keypad input is used to move or translate the virtual object along or around the x,y and z axes.

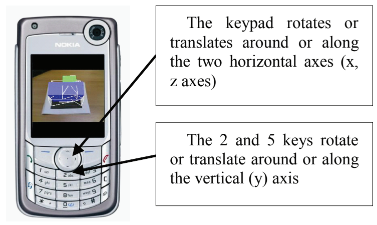
The keypad rotates or translates around or along the two horizontal axes (x, z axes)
The 2 and 5 keys rotate or translate around or along the vertical (y) axis

Figure 1: *Nokia 6680 and Keypad Input Mappings*

3.2 Tangible Input

In the tangible input case a 3D object is selected by positioning virtual cross hairs over it and clicking and holding down the joypad controller. While selected, the virtual model is fixed in space relative to the phone and so is translated at the same time as the phone is moved. Objects are deselected by releasing the joypad.

3.3 Phone Tilting Input

The third input technique explored is using the tilt of the phone to control virtual object rotation. Phone tilting input uses computer vision techniques to detect the tilt of the phone around two axes, which run across the x direction of the screen and the y direction of the screen, parallel to the front of the phone. This is done using images grabbed from the back camera of the phone and a global block matching technique similar to that used in the TinyMotion [Wang and Canny 2006] computer vision phone tilting code.

Our improved version uses full colour block matching, which is significantly more robust, and also uses a post-processing stage to allow single pixel tilting movements to be detected (in contrast to the large block movements detected by TinyMotion). This gives a smoother input at a rate of approximately 15 frames per second (limited by the frame rate of the camera). The phone input method is based on naturally occurring features captured by the phone camera meaning that it will work with any camera input provided it has enough visual texture. When we want to rotate the virtual object a key is hit on the keypad and the ARToolKit tracking is stopped and the phone tilting input started. This means that the phone tilt can be detected even when the marker is not visible.

3.4 Gesture Input with Front Camera

While the user is viewing graphics content on the phone screen it is possible to use the front camera to capture gesture input and use this to interact with the virtual content. We have experimented with both 2D gesture input using motion flow tracking, and 3D gesture input using ARToolKit tracking. In this section we describe these interaction techniques in more detail.

3.4.1 2D Gesture Input

This input method tracks a finger held in front of the phone, by using a simple frame-differencing tracking method. This relies on the fingertip creating a large difference between frames when it is moved in front of the front camera.

Two key assumptions are made in this tracking method. Firstly that the background will be relatively stable, or only change globally (such as when a light is turned on or off). Secondly, that the user will use only their dominant hand to control the interface. In our test applications, we allow the user to choose left or right handed use of the interface.

With these assumptions, it is generally known what range of angles the finger is going to be entering the field of view of the front camera from. Thresholds are used to detect when the scene is static, or changing globally, both of which do not create input. When movement is detected, the knowledge that the finger will be coming from the left or right of the view (depending on handedness) is used in order to calculate which parts of the frame difference are created by the fingertip.

The finger tracking method is robust in most situations, including when the camera is also pointing at the user's face. It may fail if used with an extremely active background, such as if used when on a train with the camera pointing out of the window. This data as to the position of the fingertip, is used in conjunction with a ”clutch” button on the phone, to give the application 2 dimensions of movement information with a resolution of approximately 100x100 pixels.

When the 2D tracking is activated by pushing a key on the keypad, the main camera video is frozen and a second smaller video window is activated to show the front camera view and give the user feedback about the tracking performance. A small red dot is shown over the user’s finger as it is being tracked (see figure 2).

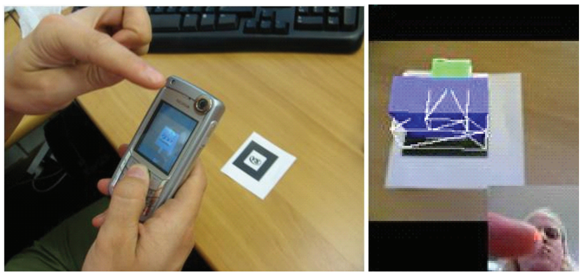

Figure 2: *Front Camera Gesture Input with 2D Finger Tracking*

3.4.2 3D Gesture Input

When interacting with 3D data it is desirable to have a 6 DOF interaction device, since this allows an optimal trajectory in rotation and translation space. The frame-difference approach described above is limited to two-dimensional input and requires the user to switch modality to access the third rotation axis.

In order to provide higher input dimensionality we use ARToolKit tracking of a small marker attached to the user’s fingertip (see figure 3). With marker tracking we can easily emulate markerless 3D fingertip tracking, currently unavailable on mobile phones. While it is possible to switch cameras between each frame, it is too slow for interaction and the cameras need a few consecutive frames to set the white balance. Instead the user switches cameras with the push of a button as in the frame-difference approach.

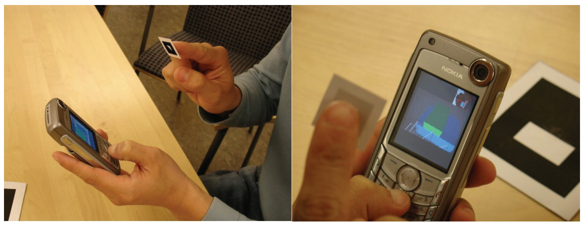

Figure 3: *Front Camera Gesture Input with ARToolKit Marker*

The front camera has a limited image quality and thus sets a lower limit for how small a marker can be, given the intended motion range of the finger. We found that a marker size of 15 mm was required for the tracking to work. This means that the marker will be visible from behind and thus it will obscure more than the fingertip alone. As in the frame-based approach the user is provided a small video image of the front camera view superimposed over the main camera view. When the marker is successfully tracked it is overlaid with a blue cube in the front camera view. This is so the user can use peripheral vision to confirm successful tracking while focusing on the main interaction task.

When used for translation, the tracking starts directly when the user switches to using the front camera. In this interaction mode there is a one to one mapping between the motion of the finger and the motion of the manipulated object. Thus the field of view of the front camera is a limiting factor. Greater motion can be achieved by using a clutching motion and temporarily switching back to view mode. Only the motion of the finger relative to the phone is recorded, not the motion of the phone relative to the marker defining the global coordinates. If the phone is moved significantly during interaction, potential confusing discontinuities will occur when returning to viewing mode.

We also implemented an ArcBall rotation technique where position and motion of the finger were mapped to a virtual sphere enclosing the object to be rotated. This is a preferred technique in desktop 3D-applications using a 2D mouse as input device. However it turned out to be far too complex on the phone. It was dismissed as unsuitable after informal testing and the 2D gesture input method used for object rotation instead.

In order to evaluate how effective these various interaction techniques could be in mobile phone AR interaction tasks, we conducted a formal user study of 3D input techniques across the various interaction methods. The user evaluation was conducted in two separate experiments to enable us to separately consider how well the interaction techniques work for translation and rotation tasks.

4 Experiment One: Translation

There were 12 subjects that took part in the experiment (10 men, 2 women, aged 19 to 40). They had all tried mobile AR demonstration software before and some of them had taken part in a manipulation user study two years before. Figure 4 shows a subject taking part in the user study.

4.1 Experimaental Task

To test the interaction techniques described in the previous section we conducted a study in which users tried to position virtual blocks in an AR interface. The subject sat at a table in front of a piece of paper with a number of ARToolKit tracking markers printed on it. They were given a Nokia 6680 mobile phone running the AR test software. When the user looked through the phone display at the ARToolKit marker they saw a virtual ground plane with a virtual block on it and a wireframe image of a second same sized target block which was translated in the x, y and z direction relative to the solid block (see figure 4). The goal of the task was to move the solid virtual block until it was positioned entirely inside the wireframe block. At each frame update, an error vector was calculated by taking the offset between the current block position and the target position. The block is regarded to have been placed correctly if the length of the error vector is less than 8 mm. When this occurs the virtual block will change colour showing the task is complete. The user wasn’t allowed to move the tracking marker. The translation task used the following input methods:

A: Keypad input

B: Gesture input using the front camera with an ARToolKit marker

C: Virtual object attached to phone

For each user the order of the conditions was counterbalanced to ensure that there was no learning effect. For each condition the subject was allowed to practice with the input technique until they felt comfortable with it. In each condition the subjects had to perform five translation tasks with target objects being placed at different locations. There were two tasks which could be accomplished by moving the block in only one direction (the one degree of freedom task). One task could be completed by moving the virtual block in

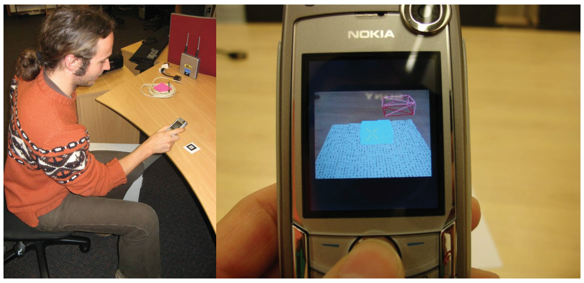

Figure 4: *Experiment setup. The Virtual Translation Task*

two directions (the two degree of freedom) and the remaining two tasks required the user to move the object in all three directions. This was so that we could study how the dimensionality of the task affected the user performance.

For each of the trials we measured the time it took the user to complete the task. After finishing each condition, users completed a subjective survey with questions about how intuitive the interface was to use, how enjoyable it was, and how accurately they felt they could move the virtual block. After all three conditions are completed, the user was give a further subjective survey which asked them to compare between conditions and rank them in order of how intuitive the interface was etc.

The experimental task was based on the task used for the earlier manipulation studies [Henrysson et al. 2005b], but in this case we have added a gesture input condition and also compare the effect increasing the complexity of the task has on the results, by increasing the number of directions that the blocks must be moved along.

In this experiment we would expect that as the user has to move the block over more directions (increasing DOF) then it should take longer in the keypad condition. However, with both the tangible input and the finger gesture input the user can seamlessly move the virtual object in more than one direction at once, and so we should find little difference in performance time between tasks. The gesture input condition has a smaller range of motion possible than the tangible input case, and so should be expected to take more time, due to the user's finger moving out of the front camera field of view and having to be moved back into the camera view.

So in terms of performance we can predict that the keypad condition will perform worse as the task complexity increases, that the gesture and tangible input will not vary much between tasks as the complexity changes, and that the tangible condition will be faster on average than the gesture condition. It is difficult to predict which conditions the user will prefer, but the gesture and tangible conditions will have more novelty value compared to the keypad input and so may be rated more enjoyable.

4.2 Results

There was a significant difference between task performance time both across the three input conditions and for the keypad condition, across tasks. Figure 5 shows the average time to complete the tasks for each condition and across each task as they become more complete. There are several things obvious from the graph. As predicted, in the keypad input condition as the task required motion in more directions it took longer. In both the gesture input and tangible input case there is little difference in task time (within the standard error) as the tasks got more complex. The tangible input case performed better than the other conditions apart from the keypad input condition in the simple one degree of freedom case when it was just as good.

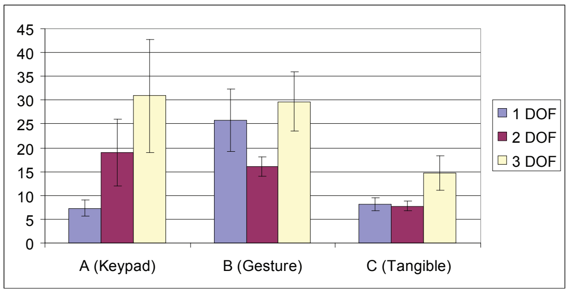

Figure 5: *Average Time (s) Per Task vs Degrees of Freedom*

Looking at the time taken against degrees of freedom, a two factor ANOVA (condition, DOF) found a very significant effect of both interface type (F(2,33) P=0.01) and degree of freedom (F(2,33), P=0.028), however it did not show an interaction between the two factors (F(2,2,99), P = 0.3). Analysing the 3 degree of freedom results alone, showed a significant difference between the interface types, (F(2,33), P=0.03).

After each condition the users were asked the following four subjective questions:

1. How easy was it for you to move the block?
2. How accurately could you move the block to?
3. How quickly could you move the block?
4. How enjoyable was it to use the application?

Each of these questions were answered on a Likert scale of 1 to 7, where 1 = not very easy, 7 = very easy, etc.

Figure 6 shows a graph of the subjective survey results across the three conditions.

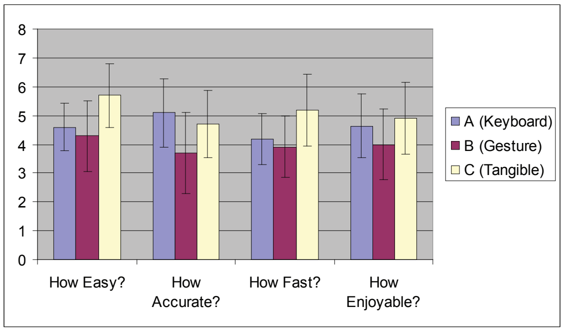

Figure 6: *Subjective Ratings for Each Interface*

As can be seen, due to the large variance in average values there is little difference between the conditions. Analysing the subjective results using a single factor ANOVA we found no significant difference in responses to any of the four survey questions, although the results for question 1, (How easy it was to move the block?), was tending towards significance (F(2,30) = 3.03, P = 0.06). This implies that it would be useful to run the experiment again with a larger subject pool.

After completing all of the tasks, subjects were also asked to rank the three conditions in order according to the following criteria:

1. How easy was it for you to move the block?
2. How accurately could you move the block to?
3. How quickly could you move the block?
4. How enjoyable was it to use the application?

Conditions were ranked in order from highest (1) to lowest (3). Figure 7 shows the average ranking values across conditions. Using a Friedman Test to compare between the average ranking scores across conditions we found no significant differences between the user's rankings in response to each question.

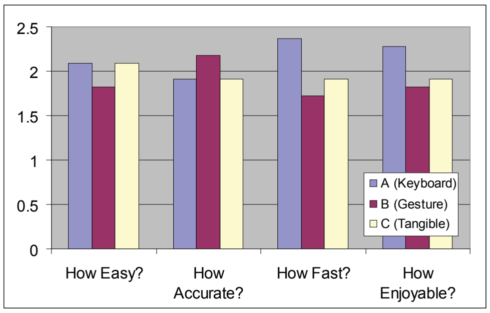

Figure 7: *Mean rankings for each interface*

Interestingly enough, condition C, the Tangible Input condition, was almost always ranked second by the subject with the other conditions split between first and third. Table 1 shows the average ranking results for each question and the standard deviation, as can be seen the standard deviation on the results for condition C, tangible input, is much lower than the other conditions. This implies that users either really liked or really disliked the keypad or gesture input method.

	A (KeyPad)		B (Gesture)		C (Tangible)	
	Avge	StdDev	Avge	StdDev	Avge	StdDev
Q1	2.09	1.04	1.82	0.98	2.05	0.30
Q2	1.91	1.04	2.18	0.98	1.91	0.30
Q3	2.36	0.92	1.73	1.01	1.91	0.30
Q4	2.27	1.01	1.82	0.87	1.91	0.53

Table 1: *Average Ranking Results for Each Condition*

4.3 Discussion

It was interesting to observe users while they performed the task. As expected, many of them had found the keypad input easy to use as they were familiar with the input technique. Conversely many of the subjects found the gesture interaction frustrating and time consuming. This was mainly due to the limited range of motion that could be tracked by the camera. Several users made comments about wishing that the phone front camera had a larger field of view and that the tracking was more robust.

There were some interesting phone behaviours also observed. In the keypad condition several users preferred to hold the phone steady with one hand while pressing the buttons with the other. In the finger input condition, where this was not possible, some users accidentally moved the phone itself, which made the object move in an unintended way. This is an inherent limitation of this kind of two handed physical interface, which could possibly cause issues for users with reduced mobility, or who find it hard to coordinate two handed activities.

The subjective test results were interesting, as they demonstrated that despite two of the interface methods being novel to the users, they were good enough for them to use. The subjects felt the neither of the gesture and tangible input methods were felt to be significantly worse than the keypad input.

It is also notable that users did not perceive the gesture input as being slower than the other input methods, despite its performance in the timed tests. It is not clear why this occurred. One possible suggestion is that users were continuously actively interacting during this task and so didn't notice the longer time, whereas in the keypad condition, the user just had to hold the button down and wait while the object is moving.

5 Experiment Two: Rotation

5.1 Experiment Task

The rotation experiment was similar to the translation experiment, although involving rotation of objects rather than translating them. It was performed with 13 subjects, 1 female and 12 male. Each user performed a set of 5 timed tasks, in 3 different user interfaces. A training task was also provided in each user interface, which the user was allowed to repeat until they were confident with the interface. Interfaces were tested in all possible combinations of ordering, to avoid learning effects.

In each rotation task, users were presented with a solid object which was initially placed at an angle relative to an identical wireframe object. The task was to rotate the solid object so that it coincided with the wireframe one. When the object was rotated correctly, the object turned yellow, to show that the correct position had been reached, and the task ended. The tasks were designed so as to get progressively harder, this was done by rotating the object away from the target position on an increasing number of axes. The first one required only rotation about a single axis to complete (1 DOF), the second required rotation about 2 axes (2 DOF) and the final task used successive rotations on all 3 rotation axes (3 DOF). Figure 8 shows the rotation task.

After each interface condition, the users were asked the same questions as asked in the translation experiment and were again answered on a Likert scale of 1 to 7, bad to good. After the user had experienced all conditions, they were then asked to rank the conditions in order, based on the same four factors, unstructured user comments were also sought at this point.

The time taken for each user to complete the task was automatically recorded, along with the longitude and latitude error for every quarter of a second during the task, in order to record how users reached to the final configuration.

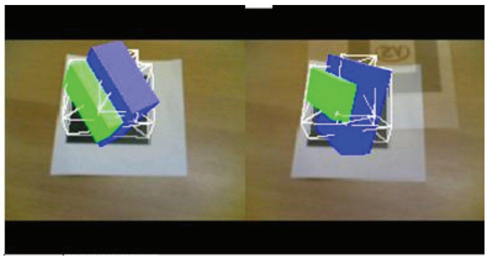

Figure 8: *Virtual Rotation Task*

The rotation task used the following input methods:

A: Keypad input

B: Finger tracking using the front camera

C: Phone tilt tracking

5.1.1 Keypad input

The keypad input used the joypad, plus the 2 and 5 buttons, in order to rotate the model around 3 axes. Joypad input caused rotation about the horizontal axes (x,z) and the 2 and 5 key input caused rotation around the vertical axis (y).

5.1.2 Finger Tracking

In tilt mode, the joypad centre button was used as a "clutch" button, to take control of the object. When it was pressed moving the user's finger in the two dimensions of the screen in front of the front camera would make the virtual object rotate around the two horizontal axes. While the object was being grabbed, the AR tracking is paused, so the object remains on screen at all times, and a "shadow" is shown, behind the current frame, so the user can see how far they have moved the object. A secondary clutch (on the 5 button) was used to twist around the vertical axis.

5.1.3 Phone Tilt Tracking

In tilt mode, the joypad centre button was also used as a "clutch" button. When this was pressed, tilting the phone forwards or sideways caused the AR object to rotate. As with the Finger Tracking, while the object was being grabbed, the AR tracking is paused, and a "shadow" is shown behind the current frame so the user can see how far they have moved the object. A secondary clutch (on the 5 button) was used in order to perform twisting around the vertical axis. This was because the tracking method used for the tilt tracking could not reliably detect this movement.

5.2 Results

There was a significant difference in the time taken between conditions to complete the rotation task, and also within conditions as the complexity of the task increased. Figure 9 shows the average times to complete the tasks for each condition.

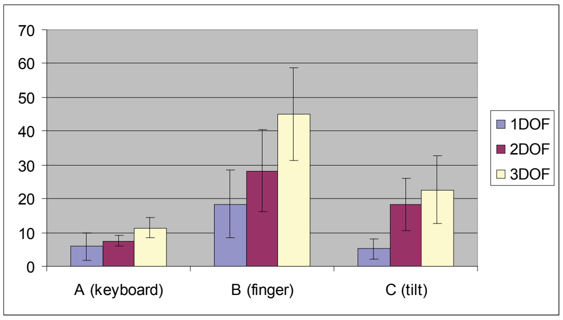

Figure 9: *Virtual Rotation Task Time (s) and Degrees of Freedom*

As can be seen the keypad input is the quickest of the three input conditions, and the with front camera conditions (B and C) the time to complete the task increased as rotations about more axes were needed. A two factor ANOVA (input condition, task DOF) was performed and found a significant difference in results across interface type (F(2,30) = 14.36, $P < 0.001$) and across task complexity (DOF) (F(2,30) = 27.52, $P < 0.001$). There was also a nearly significant interaction between the two factors (F(2,2,90) = 2.37, P=0.06).

The results of the subjective survey after each condition are shown in figure 10. In response to all of the questions the keypad input was felt to be better than the other input conditions. A one factor ANOVA across the conditions found a significant difference in responses to all the questions. Table 2 shows the average responses and the ANOVA F and P values.

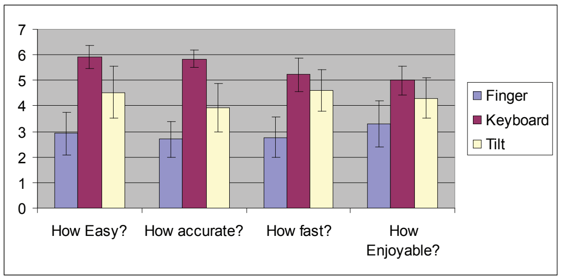

Figure 10: *Mean subjective scores*

	A	B	C	F val	P val
Q1	5.93	2.92	4.54	16.7	<0.001
Q2	5.85	2.69	3.92	23.9	<0.001
Q3	5.23	2.77	4.62	13.8	<0.001
Q4	5.00	3.31	4.31	5.9	<0.01

Table 2: *Average Subjective Scores and ANOVA Result*

However the post hoc testing of combinations of interfaces showed none of them to be significant to 95% confidence, except for the finger interface being significantly different from the keypad in the perceived accuracy question. This warrants further exploration.

Figure 11 shows the average subjective rankings across the same four questions as asked in the translation experiment. Using a one factor ANOVA we found a significant difference in the average rankings, except for Q4: How enjoyable was it to use the application? Table 3 shows the ANOVA results. All the subjective rankings, except for the measure of how enjoyable the interfaces were to use, showed 95% significance between conditions (using a Friedman test), with keyboard being best, followed by tilt and finger. However, the ranking as to how quick they were (figure 11) showed no significant difference between keyboard and tilt input, which is supported by the timing results. Enjoyment showed no significant differences between interfaces, with the keypad and tilt interfaces being particularly similar.

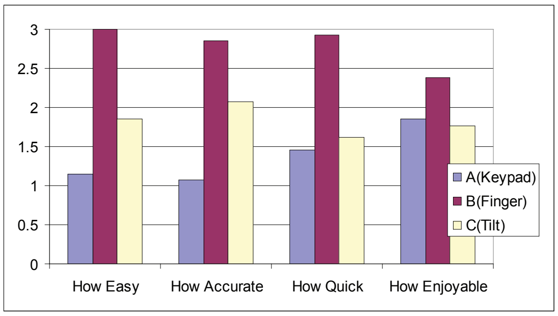

Figure 11: *Subjective rankings (1 = highest, 3 = lowest)*

	A	B	C	F val	P val
Q1	1.15	3.00	1.85	120.3	<0.0001
Q2	1.08	2.85	2.08	66.5	<0.0001
Q3	1.46	2.92	1.62	32.7	<0.0001
Q4	1.85	2.38	1.77	2.3	0.12

Table 3: *Average Subjective Ranking Scores and ANOVA Results*

5.3 Discussion

It is clear from this testing that the keypad input proved easiest for subjects to use. The keypad task times were significantly faster than the tilt and gesture input conditions and users felt that it was easier to use and more accurate. This may have been partly because it was easy for users to mentally map keypad input onto rotation about the desired axis.

The finger tracking interaction mode was difficult for users to work with. The major limitation of the finger tracking was the range of view of the front camera, which is designed for video-telephony, to display the head of the user. This meant that in order to rotate the object a large distance, the users had to clutch and rotate multiple times.

Several users mentioned that they had trouble with the coordination required for the finger input method due to having to use both hands at once; one hand on the clutch button, and the other to perform the rotation. Two of the users also mentioned that they found it hard to move their hand accurately in empty space, with no way to feel how far they were moving it. This caused problems with the users causing rotations on a different axis from the one they were attempting.

An example of the problems with accuracy can be seen in figure 12, which shows the error in rotation around the vertical axis and one of the horizontal axes was reduced during one successful completion of task 5. This was the task that required rotation about all three axes for completion. The keypad input constantly reduces error on both axes until it reaches the correct point, producing a nice diagonal line to the origin. The tilt input reduces error on one axis, then rotates around the other axis until correct. However, the finger input takes a more haphazard approach to the target position, with several accidental rotations.

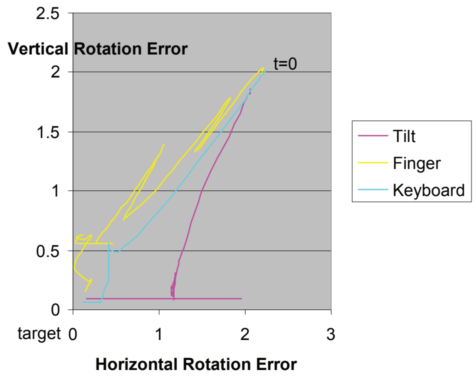

Figure 12: *User Error*

There was one final problem with finger input, which was that it could be affected by the room lighting. For example, a bright light source located above and behind the user could cause their head and fingers to be a black silhouette, meaning that the tracking algorithm could not detect any difference between the finger and the head when the finger moved in front of the face. This was a real problem for one participant, who held the phone at a different angle to the majority of participants and ended up having to turn around in order to avoid the light source.

Use of tilting for rotation was approximately as fast as the keypad input in the 1 DOF rotation task and was around half the speed in the other more complex tasks. Figure 9 earlier in the paper showed the rise in time taken to perform the task, as the tasks became harder (as measured by the number of axes which were used to rotate the object away from the final position). It is interesting to see that for finger tracking, as the tasks became harder, the time scaled roughly linearly, whereas for tilt and keypad tracking, the times became slightly worse than the simplest task, but not extremely so. However, the computer vision system used in this test for the tilt input had a major limitation, which was that it could not detect when the phone was twisted around the axis centred on the front of the phone. Several of the users mentioned this limitation being a problem for them, and one of the users was observed trying to twist the phone repeatedly during the tasks, despite being shown that it could not detect this during the practice task. Arguably, the fact that it performed at a similar level to keyboard input even with this limitation suggests that with twist detection embedded in the main interface this method has the potential to be faster than pure keyboard input.

There is another limitation to tilt input which is inherent to the input mode, which is that tilting the phone also tilts the screen. This may cause problems performing extreme tilts accurately, as the screen becomes hard to see. However, in practice this did not seem to be a problem for most users. Keyboard input worked reliably under all conditions. However, some of the participants expressed frustration with having to wait for it to turn the object around. Several users suggested that the tilting method was more satisfying to use for them because they were constantly doing something during the rotation process.

Arguably if tilt interaction could be made more efficient, it would also be more satisfying than the keyboard interaction. Tilt interaction also integrates well with the 'tangible' interaction mode used in the first study.

6 Design Implication

These user study results expand on the earlier manipulation study [Henrysson et al. 2005b] by using exploring the use of the front camera for gesture input, tilting for rotation and examining a set of tasks of different complexity. Like that study these results confirmed that tangible input can be more effective than keypad input for object translation, while the keypad should be chosen for object rotation. In mobile AR interfaces that involve object translation and rotation it seems that it would be more intuitive for users to split input techniques between tangible input and keypad methods.

However the results from the tilting technique for object rotation are encouraging and for simple rotations produce results as good as with the keypad input. For object rotation about one axis it makes sense to use tilting input, especially if other tangible input methods are being used for the object translation.

The front camera gesture input is interesting, but seems that it will need more work before it can be used for quick and accurate virtual object placement. In particular the problem of limited interaction volume will need to be addressed, perhaps by using a front camera with a wider field of view. However, there are a number of interesting applications that could be implemented using 3 DOF input from the front camera.

7 Sample Application

As a sample application to explore the properties of the front camera input we developed a simple 3D paint application (Figure 13). The paint consists of cubes dropped each frame and at a position defined by a 3D cursor. Interaction is identical to that of translation using finger input with an ARToolKit marker, except now the user must release the switch button and press the joypad button to drop cubes. In this way the user can use the three degree of freedom (x,y,z) position information of the finger tracked by the front camera to drop cubes into the 3D AR space.

The current mapping of the finger position in camera coordinates into scene coordinates is indicated by a blue cube, acting as the 3D cursor. We use the rule of thumb for the relation between marker size and tracking distance and thus define the cursor to be at the near clip plane when the distance to the marker is equal to ten times its width. The finger can then be moved towards the camera until the marker covers the camera image.

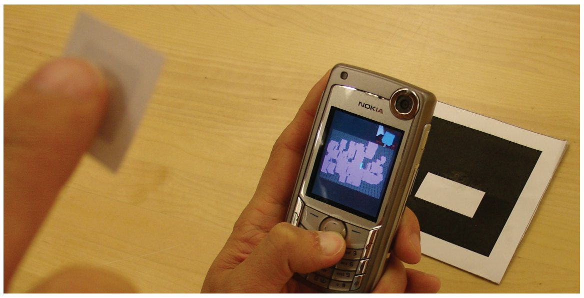

Figure 13: *3D Painting Using 3D Input*

Cubes - with same size as the cursor - are dropped continuously while the joypad button is pressed. The number of cubes is limited to 200 and after the last one is drawn the first is simply moved to the current cursor position. The 3D cube placement allows the user to create a simple virtual sculpture. We did not make a formal user study due to the demonstrational character of the application, but all participants doing the translation study tried the application and then commented on the feasibility of the concept and gave feedback on possible improvements and other application areas. Most users agreed that the range and quality of the tracking was the major factor limiting the usability.

This type of application is one that is best suited to finger tracking from the front camera because it requires a way to quickly and easily specify the 3D location of the cubes being painted into space. If this application was implemented with keypad input it would be far more time consuming and difficult for the users to create their 3D artworks.

8 Conclusions and Future Work

In this paper we have compared several different techniques for 3D object translation and rotation in a mobile phone AR environment. In particular we explored the use of the front camera in the phone to provide finger tracking input for virtual object translation and orientation. Our user study results agree with the results that we found from an earlier study, namely that tangible input techniques provide an intuitive way to translate virtual objects in a mobile AR interface, but that keypad input is better for rotation tasks.

However, in some cases for object rotation the tilting method performed almost as well as keypad input, implying that the vision based input methods could be tuned to provide better performance and make them more intuitive.

In the future we would like to explore the use of a front camera with a better field of view that will support a greater interaction volume. We will also develop alternative mappings from phone motion and finger input to find more intuitive ways to manipulate objects. Finally, our mobile AR studies to date have involved separate studies of object positioning and rotation. Next we need to explore tasks that require combined object positioning and orientation, such as path tracking.

Acknowledgements

We would like to thank Jörg Hauber and Andreas Duesner for their help with analyzing the results of the user study. The first author is funded by Brains & Bricks and CUGS.

References

BEATON, R., AND DEHOFF, R. 1987. An evaluation of input devices for 3-d computer display workstations. In *Proceedings of SPIE*, vol. 761, 94–101.

ELLIS, S. R., BREANT, F., MANGES, B., JACOBY, R., AND ADELSTEIN, B. D. 1997. Factors influencing operator interaction with virtual objects viewed via head-mounted see-through displays: viewing conditions and rendering latency. In *VRAIS '97: Proceedings of the 1997 Virtual Reality Annual International Symposium (VRAIS '97)*, IEEE Computer Society, Washington, DC, USA, 138.

HAND, C. 1997. A survey of 3d interaction techniques. *Computer Graphics Forum 16*, 5, 269–281.

HENRYSSON, A., BILLINGHURST, M., AND OLLILA, M. 2005. Face to face collaborative ar on mobile phones. In *ISMAR '05: Proceedings of the Fourth IEEE and ACM International Symposium on Mixed and Augmented Reality*, IEEE Computer Society, Washington, DC, USA, 80–89.

HENRYSSON, A., BILLINGHURST, M., AND OLLILA, M. 2005. Virtual object manipulation using a mobile phone. In *ICAT '05: Proceedings of the 2005 international conference on Augmented tele-existence*, ACM Press, New York, NY, USA, 164–171.

HINCKLEY, K., TULLIO, J., PAUSCH, R., PROFFITT, D., AND KASSELL, N. 1997. Usability analysis of 3d rotation techniques. In *UIST '97: Proceedings of the 10th annual ACM symposium on User interface software and technology*, ACM Press, New York, NY, USA, 1–10.

MINE, M. R. 1995. Virtual environment interaction techniques. Tech. rep., Chapel Hill, NC, USA.

WANG, J., AND CANNY, J. 2006. Tinymotion: camera phone based interaction methods. In *CHI '06: CHI '06 extended abstracts on Human factors in computing systems*, ACM Press, New York, NY, USA, 339–344.

WITHER, J., AND HOLLERER, T. 2004. Evaluating techniques for interaction at a distance. In *ISWC '04: Proceedings of the Eighth International Symposium on Wearable Computers (ISWC'04)*, IEEE Computer Society, Washington, DC, USA, 124–127.