Thumbs Up: 3D Gesture Input on Mobile Phones Using the Front Facing Camera

Paul Schmieder[1], John Hosking[2], Andrew Luxton-Reilly[1], Beryl Plimmer[1]

[1] University of Auckland, Private bag 92019, Auckland, New Zealand
[2]Australian National University, RSISE Bldg, Canberra, ACT 0200, Australia

psch068@aucklanduni.ac.nz, john.hosking@anu.edu.au,
{beryl, andrew}@ cs.auckland.ac.nz

Abstract. We use the front facing camera in a smart phone to capture gesture input. Thumb gestures performed above the camera are recognized and used to invoke commands. In contrast to other input modalities the camera requires no device movements and no valuable screen space is used. To be viable, this type of interaction requires gestures which are comfortable and memorable for the user and real-time accurate recognition of those gestures. Given the performance constraints of phones and their cameras we needed to determine whether accurate and reliable recognition is possible and identify types of gestures that are recognizable and user appropriate. As a proof of concept, we conducted a user study testing three gestures for performance and user satisfaction. The results demonstrate that the 3D gestural input is successful and we provide detailed insights into successful recognition strategies for this novel interaction modality.
Keywords: Motion gestures, mobile interaction, image recognition

1 Introduction

Typical mobile devices have limited input modalities available: touch screen, two or three buttons, sensors, cameras and microphones. Gesture input has the advantage of supporting continuous input thus reducing the demand for input actions – one successful example of gesture input is the swipe keyboard [15]. The disadvantage of most continuous input modalities is that they use the touch screen which obstructs the information on the display, or the gyroscope or accelerometer that require distracting device movements [10]. Recent smart phones include front facing cameras. Their location means that such cameras are easily accessible away from the screen while their field of view is visible to the user and moving a thumb in this field does not require distracting device movements (Fig. 1). In this paper we explore how *3D gestures* can be used to provide *continuous input* captured by the *front facing camera*.
Using *gesture input* is common for a variety of devices and situations. A main difference is the number of dimensions gestures are performed in. For mobile devices, 2D

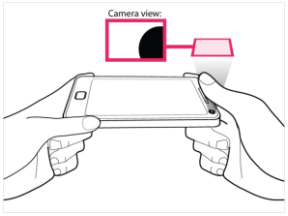

Fig. 1. A smartphone with a front facing camera and a thumb performing a gesture above it on the right.

gestures are common on the touch screen [4], including pinch and zoom, and swipe. 2D touch gestures can also be combined with 3D motion gestures [9]. These use a variety of sensors: gyroscope [9], accelerometer [12] and cameras [21].

A *continuous input* mechanism is desirable, as it provides affordances suited for controlling continuous variables, such as volume. Ideally, a continuous input mechanism is application independent allowing fast and precise manipulation in both directions (e.g. volume up and down, fast for big changes or slow for precise adjustments) such as with the scroll wheels on MP3-Players like Apple's IPod Classic or Microsoft's Zune. However, to adjust volume most portable devices, including smartphones, use buttons. Alternatively continuous input can be captured by the touch screen but this means displaying an appropriate interface element occluding current information. Sensors in smart phones are also able to provide continuous input but usually require distracting and disruptive device movements. For example Hinckley and Song's techniques [9] combine motion and touch using the gyroscope and accelerometer most of which required motions disruptive to user tasks, such as tilting the device away or shaking it.

Gesture recognition is typically a computationally intensive process. Performing such recognition on mobile devices with limited processing power is challenging, particularly when it involves image capture. Wang et al [20] used a Motorola v710 to capture gesture input and reported a maximum rate of 15.2 frames per second (fps), but when image processing algorithms were applied to the input, the capture rate dropped to 12fps. Gu et al. [7] showed that many image processing algorithms reduced frame rates below one when implemented on a variety of different phones. To solve this problem, we developed novel recognition algorithms to reliably recognize the gestures in real-time on mobile devices with limited processing power.

Cameras in mobile devices have been widely used in research systems. This has focused on cameras on the back of the device tracking the devices' position [3] or hand movements [20, 21]. Kato [14] attached a spring with a marker at its end above the back facing camera, allowing continuous input by tracking the marker in three dimensions. However, due to the camera position and the attached spring, operating the marker was rather constrained. Huerst et al. [11] tracked fingers with the back facing camera using markers attached to the tips to capture gestures.

Here we investigate the use of a front facing camera on a mobile device for continuous gesture based input, a combination we believe is novel. Using a front facing camera has multiple advantages: first, the required hardware, camera and finger are

already present. Second, the input is independent of application (except for those directly using the front facing camera) and therefore always accessible. Third, no distracting device movements are required. Fourth, by offering an interaction space away from the display, screen occlusion is reduced. Fifth, the camera can be combined with other sensors to enrich interaction.

There are three components required to support this interaction modality: the *gesture capture*, the *gesture recognition*, and the *gesture design*: all of which contribute to the user experience. To test the feasibility of using three-dimensional gestures for continuous input, we defined three gestures and developed appropriate recognizers with varying demands on usability and hardware. In the second phase we evaluated the gestures on a smart phone in different scenarios. Besides the gesture recognizer performance we were interested in the demands on participants in terms of interaction space and the comfort of the thumb movements.

The results yield three contributions. First, continuous input can be *successfully captured* by front facing phone cameras. Second, current smart phones can *accurately recognize* 3D gestures in real-time. Third, our evaluation provides insights into the *design implications* of our gestures on usability including the definition of the interaction space above front facing cameras where gestures can be comfortably performed.

2 Related Work

Our work builds on related research involving interaction techniques for mobile devices; camera based phone applications; gesture input; and computer vision input.

2.1 Interaction Techniques for Mobile Devices

Some research focuses on facilitating mode manipulations based on the input channels available in mobile devices. For example Hinckley and Song [9] explored gestures combining touch and motion using gyroscope, accelerometer and touch screen. They propose motions for continuous input such as tilt-to-zoom and discrete input such as hard-tap. Research is not limited to the sensor abilities available in current devices. Brewster and Hughes [5] investigated the use of pressure on touchscreens to differentiate between small and capital letters entered on the phone's keyboard.

2.2 Camera Based Interaction on Mobile Devices

Camera based techniques on mobile devices can be differentiated by camera location: integrated in the device or external. Vardy et al. [19] used a camera attached to the wrist to capture finger movements feeding the recognized gestures to a computer for interaction. More common is to use device integrated cameras which are predominantly on the rear. TinyMotion [20] uses a rear facing camera to measure cell phone movements. It computes phone tilts to control games such as Tetris and text input. TinyMotion requires a button press combined with a phone tilt to input a character

[20]. Another application is a see-through tool for applications such as maps [2].The device's position over a surface is recognized and augmented with information.

Kato and Kato [14]developed a tangible controller for continuous analog input. They attached a spring on top of the rear-facing camera with a marker attached to its end. The user provides input by moving the marker in any direction.

2.3 Gesture Input

Gestures, including touch and motion gestures, are commonly used for mobile device interaction. Bragdon et al. [4] investigated the impact of distractions on touch interaction. Comparing gestures and their soft button equivalent, they found gestures are less impacted by distractions and perform as well as soft buttons without distractions.

Aiming to build a gesture set for smartphones Ruiz and Lank [18] conducted a study in which end-users proposed gestures for given scenarios. They found consensus among study participants regarding movement parameters and gesture-to-command mappings. The resulting taxonomy differentiates between *how* a gesture is mapped to a command and a gesture's physical characteristics. Using thumb input on touch screens is widely researched. For example, Karlson et al. looked at the interaction between hands and mobile devices including user preferences and mechanical limitations [13].

2.4 Computer Vision Based Input

Computer vision systems are widely used to collect visual information for gesture based input [6]. The choice of technique depends on factors such as desired information, environmental circumstances and available computational power. Commonly sought information is recognition of objects and/or gestures [16] and object tracking [22]. To gain this information environmental factors such as lighting, motion speed, blur and occlusions as well as their speed of change must be considered [17].

Tracking a large fast moving object dominating a scene is challenging. Fast optical flow algorithms and graph cut approaches are viable but computationally expensive [1]. With limited compute power, temporal difference images have been used successfully to detect coarse-scale motion [6]. To further deal with changing lighting and blur, other properties of the tracked object such as object shape and color [23].

3 Gestures

The novelty of using front facing cameras for gesture input on a mobile device means the feasibility of such a system is unknown. To address this, we conducted a user study testing three gestures of varying demands on hardware performance and user input capability. This study addresses three questions: 1) are current mobile devices powerful enough for camera based gesture recognition in real-time? 2) are 3D gestures performed above the camera an easy and precise way to capture input? 3) what is the interaction space above the camera where gestures can be comfortably per-

formed? To answer these, gestures of differing complexity were tested for recognition accuracy and usability in different scenarios. For the development and testing we exclusively use a Samsung Omnia W, featuring a single core 1.4 GHz Scorpion processor and Windows Phone 7. As results will show, camera based gestural input can be successfully used in mobile devices for enriched natural interaction.

3.1 Design

We designed three gestures and an accompanying recognition algorithm to accurately identify each gesture. To allow for continuous input, each gesture can be performed such that it differentiates between two motion directions: for example to increase and decrease the volume, to zoom in and out of an image or to scroll a list up and down. As the motion direction depends on the phone's orientation, the orientation was used by the recognizers to decide what motion direction maps in to what input direction. Additionally, each gesture has different levels of acceleration: for example, one level may increase the volume by one each iteration, another by three and another by five. Each gesture has a neutral state where no changes occur.

Before the gestures are recognized each captured image is preprocessed. The first step scales the image to 40 x 30. We tested different resolutions for the gesture recognizers and found 40 x 30 to be the best tradeoff between performance and accuracy. Additionally, the image is converted from 32 bit RGB to 8 bit grayscale.

The thresholds used for the algorithms were obtained by testing the algorithms under different lighting conditions: bright: outside during a sunny day; neutral daylight: in the office with open blinds; artificial light: in the office with blinds down and light on, half dark: in the office with blinds down. Completely dark was not trialed as the algorithms require some light to track the thumb.

Tilt Gesture. The thumb is lying on the camera and its tilt determines the input (Fig. 2). The tilt direction and intensity determines the direction and acceleration of change in input. Tilting the thumb to the right maps to an increase in input and tilting to the left a decrease. The higher the tilt is the higher the level of acceleration. If no tilt is applied to the thumb covering the camera no changes in input are invoked.

The recognition algorithm analyzes the light captured by the camera to calculate the input. The first step is to split the captured image in half depending on the phone orientation: the split divides the image so that each half is covered by a half of the thumb. Second, the average brightness value is computed over the pixels in each half. The image half with the lower brightness is the side the thumb is tilted towards. Similar low brightness values mean that the thumb evenly covers both image halves so no input changes are triggered. Finally, based on the brightness values and thus the amount of tilt, a step function is used to determine the level of input acceleration: the brighter the image half the higher the acceleration (Fig. 2).

A brightness threshold determines if the tilt gesture is currently being applied. The gesture recognizer is only used if the average brightness over the complete image is below a predefined threshold. This threshold is set to 55% of an image brightness value taken when nothing covers the camera (e.g. when the application is started).

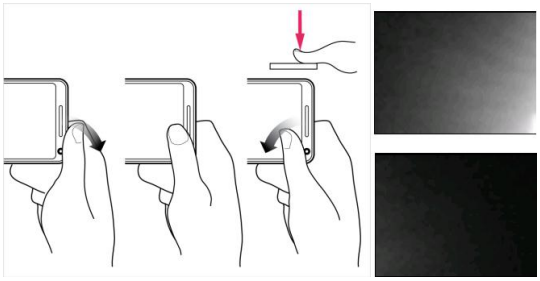

Fig. 2. The tilt gesture: Left: Tilting the finger to either side determines the direction of input and the amount of tilt determines the level of acceleration. Right: The captured images used to compute the tilt. (top) A strong tilt to the left and (bottom) a slight tilt to the right.

The chosen threshold provides the best offset between brightness ranges to determine the level of acceleration and activate/deactivate the gesture recognizer.

Distance Gesture. The distance between thumb and front facing camera is used to adjust the input (Fig. 3). A predefined distance marks a neutral point where no input manipulation happens. A smaller thumb-camera distance triggers a decreased input and a larger one an increase. The exact distance determines the level of acceleration. Input is based on the thumb's diameter in the captured image: the larger the diameter the closer the thumb. To calculate the distance, the thumb's position in the captured image has to be determined. Then, brightness values are computed over the image. Finally, the input value is determined using the thumb's distance and brightness.

To detect the thumb in the image a Sobel edge detector with a 3x3 kernel is applied. The resulting Sobel image (Fig. 3) contains objects defined by their borders. To identify the thumb three assumptions are made: 1) the thumb partly covers the center row of the Sobel image; 2) in the center row the thumb is the biggest object; 3) the thumb forms an uninterrupted blob starting at the image. Hence the recognition algorithm analyzes the center row of the image looking for the biggest group of dark pixels bounded by white pixels. If the biggest section is above half the image's width, the algorithm further tests whether the section is part of a blob connected to at least one image edge. If no object satisfies all three the criteria the algorithm discards the image and no input is invoked assuming the gesture is not currently being performed.

The size of the range between minimum and maximum thumb distance is divided into smaller spans, each accommodating a different input value. A span has to be sufficiently large to be easily found by the user in terms of distance of his/her thumb to the camera. For example, in a 40 pixel wide image the total range is 20 as the thumb must be at least half the image width wide. This leaves 20 pixels to be split into smaller spans. We found spans of one-eighth of the image width to work best. Brightness values determine if a thumb is covering the camera completely. This allows further differentiations when the thumb covers the captured image width completely but does not yet physically cover the camera.

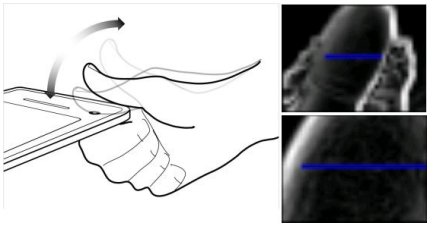

Fig. 3. The distance gesture: Left: the input depends on the distance between camera and thumb. A predefined distance is the neutral input where no changes are invoked. Right: The images after the Sobel operator has been applied with the thumb further away (top) and close (bottom) to the camera.

Circular Gesture. A circular motion is performed close above the camera to adjust the input (Fig. 4). Depending on the motion direction (clockwise or counter-clockwise) input is either increased or decreased. The speed of the gesture determines the level of acceleration.

The circle gesture recognizer first detects the thumb. Afterwards the thumb's center of mass is used to calculate the motion direction. Motion speed is based on the overlap of thumbs between two consecutive images.

To detect the thumb a flood fill algorithm is used based on the captured image's greyscale values. The thumb appears darker than the rest due to the movement close to the camera: cameras found in smart phones are not fast enough to adjust their blending settings to evenly illuminate the image. As the thumb changes illumination conditions quickly, the camera makes the background behind the thumb appear much brighter than normal. The flood fill algorithm finds the biggest connected thumb blob composed of pixels with an 8-bit greyscale value below 90. If the biggest blob covers more than 21% of the captured image it is assumed to be the thumb. Otherwise the thumb is assumed not to be in the image. If no thumb is detected in more than four successive images further computational steps are stopped and only resumed when a thumb is found in at least four successive images.

Motion direction is calculated using the cross products of three successive images. The weighted average location for each thumb blob is calculated to determine its center of mass (Fig. 4): the center of mass's x-coordinate is the average value over all the blob pixels x values. The same applies to the y-coordinate. Afterwards, the cross products between the four most recent thumb centers of mass are computed. If the sum of the three resulting vector magnitudes is positive the gesture motion is clockwise and otherwise counter clockwise.

Motion speed depends on movement and on the number of intersection points between two successive thumb blobs. To detect a stationary thumb the image is divided into four even quadrants. If the thumb remains in the same quadrant over four successive images, the motion counts as stopped. If not, the thumb is moving and the overlap of two thumbs in successive images is computed. A step function based on the overlap percentages is used to define thresholds for the different levels of acceleration depending on the number of acceleration levels. For example, given two acceleration levels, our empirical tests showed a threshold of 42% to be optimal: a higher overlap

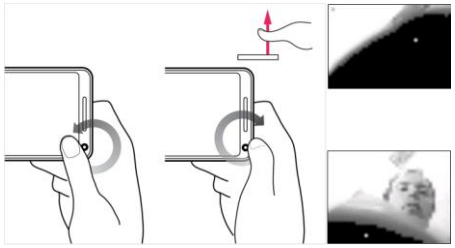

Fig. 4. The circular gesture: Left: The direction of the circular motion determines the input direction and the speed of the motion determines the level of acceleration. Holding the finger at a position invokes no change (i.e. neutral input). Right: The images with the thumb in black after the flood fill algorithm. The white dot in the black blob indicates the center of mass.

than 42% means a slower acceleration and a smaller overlap a higher acceleration. Using quadrants to detect a stationary thumb rather than overlaps is better as it compensates for small unintentional thumb movements.

3.2 Performance & Robustness

The computation of a new input value consists of multiple steps of which capturing the image is the most time costly (Table 1). If no other computations are performed the Samsung Omnia W captures up to 16 images per second. In the following we comment on the robustness and accuracy of the recognizers in general and under different lighting conditions (see section 3.1).

The tilt and circle gestures are both unaffected by different lighting conditions and fast lighting changes. Only when it was completely dark did both recognizers stopped working. The main reason for the robustness is that in both cases the thumb dominates the captured images making the background appears bright and unfocused (Fig. 4).

The distance gesture worked perfectly under the tested lighting conditions except in the half dark condition. In this condition if there were multiple dark objects in the image background they were occasionally merged with the thumb: because of the insufficient lighting the Sobel operator did not detect all the edges. Fast lighting changes did not affect the algorithm.

Table 1. Performance measurements in milliseconds using a Samsung Omnia W phone.

	Tilt Gesture	Distance Gesture	Circular Gesture
Pre-process Image	image captured (66.75msec) scaled & converted to greyscale (1.9msec)		
Process Image	-	4.93 (Sobel)	
Detect Thumb	-	1.12	2.88 (Flood Fill & mass center)
Compute Direction & Acceleration	0.97	0.51	1.76

4 Evaluation

We conducted an evaluation study to explore the interaction with a smartphone using continuous gestures captured by a smartphone's front facing camera. This study focuses on the requirements of suitable 3D gestures and the space they are performed in with the aim to inform further research regarding the basic requirements of this novel and yet unexplored interaction. Four pilot participants were used to determine algorithm parameters such as the number of acceleration levels and the rate at which changes were applied. Only one of the recognizers was activated at each point in time requiring a button press to change between them. We were specifically interested in the following questions:

- What defines the interaction space in which gestures can be comfortably performed above the camera?
- Are the evaluated gestures easy and intuitive and are their associated apps an appropriate match?
- Is the phone's hardware sufficiently powerful to guarantee real-time performance?
- Are the gestures recognized accurately in terms of direction and acceleration changes and reactiveness?

4.1 Participants

Participants had to have a smartphone as their mobile device to guarantee device familiarity. We recruited 24 participants, 8 females and 16 males, ranging in age from 19 to 41 (mean: 25.63 years, SD = 5.17). Five were left-handed. There was no compensation for participation. Since accurate tracking objects may depend on the visual properties of the tracked object, we wanted to ensure a variety of different thumb size, shape and color were represented in the evaluation. To test the robustness of our algorithms we intentionally recruited participants from different ethnic groups (10 Caucasians, 6 Indians, 5 Asians and 3 from the Middle East) and of different height as an approximation of thumb size (mean: 171cm, SD = 8.62).

4.2 Apparatus

To test the gestures, participants used a Samsung Omnia W I8350 featuring a single core 1.4 GHz Scorpion processor and Windows Phone 7. The front facing camera was located on the shorter side's corner (Fig. 1). The capture rate of the front facing camera is on average between 8-10 images per second on the Windows Phone

The gesture recognizers were integrated in a program running on the phone. Each recognizer was configured to support two directions and to differentiate between two acceleration levels: one and three. We decided to use two acceleration levels based on the results from the four pilot study participants. While the tilt and circle gestures would have worked intuitively enough with three acceleration levels, the distance gesture did not. When the thumb was moved with the lower acceleration changes were applied every 500ms to allow for accurate input. This low change rate was chosen as the pilot study participants frequently overshot the target with a faster change

rate. For the faster acceleration level changes were applied after each image was analyzed and the new value computed (i.e. on average every 75msecs). Sessions were recorded using a video camera mounted above the participants.

4.3 Tasks

We designed three tasks each dedicated to a particular gesture and one for familiarization across all three gestures. Each task required participants to use all facets of the gesture so that the invoked input direction and level of acceleration were frequently changed. The choice of task for a particular gesture was based on similar mappings found in current environments as explained in the following. The order of gestures was varied between the participants forming six conditions.

Contact List – Circle Gesture. To scroll a list using a circular motion is familiar practice. MP3 players such as Apple's iPod Classic use a circular motion to browse their content. Scrolling a list requires participants to bridge bigger distances faster as well as precise manipulations to jump between neighboring list entries.

The scrolling mini app displayed a list of 78 contacts to simulate a standard smartphone Contact list (Fig. 5a). To scroll down/up, participants performed a clockwise/anti-clockwise motion. If the acceleration level was 1 the selector, indicated by a blue color, jumped from the currently selected contact to the next neighbor. An acceleration level of three jumped three contacts at once.

In the top right corner a name was displayed which was also in the contact list. Participants were asked to scroll to that name in the list as fast as possible. The correct name had to be selected for two seconds requiring participants to provide neutral input. After two seconds the name in the top right corner was replaced by another name from the contact list. In total, participants had to scroll to 10 different names. The choice of names made participants perform precise selections (e.g. to go from one name to a neighbor) and to bridge larger gaps (e.g. Paul to Billie).

Zooming – Distance Gesture. Participants controlled the size of a green square on the screen (Fig. 5b). To zoom in, participants moved their thumb close to the camera and to zoom out the thumb was moved further away. This gesture is familiar for zooming as going closer to an object makes it naturally appear bigger and going further away makes it smaller.

An acceleration level of one increased or decreased the square's size by 10 pixels and a level three by 30 pixels. The square can have any width between 0 and 300 pixels resulting in 30 distinct zoom-able sizes.

The participants' aim was to match their square's size with a red frame shown on the display. To indicate equal sizes, the red frame turned white. Participants had to keep the size constant for two seconds, requiring them to adjust the thumb camera position to neutral, before the frame changed size and turned red again. In total the red frame changed size 10 times.

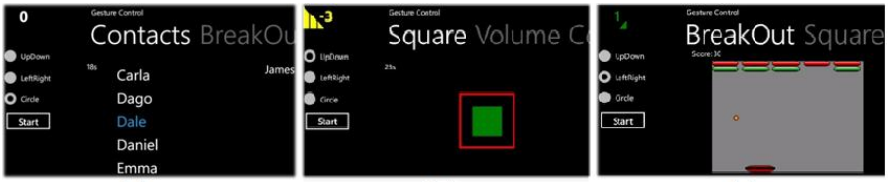

Fig. 5. The test program showing the mini apps: (a) the contact app, (b) the zooming app and (c) the Break-Out app. In the top left side an indicator is shown displaying the currently captured input ((a) neutral, (b) fast-left, (c) slow-right).

BreakOut - Tilting Gesture. Participants controlled the paddle in BreakOut (Fig. 5c) to bounce the ball upwards. The aim was to destroy all ten bricks lined in the top of the game area by hitting each with the ball. After a brick was hit, the ball was deflected down towards the paddle. The angle of deflection when the ball hit the paddle depended on where the ball hit the paddle. To destroy the bricks the user needed both small and large paddle position adjustments forcing them to use all facets of the tilt gesture. The game area was 300 pixels wide. An acceleration level of one meant a paddle move of 4 pixels while a level of three meant a 12 pixel move. In total the paddle had 75 distinct positions. Each destroyed brick scored 10 points making 100 points the highest score. The game was also used in previous research to study camera based input techniques [8].

4.4 Procedure

Participants were seated at a table with the smartphone lying immediately in front of them. They were given a tutorial guiding them through the study. The participants' first task was to complete the first section of the questionnaire.

To familiarize themselves with the Windows Phone participants were asked to pick it up and the facilitator explained the main elements on the screen. Once familiar, the participants started the test application. During this the phone was held in portrait orientation before changing to landscape to interact with the test application.

At the beginning of each gesture the facilitator explained the gesture to the participant. Afterwards the participant was told to navigate to the volume mini app and trial the gesture. Then the participant completed both a gesture specific task and the questionnaire for this gesture. This was repeated for each gesture.

After all tasks were finished the final section of the questionnaire was completed. The experiment concluded with a discussion of other possible gestures allowing continuous input. It took the participants on average 29 minutes to complete the study.

5 Results

The collected data includes the questionnaires, session video recordings and transcripts of the final discussion. The questionnaire comprised three sections. The first

gauged participants' familiarity with gesture input. The second asked participants to rate the gestures regarding their ease and recognition accuracy using a five point Likert scale. The last contained a ranking of the gestures as well as questions gauging the interaction space above the camera including camera location.

5.1 Interaction

All 24 participants held the smartphone in landscape orientation with the camera on the right. Regardless of their handedness all participants chose to use their right thumb to perform the gestures. The matches between gesture and associated mini apps were never rated as negative thus confirming our design choices (Fig. 6).

Ideal Camera Position. The camera should ideally be centered on the horizontal axis in landscape mode on the phone's right side. When asked in the questionnaire to indicate the optimal camera position on the phone all participants chose the right side of the screen. Six participants chose the top corner, eight the center and nine the lower corner. Ten participants noted that the camera on the horizontal axis should be centered between screen and phone edge. The camera on the test phone was closer to the edge than to the screen which sometimes created problems with the tilt and circle gesture: a tilt to the right (of a big thumb) meant a tilt slightly over the phone's edge. While performing the circle gesture some participants drifted away from the camera's center. They stated that they used the space between screen and phone edge as a frame of reference with their circle motion touching both sides.

Keeping the Thumb in the Camera's Viewing Field. Each gesture used a different interaction space above the camera with different demands on the user. The tilt gesture was the easiest one to keep in the camera's viewing field as it essentially was performed as a two dimensional gesture with the thumb lying flat on the camera (Fig. 6). It received ratings above neutral with 83% of the participants strongly agreeing to the ease and 17% agreeing. In contrast, the ease of keeping the distance and circle gestures in the camera' viewing field was rated lower.

The dominant usability problem with the distance gesture was to keep it in the viewing field. To measure the distance between camera and phone, the recognition algorithm calculates the thumb's diameter. If the thumb is close but not centered above the camera, it is only partially visible thus its measured diameter is smaller than it actually is so the calculated distance is incorrect. As the thumb moves away from the camera it does so in an arc. Its tip moves towards the bottom of the phone as it moves up thus leaving the camera's viewing field on the lower edge. However, this usually occurs after the thumb position starts feeling uncomfortable due to the high finger stretch. In the questionnaire we asked participants what the maximum distance between thumb and camera is before holding the thumb becomes uncomfortable. Measured with a ruler, the average distance stated was 4.75 cm (SD = 1.12cm, Min = 2 cm, Max = 6 cm). The majority of the participants stated that it becomes uncomfortable just before thumb and thumb socket form a straight line.

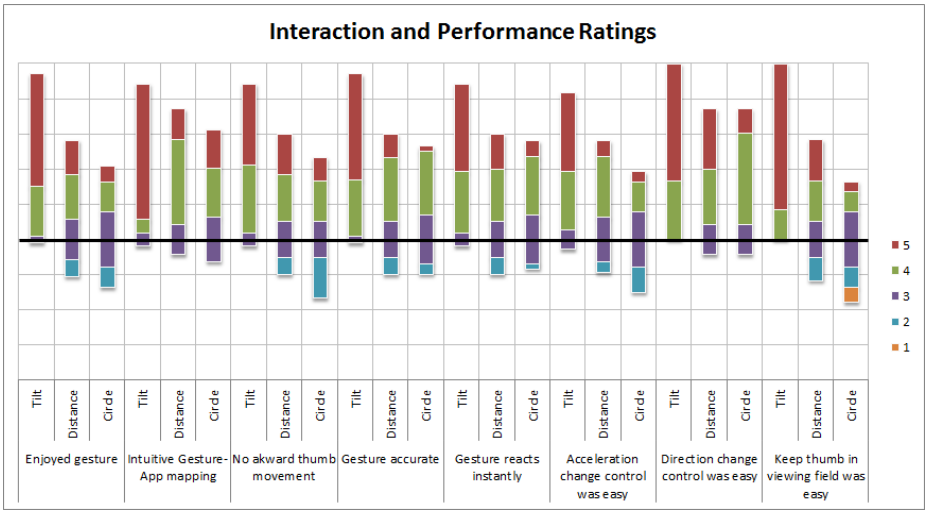

Fig. 6. The questionnaire ratings for the three gestures on a 5 point Likert scale centered around neutral. The ratings are: 1= strongly disagree, 2 = disagree, 3 = neutral, 4 = agree, 5 = strongly agree.

For correct recognition, the circle gesture had to be performed at a certain height above the camera and the motion had to circle the camera. The gesture did not require participants to draw a perfectly round circle nor one perfectly centered above the camera. However, moving outside the camera's viewing field, as some participants did, made it impossible to track the thumb resulting in incorrect recognition. On occasion the thumb was too close to the camera so the captured image was almost totally dark leading to incorrect recognition. Participants on average found it harder to keep the thumb in the right area for the circle compared to the distance gesture with the distance gesture having 17% rating it below average and the circle gesture 29%. This was also reflected in the questionnaire where eight participants said it was difficult to keep the circular movement above the camera. There were no complaints about the distance gesture.

In summary, the tilt gesture is the easiest with only positive ratings. The distance gesture is the second highest rated followed by the circle.

Ease of Gesture Motions. The thumb movements for the distance and circle gestures were perceived as physically uncomfortable at times. Having to hold the thumb still at changing positions over an extended duration (Zooming task: Ave = 141 sec, SD = 36) was perceived as uncomfortable by 12.5% of the participants. For the circle gesture, 29% of the participants experienced discomfort when performing the gesture: three stated their thumb movement was restricted as they were holding the phone with the same hand. This was particularly evident when the thumb tip moved close to the thumb base forming a right angle between the thumb's first and second phalanges.

5.2 Phone Performance and Recognition Accuracy

All gesture recognizers and mini applications ran in real-time on the phone. In the questionnaire participants rated the overall accuracy of the gesture recognizer as well as the recognizers' reactiveness and the ease and correctness of changing the acceleration level and direction (Fig. 6).

The gestures and their associated mini applications ran smoothly apart from several occurrences when the camera froze for two seconds during interaction. This occurred randomly for different recognizers and appeared to be an artifact of the phone's API to the camera buffer. We speculate that this is unlikely to be an issue with a multicore processor phone where camera and interface run on different threads.

The recognizers' reactiveness was rated neutral and above for the tilt gesture and neutral and above by 21 participants for the distance gesture and by 23 for the circle gesture (Fig. 6). To allow for precise adjustments changes were applied every 500ms, which some participants perceived as too slow. For the distance gesture, changes were invoked instantly when the thumb distance entered a new range. For example, to zoom in with an acceleration level of one the thumb had to be roughly (depending on thumb size) between 3 and 4 cm away. Any other distance range was mapped to another input. If the thumb position was changed but not enough to change ranges, participants perceived this as slow to react. For the circle gesture the thumb had to rotate half a circle to trigger any change in input. A lower threshold meant that sometimes unintended changes were invoked as the circular motion was not performed.

Changing the level of acceleration was rated below neutral for the distance and circle gesture by two and five participants respectively (Fig. 6). For the distance gesture, to change the level of acceleration, the finger has to be in the distance range associated with the desired acceleration. To find the correct range one either recalls it from practice/experience or moves towards the desired range until the input value changes to the desired outcome. As participants had little training not all remembered the positions perfectly so had to use the second strategy which some perceived as suboptimal. For the circle gestures, correctly recognizing the acceleration required the circle to be performed above the center of the camera. If this condition was violated the recognizer could not accurately differentiate between the acceleration levels of one and three. Due to the problems of keeping the thumb in the required space (Fig. 6), the acceleration levels were occasionally recognized incorrectly.

Changing the input direction was always rated as neutral or higher for all gestures (Fig. 6). Differentiating between directions was easier than changing the acceleration levels because the motions were more different thus easier to remember and recognize. For example, for the circle gesture one had to change the motion direction regardless of acceleration. Additionally, to accurately recognize the correct direction it was sufficient for the camera to capture part of the circular motion thus not requiring a centered motion above the camera.

To sum-up, well designed gesture recognition algorithms can run in real-time on current generation smartphones. The gestures allow for accurate adjustments of acceleration and directions if the gesture requirements such as motion path and accuracy are satisfied.

5.3 Summary and New Gestures

Both, natural gestures and accurate recognizers are required for satisfactory interaction. If a gesture is too complicated to be intuitively performed, a recognizer may be unable to deal with the inaccurately performed motion. The three tested gestures varied in terms of motion complexity; this was reflected by how much participants enjoyed each gesture (Fig. 6) and the final gesture ranking. The tilt was the easiest to perform and was ranked first by 21 of 24 participants and second by the remaining participants. The distance gesture was more difficult to perform mainly because the desired camera-thumb distances were hard to remember with the little practice. Thus, the distance gesture was ranked second by over half the participants (1^{st}: 2, 2^{nd}: 13, 3^{rd}: 9 participants). The circle was the most difficult of the three, as it required the participants to first, perform the motion at a certain distance above the camera while, second, centering the motion. The gesture accurately recognized only if both criteria were satisfied. 15 participants ranked the circle gesture third, 8 second and 1 first.

When asked for other gestures, participants proposed one of three gestures. The first, proposed by four participants, is a circular motion but on the z-axis above the camera. Thus staying inside the camera's viewing field would become easier. Circle direction and speed determine acceleration level and direction. The second and third are variants of swipe motions. The second, proposed by four, consists of a horizontal swipe motion with the swipe direction determining the input direction. Once the swipe in a direction reached the outer border the finger is either lifted high above the camera or besides it to return to the other side and continue the gesture. The third proposed by seven has an additional horizontal swipe that makes the extra motion to return to the start position redundant.

6 Discussion

The aim of the presented work was to investigate the potential of a yet untested interaction to capture continuous input with the focus on performance and 3D interaction. Due to the new challenges we felt that such an evaluation was required before other evaluations such as comparing the proposed techniques with state-of-the-art interaction methods (e.g. touch screen based input) were conducted. Additionally, with the knowledge gained a more detailed study looking at creating user defined gesture set is possible similar to [18].

6.1 Capturing and Recognizing Gestures

Gesture recognition using front facing cameras has to run in real-time despite a mobile devices' limited hardware performance. To track objects like a thumb a number of tracking approaches may be *unsuitable* due to high performance requirements. We found the following methods provide sufficient information while allowing for real time interaction: working with the image's *brightness*, *edge detection* and *differential images*; the last is unsuitable if fast movements are involved due to fast lighting changes.

A captured image's brightness (or the excess of it) can be used to track moving objects near the camera. If a tracked object moves fast and covers at least quarter of the image, the background is overexposed making it too bright with almost no contours. Using a threshold to extract the close darker object from the background is easy and has low performance cost as shown by the circle recognizer.

To track an object at any distance that is not moving, fast edge detectors can be used. To identify the object based on its edges additional information is required such as its shape, origin and/or size. The last two were successfully used for the distance gesture where the finger always originates from at least one corner and covers at least a certain proportion of the image.

Capturing the gestures with the camera using these algorithms has some disadvantages. Constant capturing of images consumes more power draining the battery faster. While it does not matter whether the light is natural or artificial, sufficient lighting is still required: our tests showed that except for the Sobel operator, the algorithms worked in all conditions except the completely dark one.

6.2 Designing Gestures

Designing successful motion gestures means creating intuitive gestures without constraints. As the camera has a limited field of view gestures have to be fitted to this area: gestures close to the camera have to be directly above it while those further away can be further away from the camera's center. However, gestures cannot be too far away as this becomes uncomfortable to the point where holding the device with both hands becomes impossible.

Each state in a gesture motion must be easily identifiable and performable. If it is unclear what motion to perform to effect change or if the motion requires physically uncomfortable moves, the gesture may be unsuitable for continuous input. The distance gesture showed that having no explicit knowledge of the required motion to change to a given acceleration level may dissatisfy users.

The number of acceleration levels depends on the differentiability of the gesture motions. For example, with the distance gesture there was a total range of 6 cm available, which had to be divided in sub-ranges each dedicated to a unique input value. Trying to divide this range into more than five ranges (two directions with two acceleration levels each plus a neutral range) would be unrealistic as users' ability to discriminate those ranges is too difficult as shown in the evaluation. User feedback is important, a change in gesture must immediately cause a change of input while providing sufficient control for small changes.

6.3 Designing the Device

The camera position preference, regardless of the users' handedness, is on the right side of the phone. When asked for their preferred camera position, the participants' answers did not show any strong preference between the vertical position (top, center or bottom). However, there was a preference for the camera to be centered on the phone border.

Regardless of the camera position the gestures did not impair screen visibility. When asked, participants stated that they were able to see the entire screen during the interaction. An analysis of the video recording showed that items of importance such as the next name for the scrolling tasks situated on the screen's right border next to the camera were no covered by any gesture.

7 Conclusion & Future Work

We developed, implemented and evaluated three gestures and associated recognizers to allow for continuous input on portable devices. To capture input the front facing camera is used as its location offers an easily reachable input channel away from the screen. Using camera-based gestures avoids the need to move the device, which would potentially interrupt the interaction, and does not occupy valuable screen space.

Our work shows how 3D *gestures* are successfully *captured* and *recognized*. Different low-cost recognition strategies are implemented and trialed. The evaluation demonstrates the robustness of the recognition algorithms under different conditions; e.g. different lighting conditions and different colors and sizes of the tracked thumbs. The evaluation also provides insights into, first, the design of gestures and their requirements and, second, the interaction space above front facing cameras where gestures are comfortably performed.

Motivated by the results of our study we are exploring further gestures for discrete and continuous input. We are also looking at combining camera-based gestures with other input modalities such as data captured by gyroscope and accelerometer. Motivated by [13], we are also exploring making camera based interaction accessible for one handed interaction.

References

1. Barron, J.L., D.J. Fleet, and S.S. Beauchemin, Performance of optical flow techniques. International Journal of Computer Vision, 1994. 12(1): p. 43-77.
2. Bier, E.A., M. C. Stone, K. Pier, W. Buxton and T. D. DeRose, Toolglass and magic lenses: the see-through interface. in Proceedings of the 20th annual conference on Computer graphics and interactive techniques. 1993. ACM. p. 73-80.
3. Boring, S., D. Baur, A. Butz, S. Gustafson, and P. Baudisch, Touch projector: mobile interaction through video. in Proceedings of the 28th international conference on Human factors in computing systems. 2010. ACM. p. 2287-2296.
4. Bragdon, A., E. Nelson, Y. Li, K. Hinckley Experimental analysis of touch-screen gesture designs in mobile environments. in Proceedings of the 2011 annual conference on Human factors in computing systems. 2011. ACM. p. 403-412.
5. Brewster, S.A. and M. Hughes. Pressure-Based Text Entry for Mobile Devices. in Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services. 2009. ACM. p. 1-4.
6. Freeman, W.T., et al., Computer vision for interactive computer graphics. Computer Graphics and Applications, IEEE, 1998. 18(3): p. 42-53.

7. Gu, J., R. Mukundan, and M. Billinghurst. Developing Mobile Phone AR Applications using J2ME. in Image and Vision Computing New Zealand, 2008. IVCNZ 2008. 23rd International Conference. 2008. IEEE. p. 1-6.
8. Hansen, T.R., E. Eriksson, and A. Lykke-Olesen. Use Your Head: Exploring Face Tracking for Mobile Interaction. in CHI '06 Extended Abstracts on Human Factors in Computing Systems. 2006. ACM. p. 845-850.
9. Hinckley, K. and H. Song. Sensor synaesthesia: touch in motion, and motion in touch. in Proceedings of the 2011 annual conference on Human factors in computing systems. 2011. ACM. p. 801-810.
10. Holz, C. and P. Baudisch. Understanding Touch. in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. 2011. ACM. p. 2501-2510.
11. Hürst, W. and C. Wezel, Gesture-based interaction via finger tracking for mobile augmented reality. Multimedia Tools and Applications, 2012: p. 1-26.
12. Iwasaki, K., T. Miyaki, and J. Rekimoto. Expressive typing: a new way to sense typing pressure and its applications. in Proceedings of the 27th international conference extended abstracts on Human factors in computing systems. 2009. ACM. p. 4369-4374.
13. Karlson, A.K., B.B. Bederson, and J. Contreras-Vidal, Understanding one handed use of mobile devices. Handbook of research on user interface design and evaluation for mobile technology, 2008.
14. Kato, H. and T. Kato. A camera-based tangible controller for cellular phones. in Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services. 2009. ACM. p. 1-2.
15. Kushler, C.A., Marsden, Randal J., System and method for continuous stroke word-based text input 2003, ForWord Input Inc.: USA.
16. Mitra, S. and T. Acharya, Gesture Recognition: A Survey. Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, 2007. 37(3): p. 311-324.
17. Moeslund, T.B. and E. Granum, A Survey of Computer Vision-Based Human Motion Capture. Computer Vision and Image Understanding, 2001. 81(3): p. 231-268.
18. Ruiz, J., Y. Li, and E. Lank. User-defined motion gestures for mobile interaction. in Proceedings of the 2011 annual conference on Human factors in computing systems. 2011. ACM. p. 197-206.
19. Vardy, A., J. Robinson, and C. Li-Te. The WristCam as input device. in Wearable Computers, 1999. Digest of Papers. The Third International Symposium on. 1999. p. 199-202.
20. Wang, J., S. Zhai, and J. Canny. Camera phone based motion sensing: interaction techniques, applications and performance study. in Proceedings of the 19th annual ACM symposium on User interface software and technology. 2006. ACM. p. 101-110.
21. Wigdor, D., C. Forlines, P. Baudisch, J. Barnwell, C. Shen Lucid touch: a see-through mobile device. in Proceedings of the 20th annual ACM symposium on User interface software and technology. 2007. ACM. p. 269-278.
22. Yilmaz, A., O. Javed, and M. Shah, Object tracking: A survey. ACM Comput. Surv., 2006. 38(4): p. 13.
23. Yilmaz, A., L. Xin, and M. Shah, Contour-based object tracking with occlusion handling in video acquired using mobile cameras. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 2004. 26(11): p. 1531-1536.