A Comparison Between Tilt-input and Facial Tracking as Input Methods for Mobile Games

Justin Cuaresma
Dept. of Electrical Engineering and Computer Science
York University
Toronto, Canada
justincuaresma@gmail.com

I. Scott MacKenzie
Dept. of Electrical Engineering and Computer Science
York University
Toronto, Canada
mack@cse.yorku.ca

***Abstract*—A user study was performed to compare two non-touch input methods for mobile gaming: tilt-input and facial tracking. User performance was measured on a mobile game called *StarJelly* installed on a Google *Nexus 7 HD* tablet. The tilt-input method yielded significantly better performance. The mean game-score attained using tilt-input was 665.8. This was 7× higher than the mean of 95.1 for facial tracking. Additionally, participants were more precise with tilt-input with a mean star count of 19.7, compared to a mean of 1.9 using facial tracking. Although tilt-input was superior, participants praised facial tracking as challenging and innovative.**

Keywords—Android; Mobile Games; Facial Tracking; Tilt-input; Accelerometer; Sensors; Qualcomm Snapdragon

I. INTRODUCTION

With the rampant rise in smartphone usage, the gaming industry has adjusted to address to the convenience and appeal of the mobile context. The genesis of mobile gaming dates to the late 1990s when Nokia included a re-vamped version of the arcade game *Snake* on their mobile phones. It was a standard pre-loaded feature.[1] Present-day mobile games are, of course, far superior in terms of graphics and the sophistication of gameplay. Importantly, today's mobile games are available from download sites, such as the *App Store* on Apple's *iOS* or the *Play Store* on Google's *Android OS*.

Although game developers put considerable effort in advancing the state of the art in graphics and themes for gameplay, less interest is directed at the UI (user interface) of mobile games. The controls for mobile games on smartphones are limited as they heavily rely on touch-input. Most such games employ a soft touchpad for navigational input with some games relying on taps or touch gestures. A common UI scenario is thumb input using a soft d-pad on the left and soft buttons on the right, the latter to control in-game secondary actions. An example is Sega's *Sonic the Hedgehog 2*. See Fig. 1. Another method of recent interest is tilt-input using a mobile device's built-in accelerometer [5]. Some games, such as Imangi Studios' *Temple Run*, an infinite-runner game, effectively use tilt-input while also relying on touch-based input.

[1] http://en.wikipedia.org/wiki/Snake_(video_game)

Fig. 1. Sega's *Sonic the Hedgehog 2* illustrates a typical mobile UI, with soft touch zones as input controls.

One drawback of touch input for mobile games is the lack of tactile feedback and proprioception. While it is relatively easy to implement soft versions of the buttons on traditional handheld game consoles, doing so on a flat touch-sensing surface is problematic. The lack of a tactile sense can be mitigated (somewhat) using either auditory feedback or the device's vibrotactile actuator to create a "response" when a player presses, pushes, or slides on a soft control. However, interacting with a soft control lacks proprioception – the sense of relative position of body parts (e.g., fingers, limbs) combined with a sense of strength and effort in movement. Of course, physical controls inherently have location and shape and "push back" when the user actuates them. These properties bear valuable information to the player. So, there are significant challenges in implementing soft controls on touchscreens. However, the wide variety of sensors on modern mobile devices has opened new avenues to interaction, such as the aforementioned tilt-input using the device's accelerometer. Another sensor of potential interest for gaming is the device's built-in camera.

Tracking of a user's body or a body part is available in many applications today, both in mobile and non-mobile contexts. The most obvious non-mobile example is Microsoft's *Kinect*, which uses a camera and depth sensor to locate and respond to the user's position and movement in front of a console. Position and movement information allow the player to control the in-game environment using a natural

mapping from the player's movements to the movement of scene objects, such as an avatar [2]. Face tracking is also of interest, for example, to detect the number of faces in a scene. In a gaming context, facial tracking has been predominantly used to sense and augment a user's face, providing an experience known as augmented reality (AR). For example, a 2010 *Iron Man 2* promotional website released a web app demo that used a webcam to place an Iron Man helmet on the user. The demo translates and rotates the 3D helmet object based on the user's head movements.

In mobile gaming, the notion of facial tracking is fairly new. There are a few mobile apps which use the front-facing camera on mobile devices; however, games using facial tracking are scarce. One example is Umoove's *Flying Experience* for *iOS*, introduced in February 2014. The game uses the device's front-facing camera to track the user's head movements. Head movements are used to navigate a 3D environment.

In the next section, we review previous work in mobile gaming related to tilt-input or tracking a user's body or body parts. This is followed with a description of our methodology to empirically compare tilt-input and facial tracking as input methods for mobile gaming. The results are then presented and discussed with conclusions drawn.

A. Related Work

Wang et al. [10] performed a study seeking to enrich the experience of motion-detection games by employing a face tracking system with an *Xbox* gamepad. To test forms of hybrid control schemes, they tested conventional control with and without facial tracking added. Employing facial tracking added a new axis of in-game control. Participants performed better when facial tracking was added to the conventional physical control scheme. Additionally, the experiment observed that players felt more involved in the games they were playing and stimulated a higher emotional response when facial tracking was implemented versus without it. In contrast, our experiment compares tilt and facial tracking as individual input methods and separate control schemes.

Sko and Gardner [9] conducted a similar study that used head tracking in a first-person game environment using a webcam. The experiment employed natural head movements which mapped to four different gameplay controls: zooming, spinning, peering, and iron sighting. The experiment also implemented two ambient (or perceptual) techniques: head-coupled perspective (HCP), which mimics the parallax effect of a scene viewed from a window, and handy-cam, which replicates the shaky effect of a handheld camera. These ambient techniques have no effect in gameplay controls as they only enhance visual feedback. Sko and Gardner observed a general positive feedback with participants most receptive to peering. It was also noted that the participants experienced neck fatigue and pain when performing iron sighting, which required the user to tilt his/her head to the right. In addition, the HCP technique was ineffective due to the latency of the system.

Similarly, Francone and Nigay [4] performed a study on mobile devices using facial tracking to mimic user perception. By using the front-facing camera, a pseudo sense of depth and perception was rendered. The technique was found acceptable, with participants claiming the interaction was natural, attractive, and innovative. However, Francone and Nigay also noted that the camera's field of view was limited and therefore could not always capture the face of the user. Additionally, the small screen clipped regions of the rendered image and therefore compromised the perception of depth.

Zhu et al. [11] conducted an experiment comparing head tracking and gaze tracking as an input controller for a 3D soccer game. Similar to related work, the research observed that head tracking provided a more immersive experience and kept the participants engaged. It was also shown that head tracking was inferior to gaze tracking because head motion required more physical movement.

Alternatively, Chehimi and Coulton conducted an experiment dealing with a motion-controlled 3D mobile game [3]. In the experiment, they compared a mobile device's accelerometer and physical keypad as input to navigate around a 3D environment. Participants found the accelerometer control scheme encouraging and easier to use than the phone's physical keypad. The accelerometer-based input method was also observed to have an easier learning curve versus the phone keypad. The experiment showed that the tilt-based controls provided a more intuitive experience for both gamers and non-gamers.

Cairns et al. [2] compared touch and tilt in a study using a mobile racing game, *Beach Buggy Blitz*. Tilting produced higher levels of immersion and a better performance versus touch. These results were attributed to the natural mapping of tilt to a physical steering wheel. The steering metaphor provided players an easy introduction to the game's mechanics.

Browne and Anand [1] evaluated commonly used input methods for mobile games. Participants' preferences from most to least preferred were accelerometer (tilt-input), simulated buttons, and finally, touch gestures. Similar to Chehimi and Coulton's study, the participants found the accelerometer input method intuitive and engaging. The evaluation also showed that the reason participants gave a poorer rating to simulated buttons and touch gestures was the lack of tactile feedback. In the case of the simulated buttons, the fingers of the participants occluded the button. This resulted in a confusing sequence of actions.

B. Overview

Our user study compared two input methods: tilt-input and facial tracking. The study compared both input methods and observed player behaviour including learnability and efficiency with respect to speed and accuracy. The main goal was to assess the relatively new use of facial tracking as an input method for mobile gaming. Tilt-input, as a more common input method, was used as a point of comparison.

The study used the facial tracking and recognition API of the Qualcomm *Snapdragon* processor used in mobile game environments [7]. Participants were tested using an endless runner-styled game called *StarJelly*. The game utilizes the accelerometer sensor (tilt-input) and the front-facing camera of

Fig. 2. Google *Nexus 7 HD* (released Fall 2013).

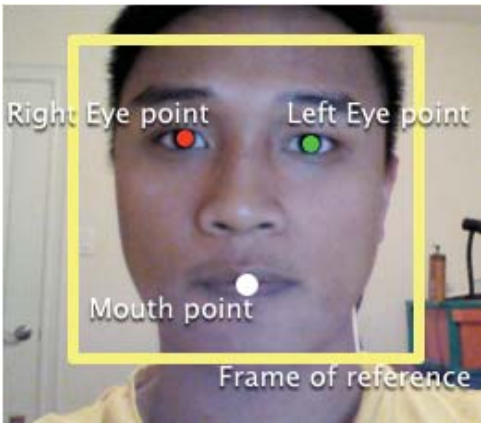

Fig. 3. Qualcomm Facial Tracking Demo App showing head-tracking capabilities.

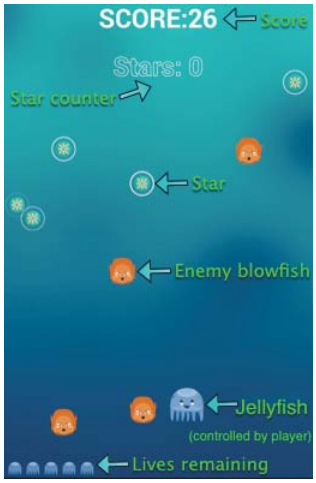

Fig. 4. StarJelly gameplay and user interface.

a *Nexus 7 HD* tablet (facial tracking). Each participant was given five lives per input method. The objective of the game was to survive as long as possible and to collect as many stars as possible. Together, these measures provided a total score per life iteration. See the Software subsection below.

II. Method

A. Participants

The experiment included 12 voluntary participants recruited from the local university campus. Ten participants were male, two female. Ages ranged from 19 to 23 years. All participants were smartphone or tablet owners with casual experience in mobile gaming. The participants were given no incentives or compensation.

B. Hardware and Software

The hardware was a Google *Nexus 7 HD* tablet running *Android 4.4 KitKat*. See Fig. 2. The device has a 7.02 inch display with resolution of 1920 × 1200 pixels and a density of 323 pixels/inch.

The software was developed in Java using the Android SDK, with special focus on the Qualcomm *Snapdragon* Facial Recognition API. See Fig. 3. The API is only available for devices with a built-in Qualcomm *Snapdragon* processor [7].

We developed a custom game called *StarJelly*. *StarJelly* is an endless-runner game set in an underwater environment. The game implements two game modes to accommodate the two input methods under investigation: tilt-input and facial tracking. The player navigates a Jellyfish avatar horizontally while blowfishes and stars advance vertically down the screen. See Fig. 4. The velocity of the blowfishes ranged from 413 to 1180 pixels per second downwards. On the other hand, the velocity of the stars ranged from 236 to 472 pixels per second downwards.

The goal was to avoid the blowfishes (to stay alive) and to collect stars (to collect star-points). Stars were collected via contact by moving the Jellyfish horizontally. More blowfishes were added as gameplay progressed; thus, the game difficulty increased, making a collision inevitable.

Each participant was given a total of ten lives: five for tilt-based gameplay and five for facial tracking gameplay. The player lost a life upon colliding with a blowfish. The score per life iteration was based on how long the participant survived without a blowfish collision as well as the number of stars collected.

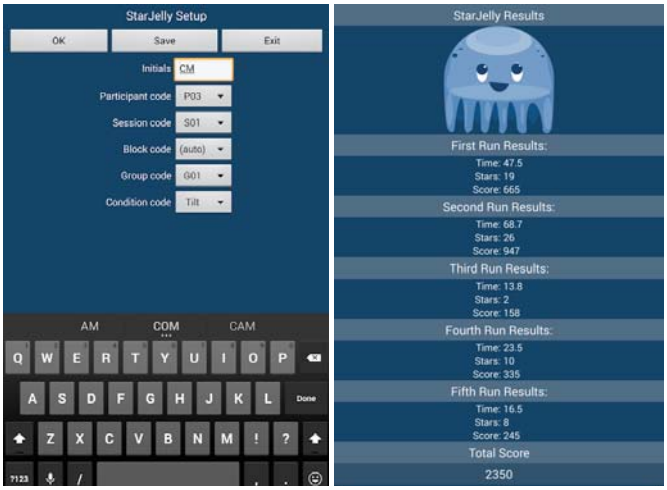

Fig. 5. a) Setup activity. b) Results activity.

The application began with a setup activity which prompted for the user's initials and parameters including a group code (for counterbalancing), input method, participant number, and session number. See Fig. 5a. When completed, the application transitioned into the game activity. The game mode depended on the input method specified in the setup activity. Each round, or "life", ended when the user collided with a blowfish. This triggered a ready state with a cool-down period followed by the next round. After five rounds, the application started an activity which presented results to the participant. The results included the score, stars collected, and survival time per round. The last

row in the results activity indicated the total score of the participant. See Fig. 5b.

In the tilt-based gameplay, the movement of the player was based on the angle of device tilt. Tilting was sensed about the *y*-axis of the device (aka *roll*) and mapped to the position of the Jellyfish. The mapping was 7 pixels per degree of tilt in the direction of tilt relative to the neutral position. On the other hand, facial tracking gameplay used the device's front-facing camera to calculate the *x*-coordinates of the left and right eye. These were averaged to yield an overall facial coordinate for movement control. The screen width of the game was 1200 px, with the home position in the middle. The player rotated his/her head left and right to move the Jellyfish left and right. The mapping was 13.3 pixels per degree of head rotation in the direction of rotation. Thus, the Jellyfish was positioned at the left edge of the screen with 45° left head rotation and at the right edge with 45° right head rotation.

The two game modes were similar in structure, with two minor differences. The first difference was in the cool down period when the player loses a life. In the tilt-input mode, the cool-down period was 3 seconds. This was ample time for the participants to prepare for the next round (life). In the facial tracking condition, the cool-down period was 10 seconds. This was necessary so participants could get a feel for the mapping of their face and could self-calibrate by moving the Jellyfish avatar around if necessary.

The other difference was a slight UI addition in the facial tracking condition. A green and red dot was rendered underneath the Jellyfish. See Fig. 6. The dot provided visual feedback on the player's face-to-game mapping. A missing dot indicated that the camera was not picking up the left or right eye coordinates of the participant. In addition, the dot allowed participants to easily navigate the avatar by providing a primary focal point. The game also utilized sound effects and upbeat background music.

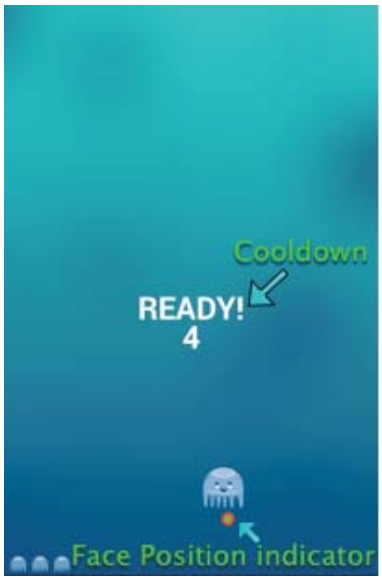

Fig. 6. The game in the ready state. For the facial tracking condition, a facial tracking dot is seen below the Jellyfish avatar.

C. Procedure

Participants were tested in a well-lit environment to provide adequate lighting for the front-facing camera. They were seated in front of a table and were instructed to keep their hands and forearms rested on the table during the facial tracking game mode. See Fig. 7. Participants were also given the option of lifting their arms off the table during the tilt game mode. Before starting the game, participants were briefly introduced to the objectives of the experiment and the goal of comparing tilt-input and facial tracking as input methods for mobile gaming. Participant were then asked to enter their initials in the setup activity as consent for participation in the experiment.

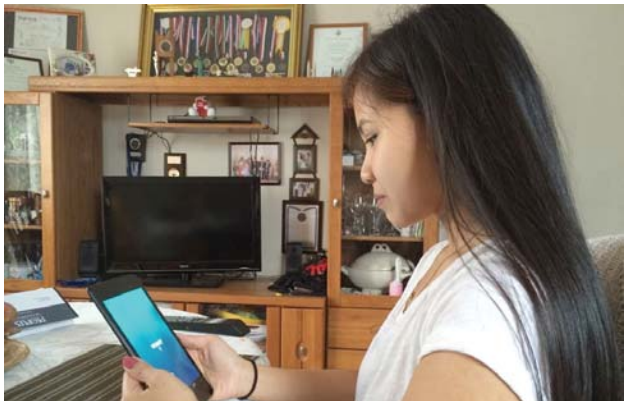

Fig. 7. Participant performing experiment.

Each game mode was briefly demonstrated. No practice trials were given. Each participant took about 10 minutes to complete the experiment. After the ten trials were completed, participants were given a questionnaire to solicit qualitative feedback. Items included participant experience, input method preference, and an open-end request for comments on the input methods.

D. Design

The experiment was a 2 × 5 within-subjects design. There were two independent variables: input method (tilt-input, facial tracking) and life (1, 2, 3, 4, 5). The dependent variables were survival time, stars collected, and score. Survival time was based on how long the participant survived each round. The score was an aggregate measure equal to the sum of 10× the survival time and 10× the number of stars collected.

Participants were divided into two groups to counterbalance the order of input methods, and thereby offset learning effects.

The total number of testing rounds was 120 (12 participants × 2 input methods × 5 lives).

III. Results

The effect of group (order of testing) was not statistically significant for survival time ($F_{1,10} = 1.312$, ns), stars collected ($F_{1,10} = 0.847$, ns), and score ($F_{1,10} = 1.234$, ns). Thus, we conclude that counterbalancing had the desired effect of offsetting learning effects due to the order of testing.

A. *Survival Time*

The grand mean for survival time for all 120 rounds was 27.2 seconds. The mean survival time travelled for tilt-input was 47.1 seconds. In contrast, the mean survival time for the facial tracking input method was 7.3 seconds. The mean survival time with tilt-input was therefore 6.5× longer than the survival time with facial tracking. Not surprisingly, the difference between the two input methods was statistically significant ($F_{1,10} = 65.74$, $p < .0001$).

The overall mean survival time by life shows that the facial tracking input method was inferior for navigating the Jellyfish avatar across the screen to avoid blowfish obstacles. The life iteration breakdown in Fig. 8 shows learning effects in both input methods.

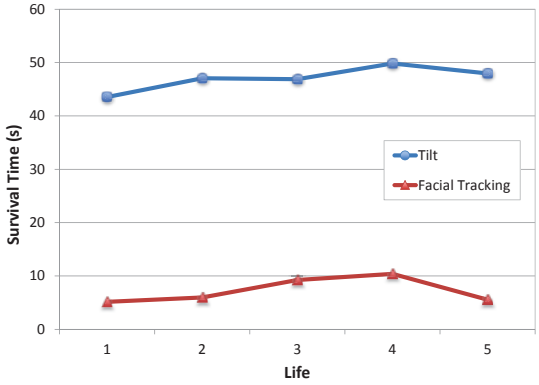

Fig. 8. Survival time per life iteration.

The longest survival time for the tilt-input method happened on the fourth life with a time of 49.9 s. On the other hand, the facial tracking input method reached a peak of 10.3 s, also on the fourth life. The highest recorded survival times were 84.1 s for the tilt input method and 34.9 s for facial tracking. As illustrated in Fig. 8, participants did learn and improve with each iteration. The survival time with the tilt-input method increased by 14% from the first iteration to the fourth iteration. In contrast, the survival time with facial tracking increased by 102% from the first iteration to the fourth iteration, indicating greater learning with the facial tracking input method. However, it was also illustrated that the mean survival time regressed back to an average of 56 s on the fifth iteration and is correlated with head fatigue, as discussed below. Due to the variability in the measured responses, the effect of life on input method for survival time was not statistically significant ($F_{1,4} = 0.143$, ns).

B. *Stars Collected*

The grand mean for stars collected for all 120 lives was 10.9 stars. The overall mean for stars collected with tilt-input was 19.5 stars. The mean with facial tracking was 2.3 stars. Thus, tilt-input averaged 9.5× more stars collected than with facial tracking. The difference was statistically significant ($F_{1,10} = 57.56$, $p < .0001$).

It was observed that tilt-input gave participants more precision and, therefore, allowed them to move around faster which resulted in more stars collected. It was also noted that participants were more interested in staying alive versus collecting more stars with the facial tracking game mode. The effect of life on input method for stars collected was not statistically significant, however ($F_{1,4} = 0.299$, ns). The breakdown is illustrated in Fig. 9.

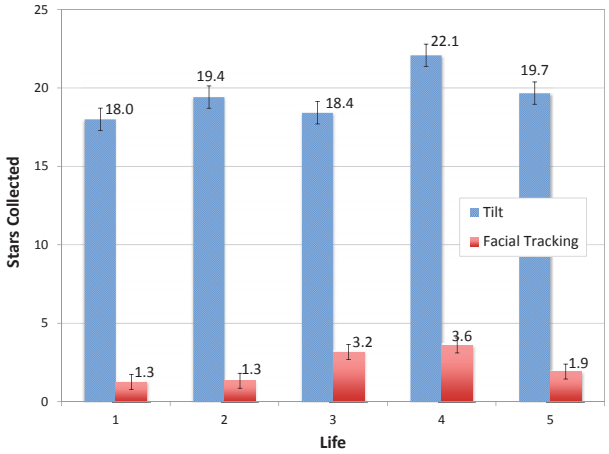

Fig. 9. Breakdown of stars collected per life based on input method. Error bars indicate ±1 *SD*.

C. *Score*

Score is an aggregate measure of survival time and the number of stars collected:

$$Score = (10 \times survival\ time) + (10 \times stars)$$

Based on the preceding results, it is evident that the scores attained with the tilt-input method would be much higher. Fig. 10 shows the substantial difference in the scores based on the two input methods.

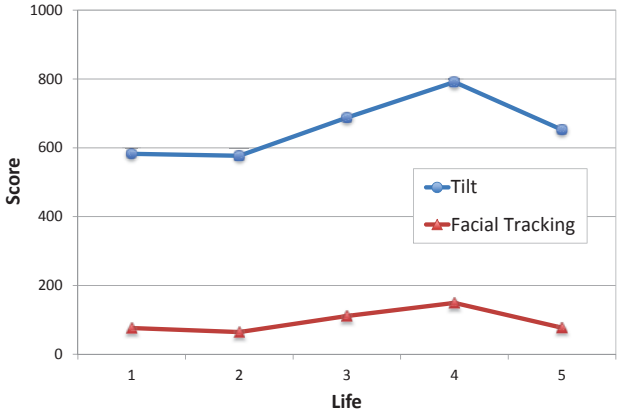

Fig. 10. Total score per life iteration.

The overall mean score for all 120 iterations was 376.9. The mean score for tilt-input was 665.8 while the mean score for facial tracking was 95.1. Hence, the mean score with tilt-input was 7× more than the mean score with facial tracking. The highest score achieved in the facial tracking mode was 479 while the lowest score was 12. In addition, the highest score

was 1101 with tilt-input while the lowest was 19. The difference in the scores between the facial tracking and tilt-input was statistically significant ($F_{1,10} = 66.49$, $p < .0001$).

The improvement in score with practice with tilt-input was 35% from the first life iteration to the fourth and 11% from the first to the last iteration. In contrast, the score improved by 96% with facial tracking from the first life iteration to the fourth. Due to variability in the measured responses, the effect of life iteration on score was not statistically significant ($F_{1,4} = 0.168$, ns). As mentioned previously with mean survival time, the increase depicts a learning effect with performance which could be amplified given more trials and rest periods.

Overall, it is clear that the facial tracking input method was inferior to the tilt input method. Several factors were involved in the large differences between the two input methods, including experience and hardware limitations. Hardware limitations are a known issue with tracking-based applications as noted by Sherstyuk and Treskunov in their head tracking evaluation [8]. The main issue is system lag, which is caused by the substantial CPU processing required to convert camera images to face-feature coordinates. For games, lag is a deal-breaker since game-inputs must be highly responsive.

With respect to lag, there was a large difference in latency between the two input methods. The average latency for each accelerometer motion event captured during the tilt game mode is 20 ms. In contrast, facial tracking events were captured every 125 ms and therefore yield a latency that is 6× greater than with tilt input. In addition, the facial tracking mode was susceptible to unexpected drops. That is, due to lighting conditions, swift head motions, or rotation of the head beyond a threshold, the facial tracker sometimes fails to detect a face in the camera's field of view. When the tracking is lost, the player avatar remains in the last known location until another tracking event is captured. These constraints were also observed in previous studies on facial or head tracking [6, 9, 10, 11]. Not surprisingly, these factors had adverse effects on performance.

As well, all the participants had at least some experience with mobile games using tilt-input. With this in mind, the participants were already skilled in tilt-based gaming. Conversely, the participants did not have experience with facial tracking input for mobile games.

D. Participant Feedback

Based on the questionnaire portion of the experiment, there was mixed feedback from the participants regarding the input method of preference. Surprisingly, half the participants preferred the facial tracking input method. One participant concisely described why he preferred facial tracking over tilt:

> *It's a lot more engaging because there isn't any other game with facial tracking. Tilting and swiping is a lot more common when it comes to mobile games and learning to control another bodily function such as head movement while playing a game is just another step further in game development. While the input method is a lot more challenging, it brings a different satisfaction once you finish playing.*

Another participant explained why he preferred tilt over facial tracking:

> *The tilt control scheme is smooth and responsive. It was easier to control. I found it much easier to get out of tight situations. This can be good when you're playing games that require fast timing.*

Participants also provided additional feedback based on both input methods. Many participants highlighted the fact that facial tracking, while difficult, provided an innovative way to play mobile games. Most of the participants believed that it just takes practice with facial tracking to obtain optimal results. Some participants highlighted that the latency of the camera was a key issue with the facial tracking input. This notion was also evident in previous work by Sko and Garder which addressed the inaccuracy and high-latency of head tracking [9]. Due to the limitation of the hardware, the facial processing was not up to speed with the frame rate of the game and this hindered participant performance. Some participants, however, used this disadvantage by timing their next movements. Many participants noted that the tilt input method was common in numerous games today and only provided a mediocre engagement level. In addition, the participant feedback shows that facial tracking was much harder in comparison to the tilt. One participant expressed his concern over the difficulty gap stating that the tilt input method was so easy that the game eventually got too repetitive.

Additionaly, four participants experienced slight head fatigue. Fatigue has been a common issue associated with tilting and rotation the head [9, 11]. This factor may have led to the decline in score, as previously noted. That is, after the 4th life iteration, the score and survival time was just as low as with the first iteration.

IV. Conclusion

Tilt-input and facial tracking were compared as input methods for mobile gaming. The experiment utilized a newly developed mobile app called *StarJelly*, an endless runner-styled game in which players avoid obstacles and collect stars. The mean scores were 665.8 for tilt-based input 95.1 for facial tracking. The difference was substantial: 7× higher for tilt-input. However, this is not a surprise as many of the participants had prior experience with tilt-based games.

Participants were also observed to collect more stars when playing the game using tilt-input. On the other hand, participants mainly focused on avoiding obstacles when playing the game using the facial tracking as input. Although the tilt-input method triumphed over facial tracking, many participants still preferred the facial tracking input and saw it as an innovative and exciting way to play mobile games.

Facial tracking is a natural interaction mode allowing players to interact using their face and a front-facing camera. However, the high processing requirements for head tracking provided adverse effects on the interaction. Two main factors affected the performance of the user during facial tracking: lack of experience and the latency of the camera. While facial tracking provides a new way to play mobile games, given the

hardware limitations [8], it is not adequate in a high-speed gaming environment at the present time.

V. Future Work

Our research is open to future work. One obvious extension is to test participants over longer and more intensive gaming sessions where learning can be tracked for more than five lives or rounds. In addition, better hardware could be used to address the latency issues associated with facial tracking. Future work could also implement facial tracking in slower-paced environments focusing on an immersive experience rather than performance. As observed in this study and previous work [6, 9], head tracking hurts performance in fast paced videogames due to issues of latency, inaccuracy, and fatigue.

References

[1] K. Browne and C. Anand, "An empirical evaluation of user interfaces for a mobile video game," *Entertainment Computing, 3*(1),1-10, 2012.

[2] P. Cairns, J. Li, W. Wang, A. I. Nordin, "The influence of controllers on immersion in mobile games," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems – CHI '14*, New York: ACM, 2014, pp. 371-380.

[3] F. Chehimi and P. Coulton, "Motion controlled mobile 3D multiplayer gaming," in *Proceedings of the 2008 International Conference on Advances in Computer Entertainment Technology – ACE '08*, New York: ACM, pp. 267-270.

[4] J. Francone and L. Nigay, "Using the user's point of view for interaction on mobile devices," in *Proceedings of the 23rd French Speaking Conference on Human-Computer Interaction – IHM '11*, New York: ACM, 2011, pp. 1-8.

[5] M. Joselli and E. Clua, "gRmobile: A framework for touch and accelerometer gesture recognition for mobile games," in *Proceedings of the 2009 VIII Brazilian Symposium on Games and Digital Entertainment – SBGAMES '09*, New York: IEEE, 2009, pp. 141-150.

[6] A. Kulshreshth and J. LaViola, "Evaluating performance benefits of head tracking in modern video games," in *Proceedings of the 1st Symposium on Spatial User Interaction – SUI '13*, New York: ACM, 2013, pp. 53-60.

[7] Qualcomm. Snapdragon SDK for Android, https://developer.qualcomm.com/mobile-development/add-advanced-features/snapdragon-sdk-android, 2014.

[8] A. Sherstyuk and A. Treskunov, "Head tracking for 3D games: technology evaluation using CryENGINE2 and faceAPI," in *Proceedings of 2013 IEEE Virtual Reality – VR '13*, New York: IEEE, 2013, pp. 67-68.

[9] T. Sko and H. Gardner, "Head tracking in first-person games: interaction using a web-camera," in *Proceedings of the 12th International Conference on Human-Computer Interaction – INTERACT '09*, Berlin: Springer, 2009, pp. 342-355.

[10] S. Wang, X. Xiong, Y. Xu, C. Wang, W. Zhang, X. Dai, *et al.*, "Face-tracking as an augmented input in video games: enhancing presence, role-playing and control," in *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems – CHI '06*, New York: ACM, 2006, pp. 1097-1106.

[11] D. Zhu, T. Gedeon, K. Taylor, "Head or gaze? Controlling remote camera for hands-busy tasks in teleoperation: a comparison," in *Proceedings of the 22nd Conference of the Computer-Human Interaction Special Interest Group of Australia – OZCHI '10*, New York: ACM, 2010, pp. 300-303.