

CHARACTERIZING THE EFFECTS OF LOCAL
LATENCY ON AIM PERFORMANCE IN FIRST
PERSON SHOOTERS

A Thesis Submitted to the
College of Graduate Studies and Research
in Partial Fulfillment of the Requirements
for the degree of Master of Science
in the Department of Computer Science
University of Saskatchewan
Saskatoon

By
Zenja Ivkovic

©Zenja Ivkovic, 2016. All rights reserved.

PERMISSION TO USE

In presenting this thesis in partial fulfilment of the requirements for a Postgraduate degree from the University of Saskatchewan, I agree that the Libraries of this University may make it freely available for inspection. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised my thesis work or, in their absence, by the Head of the Department or the Dean of the College in which my thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Saskatchewan in any scholarly use which may be made of any material in my thesis.

Requests for permission to copy or to make other use of material in this thesis in whole or part should be addressed to:

Head of the Department of Computer Science
176 Thorvaldson Building
110 Science Place
University of Saskatchewan
Saskatoon, Saskatchewan
Canada
S7N 5C9

ABSTRACT

Real-time games such as first-person shooters (FPS) are sensitive to even small amounts of lag. The effects of network latency have been studied, but less is known about local latency – that is, the lag caused by local sources such as input devices, displays, and the application. While local latency is important to gamers, we do not know how it affects aiming performance and whether we can reduce its negative effects. To explore these issues, we tested local latency in a variety of real-world gaming systems and carried out a controlled study focusing on targeting and tracking activities in an FPS game with varying degrees of local latency. In addition, we tested the ability of a lag compensation technique (based on aim assistance) to mitigate the negative effects. To motivate the need for these studies, we also examined how aim in FPS differs from pointing in standard 2D tasks, showing significant differences in performance metrics. Our studies found local latencies in the real-world range from 23 to 243 ms that cause significant and substantial degradation in performance (even for latencies as low as 41 ms). The studies also showed that our compensation technique worked well, reducing the problems caused by lag in the case of targeting, and removing the problem altogether in the case of tracking. Our work shows that local latency is a real and substantial problem – but game developers can mitigate the problem with appropriate compensation methods.

ACKNOWLEDGEMENTS

I would like to express great thankfulness to my supervisors Carl Gutwin and Ian Stavness for their great guidance and help throughout my graduate studies at the University of Saskatchewan. I thank Carl and the great people at the Interaction Lab for sparking my interest in academic research while I was a summer hire helping with others' research. I thank Jason Howard and my other friends for helping to motivate me to perform my studies in periods of downtime. I also thank my supervisors and family for their patience through the delays and downtime I encountered during my studies.

CONTENTS

Permission to Use	i
Abstract	ii
Acknowledgements	iii
Contents	iv
List of Tables	vii
List of Figures	viii
List of Abbreviations	x
1 Introduction	1
1.1 Problem	4
1.2 Solution	6
1.3 Steps in the Solution	8
1.4 Evaluation	10
1.5 Contribution	10
1.6 Outline	11
2 Background	13
2.1 Goal-Oriented Constrained Movement	13
2.1.1 Fitts' Law	13
2.1.2 Sub-movements and Feedback in Motor Control	16
2.1.3 Tracking	19
2.2 First Person Shooters and 3D	20
2.2.1 Fitts' Law in Virtual 3D	20
2.2.2 Aim Assistance	23
2.3 Latency	24
2.3.1 Network and Local Latency	25
2.3.2 Sources of Latency	26
2.3.3 Latency Compensation	29
2.3.4 Effects of Network Latency on Games	32
2.3.5 Effects of Local Latency	35
3 Comparison of Pointing in 2D versus 3D	38
3.1 Apparatus	38
3.2 Tasks and Procedure	40
3.2.1 Acquisition	40

3.2.2	Tracking	46
3.3	Differences Between 2D and 3D Conditions	47
3.4	Performance Metrics in Acquisition	48
3.5	Participants	50
3.6	Study Design	50
3.6.1	Experimental Conditions Summary	51
3.7	Results	52
3.7.1	Acquisition	52
3.7.2	Tracking	55
3.8	Discussion	57
3.8.1	Performance Differences Between 2D and 3D	59
3.8.2	Comparison with Previous 3D Studies	63
4	Latency in Real-World Systems	66
4.1	Methodology	66
4.1.1	System Selection	67
4.1.2	Apparatus and Measurements	67
4.2	Results	68
4.3	Discussion	69
4.3.1	Latency Summary	69
4.3.2	Attributing Latency to Components	70
5	Quantifying and Mitigating Local Latency	74
5.1	Controlling Local Latency	74
5.2	Tasks and Procedure	76
5.2.1	Acquisition	76
5.2.2	Tracking	79
5.2.3	Questionnaire	81
5.3	Compensating for Local Latency	82
5.4	Participants	85
5.5	Apparatus	85
5.5.1	Latency Measurement	86
5.6	Study Design	87
5.6.1	Latency Levels and Blocking	87
5.6.2	Training	89
5.6.3	Counterbalancing	89
5.6.4	Experimental Conditions Summary	89
5.6.5	Design and Hypotheses	90
5.7	Results	91
5.7.1	Pre-processing	91
5.7.2	Acquisition	91
5.7.3	Tracking	97
5.7.4	Survey Results	104
6	Discussion	109

6.1	Summary of Results	109
6.2	Explanation of Results	111
6.2.1	Effects of Latency	111
6.2.2	Compensation for Latency	113
6.3	Comparison to Similar Studies	114
6.4	Generalization to Real FPS Games	117
6.5	Implications of Results	119
6.6	Limitations	121
7	Conclusion	124
7.1	Questions for Further Study	125
7.1.1	Effects of Lag on Score in Real Games	125
7.1.2	Conflicting Results on Effects of Latency	125
7.1.3	Changes to Aim Mechanics Due to Latency	126
7.1.4	Index of Difficulty under Latency	127
	References	128
	A Consent Form	135
	B Index of Difficulty Calculations (2D and 3D)	137
	C Demographic Survey	139
	D Between-Block Survey (Non-mitigated)	142
	E Between-Block Survey (Mitigated)	143
	F End of Session Survey	144

LIST OF TABLES

3.1	Experimental PC specifications.	39
3.2	Distance and width values for all possible values of ID in the target acquisition task.	45
3.3	Target acquisition ID versus trial completion time regression results.	54
3.4	Summary of RM-ANOVA main effects and interactions on acquisition time for the acquisition task.	55
3.5	Summary of RM-ANOVA main effects and interactions for the tracking task.	57
3.6	Target acquisition ID versus trial completion time regression for both my study and Looser's.	64
4.1	Results of field latency sample, with latency shown in milliseconds.	73
5.1	Assistance strength at the experimental lag levels.	85
5.2	Experimental PC specifications.	87
5.3	Trial completion time increase due to latency.	92
5.4	Summary of RM-ANOVA main effects and interactions for the acquisition task.	99
5.5	Trial completion time increase due to latency.	100
5.6	Summary of RM-ANOVA main effects and interactions for the tracking task.	103

LIST OF FIGURES

1.1	A screenshot of Battlefield 4 (http://www.battlefield.com), showing the typical perspective of a first person shooter game.	3
2.1	Multi-directional tapping task, as described in ISO 9241-9, used for evaluating input devices.	15
2.2	Relationship between width and angular width, and distance and angular distance.	21
2.3	Actual object motion path versus the path seen by remote peers under different prediction error correction methods.	30
2.4	Taxonomy of player interactions, classified between precision and timeliness requirements. Adopted from [24].	33
2.5	Player performance versus latency magnitude in three types of games. Adopted from [24].	34
3.1	Side cut-out view of the world. The large room shown on the left is used for target acquisition, while the area on the right (tracking target shown) is used for tracking. The player location is fixed in the middle.	41
3.2	2D Acquisition task showing a sample of various target distances, widths, and directions.	42
3.3	Target tracking task showing target movement from initial position (T0) through four direction changes (T1-4).	46
3.4	Cropped view of the tracking task in the 2D environment (top) and the 3D FPS environment (bottom).	48
3.5	Cropped view of the acquisition task showing a sequence of frames from one trial (left to right) in the 2D environment (top) and the 3D FPS environment (bottom).	49
3.6	Index of difficulty versus trial completion time.	53
3.7	Target speed versus time on target.	56
4.1	Scatter plot of latency results with a distinction between PC and console systems.	69
5.1	Target acquisition task showing all possible target locations and sizes. The six IDs (numeric labels) were chosen to be typical of targets in FPS games.	78
5.2	Target tracking task showing target movement from initial position (T0) through four direction changes (T1-4).	80
5.3	Latency levels over the course of a session, grouped into blocks.	88
5.4	Effect of latency on targeting performance.	93
5.5	Effect of latency on targeting performance by ID and compensation.	95
5.6	Effect of difficulty on targeting performance by latency and compensation.	96
5.7	Effect of latency on error rate in target acquisition.	98
5.8	Effect of latency on tracking performance.	101
5.9	Effect of latency on tracking performance, based on target speed.	102

5.10	Summary of between-latency survey results.	105
5.11	Summary of post-first session survey results.	107
5.12	Summary of post-second session survey results.	108
6.1	Comparison of results of my study versus MacKenzie. Left side shows performance relative to baseline, while right side shows absolute movement time. .	115

LIST OF ABBREVIATIONS

2D	Two-dimensional
3D	Three-dimensional
AI	Artificial Intelligence
ANOVA	Analysis of Variance (statistical method)
BSP	Binary Space Partitioning (3D acceleration structure)
CRT	Cathode Ray Tube(type of display)
FPS	First Person Shooter
GPU	Graphics Processing Unit
ID	Index of Difficulty [33]
Lag	Latency
LCD	Liquid Crystal Display
MMORPG	Massively Multiplayer Online Role Playing Game
MT	Movement Time
RM-ANOVA	Repeated Measures Analysis of Variance (statistical method)
SD	Standard Deviation
TP	Throughput [42]
V-sync	Vertical synchronization (for display output)

CHAPTER 1

INTRODUCTION

First person shooter (FPS) games are a popular video game genre with millions of active players worldwide [9]. The genre is characterized by games in which the player takes a *first-person perspective* within the game avatar that is being controlled; that is, the view displayed on the screen originates within the avatar’s head, and the view direction is directly tied to the direction that the avatar faces. Another important characteristic of the genre is that players possess weapons that they use to fire at enemy targets (which can be controlled by other players or AI), and the game objective typically involves eliminating the targets either as a direct or intermediate goal. For example, a common objective in multiplayer FPS games is to obtain the highest score, with score being gained by killing enemy players, and an objective more common to single-player or cooperative FPS games is to complete story-based tasks, which normally involves eliminating AI-controlled opposition along the way.

FPS games tend to be a serious and competitive genre, with relatively few games marketed for casual play. For example, FPS games are often featured in professional game competitions such as Major League Gaming.¹ Refinement of FPS game mechanics and gameplay elements can have a significant effect on player performance and enjoyment, particularly when games are targeted toward competitive play. An understanding of low-level interaction details in FPS gameplay is therefore an important step toward improving FPS game design and mechanics.

FPS games can be classified into several broad categories. Recent FPS games tend to use elements of realistic warfare, with player avatars and weapons being modeled to make use of realistic mechanics. Avatars are relatively easy to destroy, typically requiring only one

¹<http://www.majorleaguegaming.com>

to five hits with a rapid-fire weapon, and therefore quick and accurate initial reactions to stimuli, such as encountering an enemy, are important. The game weapons are modeled after real-life weapons, and include design elements such as limited magazines of ammunition that must be periodically reloaded. Weapons also typically have recoil when fired, and there is some degree of spread (i.e., random deflection of bullet direction) applied to successive shots rather than weapons having perfect accuracy with respect to aim direction. Two of the most popular franchises of games of this type are the Call of Duty² and Battlefield³ (Figure 1.1) series.

Another common style of FPS is that in which game design makes little attempt to achieve realism – in these games, the overall pace of the game is generally quicker. Avatars move faster and can absorb more weapon hits before being killed, and weapons often have little to no recoil or inaccuracy. Continuous fire beam-like weapons that require precisely tracking the target with the aiming reticle (i.e., crosshair at the center of the screen) are common. Prevalent games of this type are the Quake⁴ and Team Fortress⁵ series.

Additionally, some games, while not being first-person shooters, behave similarly to FPS games in regards to pointing. For example, the Grand Theft Auto series⁶ gives the player a third-person view external to the avatar, but the mechanism of shooting at targets behaves essentially the same as in first person shooters. The game’s visual appearance is similar to that of many shooters, and the targeting reticle stays centered with respect to the screen. In such games, weapon design and target behavior is typically no different than what is found in first person shooters. However, players are sometimes given the option to use aiming assists to help them in acquiring targets.

A critical aspect shared by almost all FPS games is that of aiming. Aiming actions can generally be separated into two types of tasks: target acquisition, and target tracking. Target acquisition is the act of moving the aiming reticle onto the target and pressing a button to fire at it, while tracking is keeping the reticle on the target as it or the player moves around.

²<http://www.callofduty.com>

³<http://www.battlefield.com>

⁴<http://www.quakelive.com>

⁵<http://www.teamfortress.com>

⁶<http://www.rockstargames.com/grandtheftauto>



Figure 1.1: A screenshot of Battlefield 4 (<http://www.battlefield.com>), showing the typical perspective of a first person shooter game.

While these two tasks are often performed consecutively in practice (i.e., acquire a target, then track it while continuously firing at it), it is useful to distinguish the two parts and study them separately since there may be factors that interact with one task differently than the other. A player’s overall performance, such as number of kills made during a match, depends heavily on their ability to perform these tasks. Therefore, improving our understanding of the low-level details involved with target acquisition and target tracking in FPS game environments can inform the design of game elements that have an effect on the player’s experience.

Real-time games require quick and timely responses to other players’ actions and game events [85]. Any delays in the responsiveness of a game are disruptive because they create mismatches in timing between a player’s actions and visual feedback of those actions, opponents’ actions, and other game events. Latency (or lag) in FPS games has been an important problem since the beginning of the genre because it causes detrimental effects on important aspects of gameplay such as aiming. Even low levels of latency caused by slow game update rates have been a problem since the early FPS games such as Wolfenstein 3D and Doom by

iD software.⁷

Once it became common to play multiplayer games through the Internet or direct modem connections, the poor network connectivity that was common at the time resulted in high amounts of latency in the of 150 to 300 ms. Latency arising from network communication has been extensively studied in a number of contexts, and delays above 100 ms have been shown to be disruptive to racing games [68] and first person shooters [13]. Since latency originating from the network has highly disruptive effects on gameplay, game developers quickly began developing techniques to mitigate network latency. Combined with these mitigation techniques and significant improvements in network speed and reliability in recent years, latency arising from the network has had diminishing effects on gameplay.

However, one important area that has not been examined within the context of 3D first person shooter games is that of the effects of *local latency* – that is, latency caused by phenomena on the local computer, rather than due to networking. Local latency arises from input devices, displays, and software processing. Latency on the game output subsystem (e.g., computer monitor, television, or graphics drivers) forces players to react based on old information, and input latency adds to the time taken for player actions to be registered by a game. In both situations, player performance suffers, either because of a missed opportunity, incorrect response to an opponent’s action, or reduced ability for the player to skillfully control their movement. As a further drawback, local latency causes reduced player performance and enjoyment even when it may not be noticeable [44]. Whereas the magnitude of network lag is generally decreasing, local latency is often increasing due to the use of wireless input devices and high-latency displays such as televisions. My measurements suggest that local latency in these situations is often above 100 ms (see Chapter 4).

1.1 Problem

It is known that latency is a problem in any interactive tightly-coupled tasks, but there is a lack of knowledge regarding precisely how local latency affects aiming performance in first person shooters. Without quantified knowledge of the precise effects of latency on 3D FPS

⁷<http://www.idsoftware.com>

gameplay, it is difficult to judge how severely lag affects different game tasks, to develop effective compensation techniques, to motivate consumers to consider latency when making game system purchases, and to motivate hardware and software developers to minimize local latency (or susceptibility to its effects) in their products.

There is a varied but generally poor understanding of the prevalence and effects of local latency among video game communities. Players involved in professional competition or other forms of serious gaming tend to be more aware of the problem, but others are less informed. Many enthusiasts and casual gamers do not realize that the gaming systems they use suffer from latency even during offline play. While many others have some level of awareness regarding local latency, its negative effects are often dismissed as insignificant or unimportant at levels that are typically not perceptible (e.g., below 100 ms). Arguably, one of the major causes of this lack of awareness is the absence of specific information regarding the prevalence and effects of local latency. If local latency was studied in detail and quantified within the appropriate context (i.e., the relevant game genre – in this case, first person shooters), players would have the information needed to make appropriate decisions to help maximize their level of performance and enjoyment in games.

Previous research has shown that network-based latency has a significant negative effect on player gameplay performance [22, 24, 7, 20]. However, local latency has different effects on gameplay than network latency, such as not affecting the local view feedback that is shown to the player (see Section 2.3.1). Likewise, the effects of local latency have been studied for pointing in 2D environments [61, 72, 71], but not in 3D FPS games, and it should not be assumed that the effects are equivalent due to important differences that make FPS games more sensitive to latency than many other games or tasks [22].

In order to determine the effects of latency on aiming in first person shooter games, it is necessary to have an understanding of aiming in such games. A study of latency in pointing should be framed within the models of pointing. Existing research about aiming in 3D such as Vicencio [89] tends to use existing models of pointing in 2D contexts and assumes that such models are also valid within similar 3D contexts. However, there are several differences between 2D and 3D settings that could alter the performance of aiming – in particular, there are substantial changes in the visual information presented in the two environments, and

therefore FPS games require for immediate feedback when controlling the view [24].

One major difference is that in FPS games, the game world visually pans around a fixed central reticle, while 2D environments typically consist of a fixed background with the player directly moving the cursor to change their pointing target. Substantially more optical flow is presented to the user while panning in a FPS environment as compared to moving the cursor in a 2D application. In addition to optical flow, FPS environments typically include other 3D visual cues, such as perspective, parallax, shading, shadows, and sometimes stereopsis that are absent in 2D graphics. These differences result in a substantial increase in the amount of visual information that needs to be processed by players, which tends to slow down feedback processing time [35]. Looser [54] reported that target acquisition in 3D FPS environments is approximately 25% slower than in 2D scenarios. This difference in performance suggests the effects of latency in 2D and FPS environments may also differ.

Despite these numerous differences, relatively few studies have examined aiming in 3D FPS games, although a significant amount of previous work has been performed that examines target acquisition and tracking in the context of a 2D environment. For example, a large body of research has been performed around the classic Fitts' Law [33] limb movement time model. Similarly, studies have been performed related to 2D-based target tracking, such as the modeling of trajectory-based tasks with the Steering Law [3]. Since the 3D FPS environment and pointing behavior is quite different than pointing in similar 2D tasks, it should not be assumed that previous 2D-based research applies equivalently to 3D first person shooters.

1.2 Solution

The effects of local latency on first person shooter aiming performance have not been previously studied, and quantifying these effects is important for improving player gameplay experience. Additionally, there are no established in-game mechanisms for mitigating the effects of local latency, and there is little knowledge regarding the range of local latency that may be found in gaming systems. Furthermore, differences between aiming in 3D FPS and 2D environments have not been sufficiently established, and characterizing these differences

is helpful for motivating the study of phenomena such as latency specifically within FPS environments. In order to solve these problems, I performed three studies.

In study one (Chapter 3), I sought to investigate and quantify the difference in performance between standard FPS aiming tasks and equivalent pointing tasks in a 2D environment. I tested both target acquisition and target tracking. I also created two different controlled experimental environments: one 3D environment that emulates an FPS game, and a similar environment in 2D. Specifically, in the 2D environment, the mouse directly controlled the aim reticle against a fixed background to perform aim. The difficulty of aiming tasks was controlled and consistent between the 2D and 3D conditions. Player performance in each condition was modeled and compared, and I quantified the differences in task completion times for both target acquisition and target tracking.

Before studying the effects of local latency in FPS games, I performed study two (Chapter 4) with the goal of ascertaining an approximate distribution of local latency across various types of gaming systems. I selected existing real-world systems that covered a range of input, output, processing devices, and games, and I measured the latency of each setup using a measurement methodology that I devised. This study determined an appropriate range of latency values to examine in the following study, as well as demonstrating that real-world latencies are high enough to cause problems.

Finally, study three was performed to examine the effects of local latency on aiming performance in first person shooters. The objective was two-fold: to quantify the negative performance effects of latency, and to find a mitigation strategy to minimize the effects. As in study one, aim was separated into target acquisition and target tracking tasks. Participants performed the same tasks at each of five levels of local latency. I compared task completion time within each level of latency to a base-case condition with minimum latency, thus quantifying the effect of latency and improving our knowledge of it.

As the second part of study three, I also created and evaluated a technique to mitigate the effects of local latency through the use of aiming assistance. Participants performed the same tasks in the same manner as in the first part of the study, but with the presence of the mitigation techniques (i.e., aim assists). The strength of the aim assists was based on the current latency level set in the experiment, with the goal of decreasing or eliminating the

differences in performance found between different levels of latency.

These studies improve the body of knowledge regarding differences between 2D and 3D FPS environments, particularly when it comes to the effect of local latency. I demonstrate that local latency is pervasive and substantial in magnitude in gaming systems, and because aiming in 3D first person shooters has important differences to pointing in 2D settings that may make studies of latency in 2D to be inapplicable to FPS, I quantified the effects of latency on aiming in FPS games and also demonstrated effective techniques to compensate for the latency.

1.3 Steps in the Solution

The process of improving the understanding of local latency in first person shooter games involved a number of steps:

1. It was necessary to determine how aiming in first person shooters is performed, starting with identifying the components of aim. I found that aiming can be split into target acquisition and target tracking as two related but separate components of aiming.
2. An initial pilot study was performed to informally examine the effects of local latency on target tracking in a simple 3D scenario.
3. I determined useful constraints in acquisition and tracking behavior for FPS games based on typical target behavior and commonly encountered aiming patterns. For acquisition, it is important to evaluate targets that could appear anywhere visible on the screen, with both vertical and horizontal deflection required to aim at them. For tracking, it is sufficient to have targets move only horizontally because this is the primary style of movement seen in FPS.
4. I needed to find a model of target acquisition that is valid in FPS games. Since Fitts' Law [33] is the canonical model for pointing in 2D, I picked it as the starting point. Study one was performed to compare target acquisition and tracking performance in 2D versus 3D environments. I determined that Fitts' Law is valid in 3D as well as 2D,

except that performance constants differed, with 3D tasks taking longer to complete in the equivalent conditions.

5. I identified the components that contribute to local latency in gaming systems. This process involved reading about such contributing components from existing literature and experimenting with factors that may affect latency, combined with the author's knowledge of computer hardware, software, and operating systems.
6. I determined a procedure for measuring the total local latency of a computer gaming system. It was important to be able to test systems quickly and easily, so that a number of existing real-world systems could be measured.
7. I measured the local latency of a representative sample of real-world systems in study two based on my knowledge of what contributes to latency. The measurements were an important step toward motivating the significance of the local latency problem, as well as determining a range of latency values to use for evaluating the effects of local latency.
8. I identified the ways in which latency affects aiming tasks. This was done based on observing how aiming was affected by latency in the previous pilot study, and also through the author's experience with latency in FPS games.
9. I created a latency mitigation technique based on existing aiming assistance techniques. The types of assistance needed and the shape of the assist functions were identified based on knowledge gained through the previous step. The constants for the assist functions were tuned by performing a pilot study that employed the aiming assistance in the presence of varying latency.
10. I performed study three to quantify the effects of local latency on FPS aiming performance. The experimental procedure was very similar to that in study one, but with the presence of varying amounts of latency.
11. Also as part of study three, I evaluated the effectiveness of my latency-mitigating aim assistance techniques, and had participants answer questionnaires regarding their

subjective experience with the latency and aiming assistance.

1.4 Evaluation

There were two main evaluative components to the concepts described in this thesis: study one (Chapter 3), which determined the effects of dimensionality (i.e., 2D versus 3D environments) on aiming performance, and study three (Chapter 5), which determined the effects of local latency on FPS aiming as well as the effectiveness of aim assistance toward reducing the effects of latency. Both studies made use of controlled experiments in which human subjects performed tasks in game-like applications that I created to obtain performance metrics in varying conditions. Study two was a collection of descriptive statistics.

1.5 Contribution

My work is the first to investigate and improve the body of knowledge regarding the effects of latency in first person shooter games. My primary contribution is the result of my experiments in Chapter 5, which shows the degradation in aiming performance caused by local latency and mitigation techniques that can be used to decrease or eliminate the effects of local latency. I provide the first empirical evidence that local latency is a real and substantial problem for FPS games – even at very low levels, and levels lower than were measured in many real-world systems. My work shows that designers and publishers of FPS games (and other high-speed interactive systems) can benefit from compensating for local latency.

The secondary contributions of this thesis are comparisons of aiming between 2D and 3D environments in Chapter 3 that are otherwise equivalent, and the finding of an approximate distribution of local latency and its causes in real-world systems (Chapter 4). This work is the first to show that target acquisition performance is different in 3D compared to 2D, for matched-difficulty targets.

The knowledge gained from my experiments can help improve first person shooter game design and experience, in the following ways:

Game and Hardware Design. An understanding of latency and its effects can motivate

consumers, manufacturers, and game designers to decrease the prevalence of latency or limit its negative effects (see Section 6.5).

Level design. A better understanding of ideal target sizes and their relationship to map size and layout can improve the design of worlds and levels with particular difficulty characteristics.

Game balancing techniques. A number of game balancing techniques are currently used in both 2D and 3D games in order to dynamically adjust difficulty levels or to allow players with different skill levels to play together. Correct implementation of game balancing techniques such as aiming assists [62] requires a thorough understanding of aiming in the game environments.

Training of skilled FPS players. The performance and mechanics of aiming techniques can be an important part of training for expert gamers. Much like the way in which training for athletics is informed by detailed understanding of the low-level actions of body movements from studies in exercise physiology, training regimes for FPS gamers could be devised based on a deep understanding of the low-level actions in FPS game tasks from studies in human computer interaction.

1.6 Outline

The following is an outline of the remaining parts of this thesis:

Chapter Two shows a look at the research that has been previously performed that is related to the topics of this thesis. Topic areas include goal-oriented constrained limb movement (e.g., models of pointing temporal performance and mechanisms of execution), including the applications of such to first person shooter games. The different types of latency and their sources are also examined, and some research regarding their effects on performance is reported.

Chapter Three presents my study one, which looks at the differences in performance between 2D and 3D environments in target acquisition and target tracking tasks using a

human study. The potential reasons for these differences are discussed as well.

Chapter Four shows my findings regarding the distribution of local latency that can be found in real-world gaming systems (study two). I also look at how changing some specific factors can affect latency in systems that are otherwise equal.

Chapter Five presents study three, which answers the main questions that I asked about latency. Using results from a human study, I examine how different levels of local latency affect target acquisition and target tracking performance in a controlled FPS mini-game, and I also report on the effectiveness of using aim assistance to compensate for the effects of latency.

Chapter Six is a discussion of the results of this thesis. Results are summarized, and possible reasons are given for the findings. The findings are compared to other similar studies and their generalizability is discussed. The implications and limitations of my research are also presented.

Chapter Seven presents a brief conclusion that summarizes the main findings of the thesis, and also discusses potential areas of future investigation.

CHAPTER 2

BACKGROUND

In this chapter, I present the background information needed to understand my research while we examine the previous work that has been performed on related topics. Since my work combines the topic areas of pointing and motor control, first person shooters, and latency, I divide this chapter into those three parts.

2.1 Goal-Oriented Constrained Movement

This thesis focuses heavily on aiming at targets in virtual environments, which involves fine motor control. This type of task fits well under the extensive literature that has been written on goal-oriented constrained limb movement (e.g., pointing in the real world or in virtual environments, reaching for objects, moving objects to constrained areas, etc). Although much of this previous work has been targeted at real-world physical interactions rather than computer-based on-screen virtual interactions, the results are generally accepted to apply to both [56, 84].

2.1.1 Fitts' Law

Fitts' Law [33] is the most widely accepted relationship for predicting performance of constrained motor movement tasks. The original work by Fitts, published in 1954, is based on three types of experiments with physical object manipulation: a reciprocal tapping task, a disc transfer task, and a pin transfer task. It has since been extensively used in computer science research to model and predict the amount of time taken to perform a pointing task in relation to the difficulty of the task. The model is based on information theory; in this case, distance to the target is said to be equivalent to the signal amplitude, and the width of the

target is similar to noise amplitude (i.e., a tolerance). Fitts proposed a metric to quantify the difficulty of a task, named *Index of Difficulty* (ID), measured in bits:

$$I_d = -\log_2 \left(\frac{W_s}{2A} \right)$$

where W_s is the tolerance range (width) and A is the amplitude of the movement (distance).

Fitts also proposed an *Index of Performance* (IP) that indicates motor performance in a way that is constant across a wide range of amplitudes and tolerances, measured in bits per second:

$$I_p = -\frac{1}{t} \log_2 \left(\frac{W_s}{2A} \right)$$

where t is the time per movement.

Although Fitts' Law was not originally designed to model computer-based pointing, it has become the standard basis used in human-computer interaction research for target acquisition with a wide variety of input devices. Over the years, several improved variations to Fitts' original model have been created. The formulation suggested for use today [84] is based on Shannon's model for information capacity of a communications channel [82], which was the basis for Fitts' work. This modern formulation for Index of Difficulty, which is better at predicting small-scale movements, and proposed by MacKenzie [55, 56] is commonly called the Shannon formulation:

$$ID = \log_2 \left(\frac{A}{W} + 1 \right)$$

It has become standard practice to use the models for ID and IP to perform regression in order to predict movement times or test for goodness-of-fit. This usage yields the equation commonly referred to as Fitts' Law [56]:

$$MT = a + b \cdot ID$$

where a is a constant reaction time cost and b is the relative performance of the user and device pairing.

This formulation has become the basis of an ISO standard, ISO 9241-9, which is used to evaluate non-keyboard input devices [42]. The standard presents a multi-directional tapping experimental task to be used for evaluation, consisting of a circular arrangement of circular

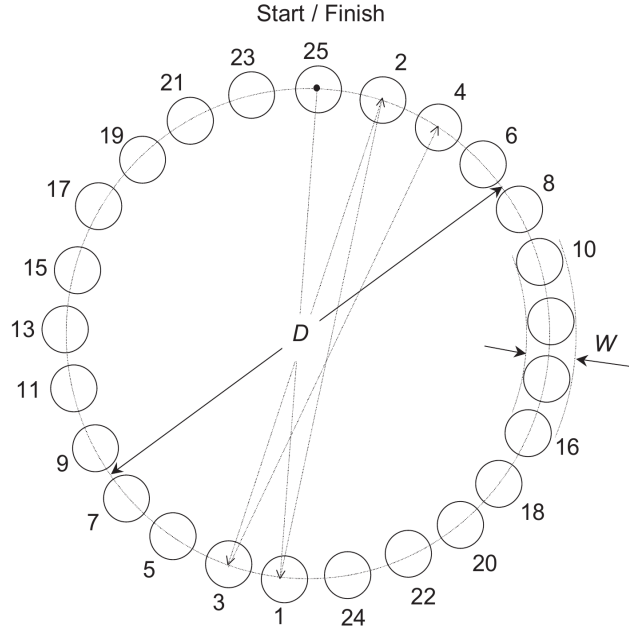


Figure 2.1: Multi-directional tapping task, as described in ISO 9241-9, used for evaluating input devices.

targets (Figure 2.1), appearing in many different directions around a central point (due to the fact that targeting performance varies based on direction of approach [91]).

The standard recommends using a measure which it names *Throughput* (TP) to evaluate the bandwidth of devices and conditions. Throughput is formulated as:

$$TP = \frac{ID_e}{MT}$$

where MT is the time to complete the targeting condition (or movement time), and ID_e is the Index of Difficulty adjusted for accuracy – an idea proposed by Crossman in 1956 (see [90]) to better reflect the task that participants performed rather than what they were asked to do. ID_e is calculated as in the Shannon formulation for ID, but with width being adjusted as follows:

$$W_e = \begin{cases} W \cdot \frac{2.066}{z(1 - Err/2)} & \text{if } Err > 0.0049\%, \\ W \cdot 0.5089 & \text{otherwise} \end{cases}$$

where W is the target width, Err is the error rate, and z is the normal distribution quantile function (*qnorm* in environments such as the R language). Another way to calculate W_e (omitted here) makes use of the standard deviation of click distances away from the center

of the target. An important benefit of using Throughput is that it is immune to the speed-accuracy tradeoff [58]; that is, Throughput is constant regardless of whether users focus more on speed or accuracy.

2.1.2 Sub-movements and Feedback in Motor Control

Although Fitts' Law has been repeatedly demonstrated to predict the overall time taken to arrive at a target, it does not describe the underlying process of movement. There is a consensus that most movements that are consistent with Fitts' Law are comprised of multiple submovements [28, 25, 64], but the nature of these submovements and the feedback control involved is less clear. Here, I present a brief summary of several different sub-movement models; for an in-depth coverage, see [28].

One of the earliest well-supported models of limb control is the *iterative correction model*, originally proposed by Crossman [25] and later refined by Keele [46]. The iterative model suggests that submovements occur as consecutive ballistic impulses that are prepared by visual feedback from the previous impulse. Each impulse travels a constant proportion of the remaining movement distance and takes approximately a constant amount of time to complete, with total acquisition time being dependent on the number of impulses performed. Crossman estimates that visual feedback reaction time is much quicker than the normal accepted values for reaction time, with times of 20 to 50 ms being proposed, and hypothesizes that the short times are due to the time windows for feedback being anticipated (since the start and stop times of limb motion are known to the participant).

Although there are several other models to describe limb aiming movement, the currently accepted one is the *optimized submovement model* [28], which was originally developed by Meyer [64]. This model was created in order to explain the deficiencies of the iterative correction model: the iterative model predicts a constant initial submovement duration, even though the duration has been observed to vary based on distance and width, and the iterative model predicts each movement to cover a constant proportion of the remaining distance, despite variability of submovement endpoints being reported by many [28]. The optimized submovement model predicts that movement to a target is composed of two submovements, regardless of target distance and width: an initial primary movement that is programmed

to end at the target, and an optional secondary corrective movement. The secondary movement only occurs if the primary movement misses, which occurs due to perturbations caused by neuromotor stochastic noise. The duration of each movement is determined by subjects adapting to optimize the total movement time as required due to neuromotor noise [64]. Meyer also tested the optimized submovement model under a condition with no visual feedback, and the model was successful in this case as well. With no visual feedback, the primary movement was relatively unaffected, but the duration and number of submovements of the secondary movement were increased.

The initial ballistic submovement of an aiming task is performed with the goal being to minimize the time needed to correct for the error resulting from the movement [38]. Intuitively, it may seem like the error distribution from the initial movement would be centered on the target, with equal amounts of undershoot and overshoot – models such as the optimized submovement model would also predict this outcome [64]. However, only a small portion of initial movements results in overshoot in practice [30]. This is because the time needed to correct for an overshoot is greater than for undershoot [29], since overshoot requires both overcoming the inertia of the initial movement, and a swap of muscle roles, with antagonist muscles taking agonist roles and vice versa.

There are conflicting findings regarding which portion of the aiming movement is affected by visual feedback. One study [12] shows that removing vision of the hand and target 290 ms before the movement finishes has an insignificant effect on performance, presumably because there is insufficient remaining time for visual feedback to have an effect. Other studies [17, 21] indicate the opposite – only visual information from the final portion of the aiming movement (25% or 135 ms in the case of [17]) is important. Despite the presence of such studies that discount the importance of visual feedback during certain phases of movement, the body of research as a whole indicates that vision has an important effect both during the ballistic and corrective portions of movement [28]. For the ballistic portion, vision mostly provides feed-forward information to plan the movement, while vision guides the corrective portion in an on-line feedback fashion. There is also some evidence that visual feedback is used throughout the entire motion: Saunders [77] shows that for fast reaching movements in a virtual space, participants reacted to perturbations to their virtual fingertip movement

approximately 160 ms after the event, even when the perturbation happens early in the movement and despite the subjects being unaware of it.

It would intuitively seem that a sufficiently practiced movement can eventually be performed without on-line feedback, and some older research does suggest that this is the case when movement is practiced [46, 80], or when movement times are below approximately half of a second in length [94]. However, more recent research contradicts these findings, suggesting that visual feedback is always involved. Proteau [74] had participants practice an aiming task either 200 or 2000 times, with some participants having full visual feedback available while others could not see their arm. Results showed that the participants with more practice were actually more greatly affected by the removal of feedback, and thus were more dependent on it. One possible explanation shows that increased practice leads to participants making more efficient use of feedback rather than changing the ballistic portion of movement or reducing the number of submovements [1]. Similarly, Elliot shows that people adjust their initial ballistic movement by increasing its speed in order to make more time for corrective actions near the end of the movement [31].

However, there is evidence that, in some conditions, the entire aiming movement can be made in open-loop with no feedback and with just one ballistic motion. Thibbotuwawa [87] shows that there is a critical target distance-to-width (A/W) ratio of approximately 10:1 at which the transition to open-loop aiming occurs. Below the 10:1 A/W ratio, movement time increases linearly with the square root of distance. Above the 10:1 ratio, movement times were modeled accurately by Fitts' Law.

In summary, although there is some evidence that target acquisition can be performed in an open-loop manner under specific conditions, there is overwhelming support for visual feedback being an important part of the aiming process. When trajectory errors are made due to neuromotor noise or external disruption, visual feedback strongly contributes toward correcting the trajectory. Therefore, the lack of visual feedback, or the disruption of it for reasons such as latency in feedback, is likely to result in decreased aiming performance.

2.1.3 Tracking

Previous research on target tracking is not as extensive as that of Fitts-like target acquisition movements. Tracking actions have also seemingly not been modeled in their entirety as acquisition has, but there is a significant amount of research about how tracking is performed and its characteristics.

A number of studies indicate that pursuit tracking (i.e., following a moving target) is composed of a sequence of discrete submovements [36, 70, 73], rather than being performed through a continuous smooth motion. For increasing movement frequency (and thus target speed), submovement speed increases in amplitude, but the submovement duration is unaffected [81]. Faster target movement results in greater error in tracking – this speed-error relationship is observed because targets travel farther in a fixed amount of time at higher speeds (and especially during delays during which feedback is awaited), which necessitates larger corrections.

Tracking movements make use of both visual feedback and feedforward mechanisms. Each submovement is a planned corrective movement programmed to compensate for error in tracking. After each submovement, visual and kinesthetic feedback is used to assess the trajectory and compare the endpoint position to the target [36, 70]. Each correction has a response latency that is based on reaction delays in feedback. The delays vary and increase with target speed, with a range of approximately 100 to 330 ms reported by Grossman [36] within the tested speed range when subjects had no preview of future target motion.

Although movement anticipation is performed in most conditions, when target speed and relative difficulty of the tracking task becomes high enough, subjects start being unable to maintain accurate tracking and they become increasingly more reliant on anticipating target movement. According to Grossman’s findings, while multiple submovements occur for each target direction change at lower frequencies, once movement frequency reaches approximately two direction changes per second, subjects are able to perform only one corrective action for each target direction change [36].

Keele [46] presents an extensive and in-depth examination of movement control in skilled motor performance, covering both single target acquisition under Fitts’ Law and continuous

target tracking movements. Visual feedback processing time is said to be an important factor in both. When tracking a target and the target changes its direction, it takes approximately 200 to 300 ms before corrective motor action is applied, and thus targets with more frequent direction changes should be more difficult to track. It is also suggested that continuous tracking should be predictable as a series of submovements. Faster target speeds effectively increase the distance to the target when tracking errors occur, and smaller targets make the series of acquisition tasks more difficult, as predicted by Fitts' Law.

The Steering Law [3] models the action of moving a pointer through a tunnel, with the trajectory being constrained on both sides. According to the Steering Law, time to complete the steering task increases linearly with increasing tunnel length, and decreases with an increasing tunnel width. Although steering seems similar to tracking or following a target, it is not equivalent because the Steering Law assumes that the tunnel side boundaries are never crossed and it does not consider the longitudinal direction – that is, the user could stay consistently ahead of or behind the target, which would cause errors in tracking that the Steering Law does not consider.

2.2 First Person Shooters and 3D

My work directly focuses on aiming performance in 3D first person shooter game settings. Therefore, we examine the literature on player performance in FPS games, as well as its ties to the relevant general motor control performance topics.

2.2.1 Fitts' Law in Virtual 3D

Since Fitts' Law applies to constrained motor movement tasks in general, it should apply to first person shooters as well. However, some transformations must be performed first due to the way that pointing is executed in 3D virtual environments that are projected onto 2D screens and manipulated with 2D input devices (such as mice) [86]. The magnitude of input movement required to acquire targets at varying locations in the virtual world is non-linear due to input movements resulting in a rotation of the view (and thus aiming direction) rather than a translation. For example, a target that moves laterally in the virtual world

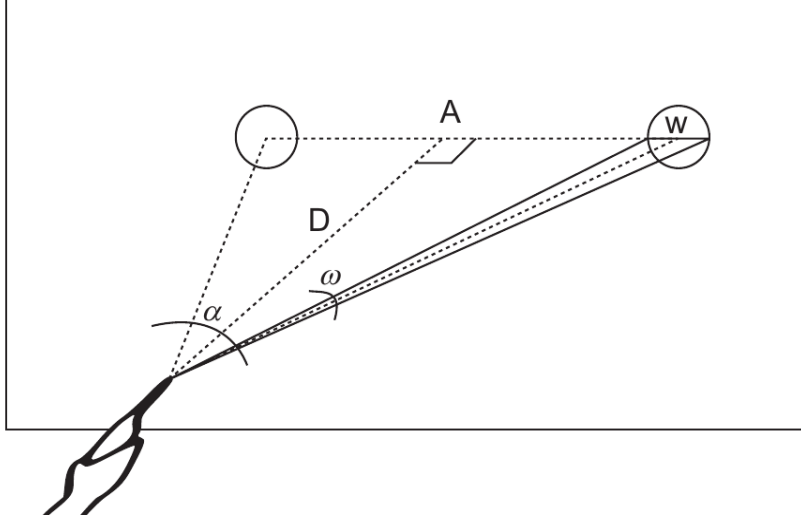


Figure 2.2: Relationship between width and angular width, and distance and angular distance.

by d units might require a 1 cm movement of the mouse, but if the target moved $2d$ units instead, the input required would be less than 2 cm, but the target would appear smaller at its new location. Figure 2.2 shows the similarity between angular and non-angular width and distance terms.

Kopper examined distal pointing behavior in real-world 3D pointing using a laser pointer to point at targets [48] and reached conclusions and transformations similar to mine. He defined angular distance as

$$\alpha = 2 \arctan \left(\frac{0.5A}{D} \right)$$

and angular width as

$$\omega = \arctan \left(\frac{0.5(A+W)}{D} \right) - \arctan \left(\frac{0.5(A-W)}{D} \right)$$

and then proposed a new formulation for Index of Difficulty:

$$ID_{angular} = \log_2 \left(\frac{\alpha}{\omega^k} + 1 \right)$$

In this formulation, the width term gains an exponent due to this approach fitting the data better in Kopper's experiment. The angular width seemingly had a more important effect on trial time than distance. The given possible explanation for this finding is that the ballistic portion of the movement was found to almost always be quick, and the vast majority of time

was spent in the corrective portion which was more affected by width. Potential reasons given for this difference are the hand tremor present in using a laser pointer, and the disturbance of pressing the button on the pointer. Neither of these factors are likely to be significant in mouse pointing.

Looser et. al. performed a study [54] that is similar in concept to my study presented in Chapter 3 in which he studies target acquisition tasks in both a 3D first person shooter environment and a standard 2D setting. They compared Fitts' Law regression results in both 3D and 2D and found that Fitts' Law holds in both cases, suggesting that FPS aiming tasks can be used in place of traditional ones, with the motivation being that participants would find the task more compelling and less tedious to complete. Unlike Kopper, Looser used the standard Fitts' Law model with no exponential term on angular width, but in order to work in 3D, he also converted distances and widths into angular units.

Zaranek [97] performed a study with a shooting task based on the ISO 9241-9 experimental protocol [42], and proposed that two concerns with using Fitts' Law in 3D are perspective distortion of targets near the edge of the screen and the effect of target depth. While these are valid concerns and may affect the pre-programmed ballistic phase of acquisition due to the distorted apparent size of the target, targets are ultimately fired upon once centered in the screen and thus perspective distortion during the corrective phase should be relatively minor. Likewise, if neither the player nor the targets are moving, a change in target depth should be equivalent to an inverse change in target size when centered, although perspective distortion may again have an effect on the ballistic initial motion. Regardless, Zaranik does find some significant effects of target distance, especially on error rate and when using motion controllers [97].

Grossman and Balakrishnan [37] present and evaluate a version of Fitts' Law for three-dimensional space in which the A/W term becomes separated into A/W , A/H , and A/D (width, height, depth). Each of the three terms is weighted by empirically derived constants, and the Euclidean norm is then taken, with the resulting value replacing the original A/W term. However, their work focused on true 3D space, where the input device has three degrees of freedom for movement, and thus no projection onto 2D screens nor rotational input device movement is involved.

2.2.2 Aim Assistance

Aiming assistance is the usage of techniques to aid users in tasks that require aiming at targets. Since modeling of aim is rooted in Fitts' Law, these assistance techniques focus strongly on manipulating the factors that describe the difficulty of the aiming task – that is, target distance and width [10]. Assists can seek to reduce the effective target distance in ways such as bringing targets closer or increasing effective target width (e.g., increasing target size when aiming near it), or to affect both distance and width.

In addition to the goal of improving aim performance in general use, aim assists have been applied to various targeted applications, such as helping older [95] or motor-impaired people [32], compensating for control devices with poorer performance such as gamepads [66], or balancing opponents' skill levels in games [11, 89]. Many targeting assistance techniques have been developed for both 2D and 3D environments. We will take a brief look at several techniques and applications here – see [89] for a more thorough review.

My form of aim assistance used in Chapter 5 is inspired by two existing aim assistance techniques: sticky targets and force fields. Sticky icons [95] (or targets) improve target acquisition by increasing the effective width of the target in motor space as the cursor passes over the target. Therefore, the assist does not help the user to close the distance to the target, but rather it makes it more difficult to overshoot the target or otherwise accidentally move away from it. While sticky targets affects the speed of the cursor over the target, force fields [6] affect the position of the cursor while it is within the region that force fields apply (generally within the target). The cursor is displaced toward a certain position on the display while affected by the field, which is typically toward the center of the target.

Another widely used assistance technique that improves effective target width is area cursors [45]. While pointing and target acquisition is typically performed by aiming with a single point (i.e., the tip of a mouse cursor or center of a targeting reticle), area cursors increase the targeting area to a rectangle, so that the entire area within the rectangle is sampled for hits. This technique allows much easier selection of small targets, but fails to perform well when target density is high enough that multiple targets become covered by the area cursor. Another technique, DynaSpot [19], improves on this limitation by activating the

area cursor only while the cursor is moving fast enough, and using the standard point cursor while at rest or low speed.

Vicencio-Moriera and colleagues [89] examined several forms of aiming assistance as aids for game balancing in FPS games when players possess disparate levels of skill, including target lock (based on related work by Guiard [39]), bullet magnetism, area cursor [45], sticky targets [15], and target gravity [11]. Their work finds that there is difficulty in transferring the 2D-based assists into 3D FPS environments, and that many techniques lose their effectiveness. This suggests that there would be value in directly comparing targeting tasks between 2D and 3D settings, and that the comparison would be useful when developing aim assistance or manipulation techniques for 3D applications.

2.3 Latency

In the context of interactive computer systems, we define latency (or lag) as the time delay between a causative input or triggered event and the expression of the corresponding effect. Latency can be observed in scenarios such as loading an image from disk and waiting for it to be displayed, or pressing the trigger button in an online first person shooter and waiting for the resulting shot to be registered on the server. Latency is pervasive in digital systems and can vary greatly depending on the devices in the system (e.g., some LCD televisions have high latency due to built-in image processing). Latency almost universally induces negative effects on user performance or experience throughout a wide variety of contexts. The negative effects of latency have been reported for a number of computer applications, such as collaborative groupware [85], video scrubbing [63], digital sketching [16] and video games [13, 68].

My work mainly focuses on the effects of local latency on player performance in first person shooter games. Therefore, it is important to examine the previous body of research related to latency. Some of this previous work is based on local latency, whose detrimental effects on performance have yet to be successfully compensated. Other work is based on the effects of forms of latency that can be compensated at least partially, such as network-based latency.

2.3.1 Network and Local Latency

Users may encounter latency in many forms; in games, sources of latency can be broadly classified as either network latency or local latency. With computer-based interactions being increasingly tied to the Internet in modern usage, many people (and particularly gamers) associate latency primarily with network-based sources of latency, which arise due to signal travel distances along network paths and processing delays of routers that make up the network. Networking latency, which is also commonly referred to as *ping*, is readily apparent in contexts such as online games where a connection quality meter is often prominently displayed to indicate the player's or opponents' ping times. These ping meters have been widespread since the early days of online gaming, likely because of the strong negative effects of latency on player performance or quality of experience [68], and the ability for varying ping levels between opponents to cause unfair disadvantages for players more heavily affected by latency [96]. Ping varies significantly depending on distance and routing to the server. Typical ping times for internet connectivity are very approximately 30 ms to 100 ms, but even for a given distance to the server, network congestion can a wide range of ping times to be experienced.

Local latency, unlike network latency, is caused by factors local to the particular computer system being used for a task or activity. These sources of latency are due to elements such as buffering delays in display hardware, buffering or filtering during software runtime, or synchronization delays between different stages of a system (such as the application processing loop and display re-draw events). Terms equivalent to local latency have seemingly not been created in research literature previously, and while the term *input lag* is typically used to describe the phenomenon colloquially, it implicitly appears to refer to latency caused by the input aspects of a system, although in reality it typically refers to the total latency in a system (aside from network-based sources). Therefore, I use the term local latency to refer to this type of latency, which I examine in Chapters 4 and 5.

In online games, latency occurs due to both network lag and local latency, while in offline games, latency is solely due to local latency. In many contexts, including online games, local latency manifests itself differently than network lag. For example, local latency delays the

player's own cursor or view movement feedback, while network lag rarely does. Also, online games typically employ mechanisms to compensate for the effects of network latency, such as the dead reckoning technique [69]. Due to differences such as these, it is important to examine local latency and its effects separately from network latency.

2.3.2 Sources of Latency

The magnitude of local latency in a system is determined by the combination of individual components in the system, including the input device, software processing components, and output devices. Examining the contribution of each component toward total latency has not been documented in literature, nor is it within the scope of my research to experimentally test individual components. However, it is helpful to understand where latency comes from and how significantly each element in the system contributes to it, and therefore I present a brief outline here based on my knowledge of computer system architecture and components, separated by component type. For another similar examination, refer to [67] which takes a detailed look at where latency comes from in an Android smart phone. Although some of the Android components do not apply in computer gaming (such as the touch screen hardware), most other components are equivalent.

Input Subsystem

Most input devices in a modern system are connected through the universal serial bus (USB) which polls devices at discrete intervals. In Windows operating systems, input device sampling occurs at 125 Hz (i.e., every 8 ms) [18], which means that there can be up to 8 ms of latency on the input, or 4 ms on average. For example, if a game reads the input state from the OS 1 ms before a USB poll occurs, the game is operating on input that is 7 ms old. While it is possible to raise the default polling rate through third-party tools, such operations are typically unsupported.

For complex input devices that require heavy processing or have limited update rates, such as the PlayStation Move or Microsoft Kinect motion control systems, latency can be significantly higher. Also, inherently jittery input devices such as touch screens or game pad joysticks are often filtered for smoothing before being processed, which adds latency.

Application processing

The game or application itself can often be the cause of the majority of latency in a system. Since applications can do any number of things that cause latency, the magnitude of latency can vary widely; however, there are some common patterns. Games are updated in discrete intervals (also called frame rates when referring to visual output), causing latency that is potentially as long as the update period. Games typically run at 30-60 frames per second which results up to 17-33 ms of latency, although the average would be approximately half the maximum amount.

Vertical synchronization (v-sync) is commonly used in order to eliminate screen tearing and achieve potentially smoother visuals. When enabled, v-sync causes the game update to wait until the display has finished its update scan-out. Since this results in the input state being read immediately after the display updates, by the next time the display is updated, the displayed image will be based on input that is a full display refresh period out of date. Since the vast majority of current displays receive frame updates at 60 Hz, this results in 17 ms of latency. Furthermore, unless triple buffering is used, if a game update takes longer than the display refresh period (causing it to miss the v-sync point), the resulting frame will be shown at the next v-sync point or later. The consequence of the miss is double the latency, half the frame rate, and perceptible stutter. Triple buffering eliminates this downside of v-sync, but potentially induces more latency itself, depending on implementation.

Game loops are typically implemented in one of several common ways [88]. Simple coupled input-update-render mechanisms result in low latency, but other designs are often used due to requirements of multi-threaded systems or deterministic simulation. For example, online games, games with replays, and simulation games often require decoupled game update and render subsystems. This strategy allows logic updates to be performed with a fixed, deterministic time step, while simultaneously allowing rendering to be performed more frequently, which results in smoother visuals. The disadvantage of this scheme is that the rendering system needs to interpolate between two logic states, which necessitates additional latency equal to the logic update time step. Games commonly perform logic updates at 30 Hz, which results in a fixed 33 ms of extra latency.

Another way that games can induce latency is by performing smoothing or other filtering on the input, which is often done in systems where a mouse is not the primary method of input. Also, some rendering techniques require compositing with previous frames, which also adds latency. Additionally, any latency such as waiting for disk access will delay the process, causing momentary lag. Overall, the latency caused by software delays is often 67 to 133 ms.¹

Output subsystem

The display is a main contributor to local latency. While old CRT displays were almost universally low latency, LCD displays are often laggy, which has been shown to make them not comparable to CRT displays in uses such as pattern reversal visual evoked potentials [41]. Although specialized gaming monitors have low lag (about 10 ms), most monitors average about 20-40 ms of latency [27]. Televisions, which are popular in the game console community, often have significantly greater latency than monitors due to internal processing such as motion smoothing and on-screen menu support. The pixel response time in displays – that is, the time needed for a pixel to fully transition from one color to another – also contributes to the latency (2-10 ms). Additionally, displays almost always operate at 60 Hz update rates (even when they interpolate to higher rates, as is commonly seen in televisions), which means that full screen updates cannot be shown any quicker than the refresh rate allows.

The operating system and video drivers can also contribute to latency. Modern operating systems use compositing desktop environments which aggregate all visible application windows into a single frame to be drawn. The compositing step can add some latency, particularly when v-sync is used, but games are able to bypass this factor by using exclusive-mode fullscreen output [34]. Additionally, depending on application implementation and video driver settings, the video driver may buffer several full frames of output before drawing to the screen. For example, Direct3D-based applications often make use of frame pre-rendering, which by default has the CPU prepare 3 frames in advance before they are rendered by the GPU. This practice can result in smoother frame rates, but since this means all output is delayed by 3 frames, there is a high latency penalty of 50 ms or more even at a relatively high frame rate of 60 Hz [92].

¹https://en.wikipedia.org/wiki/Input_lag

2.3.3 Latency Compensation

The presence of latency in real-time interactive online games and other groupware has long been a significant problem that causes issues with user performance and quality of experience. Without compensation, latency causes the user's own actions and their response feedback to be delayed, as well as seeing an outdated state of the world that requires the user to anticipate what the true state is at any given time. As such, a majority of online games and applications make use of latency compensation techniques in order to improve user experience. I present here a brief overview of common techniques to manage effects of latency, both in the well-studied network latency context, and a brief look at the early work to mitigate local latency.

Network Latency

A common approach for minimizing the appearance of latency is the usage of prediction techniques such as extrapolation in the form of dead-reckoning [5, 69]. In this approach, clients predict the current state of other users by using information about the last known state. For example, in FPS games, if the previous position and velocity information of a remote avatar is known along with the timeliness of that information, the current position can be predicted by adding the velocity vector to the old position of the avatar, scaled by the amount of time that has passed.

Although prediction is helpful in mitigating the perception of latency, significant problems arise when using prediction in FPS scenarios because the client cannot know what actions other users will perform during the predicted period. The result of such mispredictions is seen as jerky, jumpy movement of other players' avatars. One way to reduce the severity of prediction errors is to smoothly interpolate between the incorrect avatar position and the new correct position once a misprediction is detected [79] (Figure 2.3). These smooth corrections are more effective (i.e., less perceptible) when gradual adjustments are made to correction rate rather than adjusting at a constant rate, and also, the speed of the correction is more important than the magnitude in determining whether it is perceptible [78]. Another way to improve prediction is to reduce the magnitude of prediction errors by making more intelligent predictions through means such as analyzing play patterns [4].

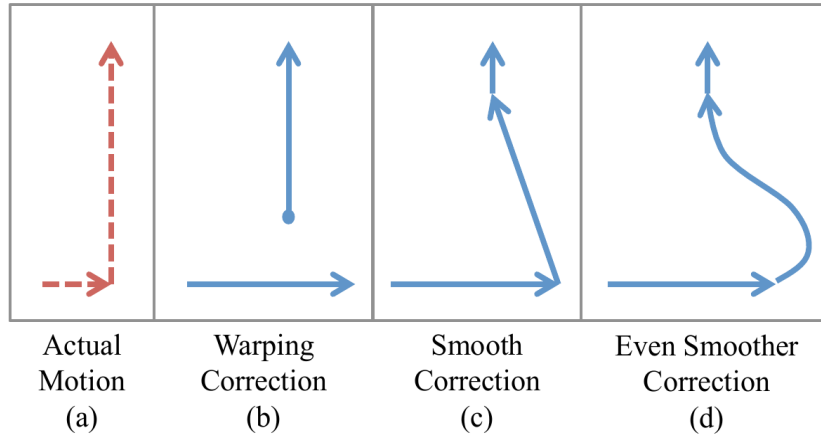


Figure 2.3: Actual object motion path versus the path seen by remote peers under different prediction error correction methods.

Because of the downsides associated with extrapolative prediction, FPS games more commonly use a system of interpolation combined with remote lag. This approach has been used and documented [14] by the makers of *Half Life*² (a dated but long-popular FPS game) and the general technique is still dominant today. In this technique, other players are interpolated between two previous known states. Rather than using the most recent update and the one before it to interpolate between, states that are one or two updates older are used instead, which results in more latency but creates tolerance for missing an update or two. As a result, enemy movement is smooth and predictable when a reasonable connection quality is used. Extrapolation may be used when there are connection dropouts longer than the period that interpolation is effective.

Interpolation results in a smoother experience, but displayed enemy positions are even more out of date than if they were simply displayed as updates arrive. As a consequence, compensation is required in order to allow players to interact with others in a manner that hides the latency – for example, players should be permitted to shoot at a target at the location that it currently appears to be rather than needing to predict where the target really is. One way to accomplish this is to trust clients regarding the outcomes of their actions. For example, if a player fires at a target and appears to hit it in their own representation of the world state, the server (or other clients) could accept this claim and execute the action as

²<http://www.valvesoftware.com/games/backcatalog.html>

intended. However, in games that are open to the public, clients are rarely trusted with such decisions due to the wide prevalence of cheating. Instead, as is done in Half Life, before the server simulates the outcome of an input update from a client, it first temporarily rewinds time to a state that the world was in at the time the client performed the action [14]. In this way, the server can judge whether such an action was feasible to be performed on the client, greatly reducing the potential for cheating.

Other techniques have been devised for groupware where consistency and integrity is a higher priority than minimizing perceived latency. Local lag [26, 85] purposely adds latency to the local client in order to present what remote peers will see at the time the messages are received. Pointer trails [40] display a visual representation of past actions performed by peers in order to give more context for during which state the actions were performed.

Local Latency

Although seemingly little research has been performed on compensating the effects of local latency, there is some recent work that may lay foundations for future development. With head-mounted virtual reality (VR) displays experiencing a recent resurgence, there is a push for finding methods of minimizing latency or making it less perceptible. With VR in particular, even small amounts of latency can result in motion sickness or otherwise a significantly degraded experience.

Asynchronous timewarp [8] is a technique employed by the Oculus VR SDK.³ With this technique, a game frame is processed the standard way at first – that is, input is sampled after a previous frame has been rendered, then the game logic and rendering is done before a frame is sent to the display. However, with timewarp, an extra step is timed to occur shortly before the display refresh occurs: the frame is re-rendered again in a quick, lightweight method using the most recent rotational and positional sensor data from the VR headset. Game logic does not execute a second time, but rather the view is rotated and translated using the existing computations. The end result is less perceived latency and judder, although there are technical limitations that can limit the efficacy of the technique, such as limited timing precision, increased GPU load, and black edges appearing where there is no rendered data.

³<http://www.oculus.com>

Alternatively, Oculus is considering predicting headset rotation and position by extrapolating the original sampled data, which has its own set of disadvantages.

There is also a patent on a device that is intended to compensate for local latency caused by frame buffering [47], although there seems to be no research or accompanying product associated with it currently. The system appears to work in a way similar to the remote lag technique described in the network latency compensation section above – when evaluating the outcome of a user’s inputs, the actions are executed within a previous state that has been saved by the system.

2.3.4 Effects of Network Latency on Games

The negative effects of network-related sources of latency on online games have received significant attention in research literature. Although such latency manifests in different ways than local latency does, it has significant effects on player performance and balance in games [22]. Interestingly, the relationship between latency and player performance varies widely depending on the view type, and the timeliness, precision, and impact of actions in the game [24].

In a 2006 publication, Claypool examined existing work related to network latency in several different types of games [22]. He proposed that interactions in games can be classified into a two-dimensional taxonomy between precision and time deadline of actions (Figure 2.4), with actions requiring more precision and quicker timing being most sensitive to latency. Likewise, since different game genres are made up of games requiring actions with varying precision and deadlines, latency sensitivity of different game types was found to vary significantly. Games where players control an avatar in first-person perspective (e.g., first person shooters and racing games) were found to be most sensitive to latency due to the first-person view requiring immediate feedback in order to achieve a high degree of control. Games where avatars were controlled with a third-person view camera were less sensitive to latency, and games using an omnipresent camera (e.g., real-time strategy) were considerably less lag-sensitive (Figure 2.5). Using the aggregated performance versus latency charts, threshold levels of latency needed for an approximately 25% performance drop compared to no latency were suggested: 100 ms for first person avatar, 500 ms for third-person avatar,

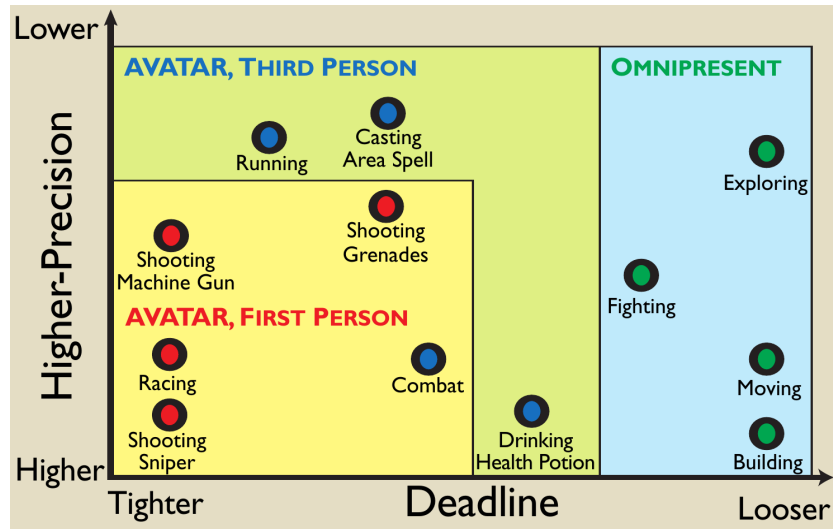


Figure 2.4: Taxonomy of player interactions, classified between precision and timeliness requirements. Adopted from [24].

and 1000 ms for omnipresent games.

In followup studies, Claypool performed a more detailed examination regarding the precise effects of deadline and precision on performance; however, these studies used AI-controlled players and therefore the applicability to games played by real players may be compromised since players are capable of adjusting to latency and the AI did not seem to attempt to do so. AI characters played a first-person 3D combat tank game (BZFlag⁴) in which the size of enemy tanks (and therefore precision required) and the speed of the shells (deadline) varied [23]. When the tank was 25% of its normal size, the lagged player scored approximately 10% worse, and there was a negligible effect of lag when tanks were at 400% of their normal size. The results were similar when bullet speeds of 25% vs 400% were compared as well, with faster bullets being more sensitive to lag.

In 2015, Claypool introduced a third factor to the taxonomy, that being the impact of an action [24], and evaluated the interaction of impact with latency in a Space Invaders-like game where the impact of different actions can be evaluated. In summary, attack actions that deal more damage, travel faster, or have a greater area of effect have more impact on the importance of the action and are therefore more sensitive to latency. Using this information, Claypool proposed and evaluated a network congestion compensation technique

⁴<http://bzflag.org/>

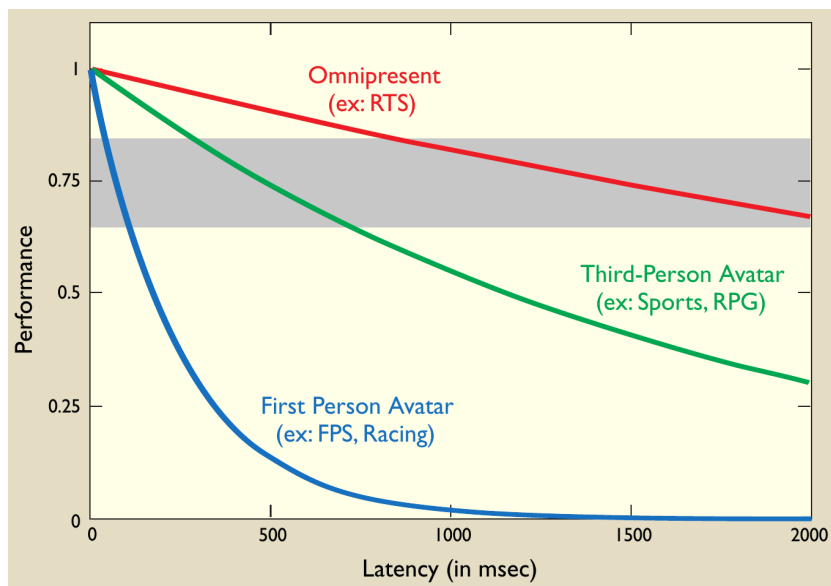


Figure 2.5: Player performance versus latency magnitude in three types of games. Adopted from [24].

that prioritizes packets that contain actions predicted to be more important (using deadline, precision, impact) when some packets must be dropped, and showed that the compensation was effective at reducing the effect of latency.

Studies have also been performed that examine the effect of latency on metrics other than player score. The level of jitter that causes players of Call of Duty: Modern Warfare 2⁵ to report a poor experience was studied by [7]. It was shown that when the congested player is the host of a session, jitter of up to 100 ms (on top of 100 ms of static latency) was required for a degraded experience, while jitter of 250 ms was required when the affected user was not the host. Interestingly, while expert players were shown to be sensitive even to the condition with no jitter and 100 ms of latency, they were also the best at compensating for network conditions to receive the least performance penalty.

Another study performed in the context of a massively multiplayer online role-playing game (MMORPG) [20] showed that players were much less likely to play for extended periods of time when network conditions were poor. While players logged sessions of 4 hours with normal latency of approximately 150 ms, they played for less than one hour when ping was 250 ms. Similarly, even small amounts of jitter (10 ms) and packet loss was shown to cut

⁵<https://www.callofduty.com/>

play times in half. From the data, the authors created a model that predicts the likelihood that a user will quit at any given point given the latency, jitter, and packet loss levels.

2.3.5 Effects of Local Latency

Local latency has recently become a subject of interest to gamers⁶ with new types of gaming systems that use high-latency devices, such as gesture input or televisions, and latency-sensitive applications such as virtual reality headsets. Additionally, many systems, such as tablets, smart phones, and Nintendo's Wii U, now rely on touch devices that suffer from significant local latency. While local latency increases in prevalence, it has been shown that lag can have significant effects on performance at surprisingly low levels – in a study employing a Fitts' Law-like task with direct touch input, it was found that latencies as low as 10 ms reduced drag-and-point performance [44]. Also, in addition to my own work, it has recently been shown by another study that local latency accounts for a significant portion of total system latency in online games, and should be accounted for [76] when considering latency-based factors.

Several studies have been performed examining the effect of latency on target acquisition in 2D cursor-controlled tasks. In an early study by MacKenzie, it was shown that 75 ms of latency has a substantial effect on acquisition time (16% longer movement time), and that acquisition took 64% longer at 225 ms compared to baseline latency [61]. MacKenzie suggested that these severe effects at high latency could be because a person's natural tendency to anticipate motions is severely compromised by lag. A similar study by Pavlovych found different results: acquisition time at 33-83 ms of latency was similar to baseline, with significant differences appearing starting at 108 ms [72]. Pavlovych directly compares these results to MacKenzie's and hypothesizes that the reason for the discrepancy could be that MacKenzie's system had approximately 60 ms more latency than expected, but such claims cannot be verified. Pavlovych also found that there is a significant interaction between target width and latency, with smaller targets being more highly affected by latency than larger targets. This finding is consistent with work in 3D VR environments where it has been shown

⁶Evidence for this can be seen from the numerous discussions returned by a web search for "input lag."

that width and distance should be treated separately under the presence of latency [83].

Target tracking involves keeping a pointing device over a moving target. Pavlovych examined the effect of lag on target tracking in 2D and found significant interaction between lag and longitudinal tracking errors [71]. Error increased quickly for latencies over 110 ms, for jitter above 40 ms, and for dropout rates above 10%. Also, target speed interacted significantly with latency, resulting in faster targets being significantly more difficult to track under higher latency compared to lower latency. However, the 2D target motion used Lissajous curves,⁷ which are smooth curves that do not match the horizontal ground-plane movements seen in FPS games. The gradually curved target motion using Lissajous curves may aid the participant's ability to predict the target motion in high latency conditions and therefore extrapolation to FPS games, where player movement is more chaotic with quicker direction changes, could be limited.

Cloud Gaming

The research area that is perhaps most relevant to local latency in 3D first person shooters is that of latency in cloud gaming environments [43]. In cloud gaming, a *thin client* transmits inputs to a server that actually runs the game and processes the input. The resulting rendered frames are sent back to the thin client to be displayed. In this manner, the client does not need powerful processing hardware; however, the client also does not know about game state and thus cannot make use of latency compensation techniques available to normal full clients. This lack of compensation extends to core mechanics such as rotating the view, which has lag-free feedback in full clients but is delayed in cloud gaming. Thus, the effect of network latency in a cloud environment is similar to that of local latency in a standard setting, although cloud gaming is also more prone to jitter in latency, packet loss, limited bandwidth, and other such characteristics of networks.

Lee examined how latency in cloud gaming affects player experience in various games using an electromyograph (EMG) device [52]. Similarly to Claypool's findings for network lag outside of cloud environments [22], Lee found that latency in cloud gaming is more detrimental for some types of games than others. Once again, first person shooters were most

⁷https://en.wikipedia.org/wiki/Lissajous_curve

affected by latency, followed by role playing games, and action games were least affected due to attacks being easier to connect with. Lee also created a model that predicts the suitability of any given game to cloud gaming. The model is based on the ratio of screen dynamics to input rate, with a higher ratio requiring higher real-time strictness.

Another study designed to determine the significance of latency in cloud gaming found that individual differences determine the degree to which players can tolerate latency [75]. A small number of players could only detect delays of 97-182 ms, and some others could perceive latency as low as 26-40 ms. In the median case, people could perceive 51-90 ms of latency. The study found that while a player's amount of gaming experience is not the determinant factor on whether latency is perceptible, the number of corrections used in movements is correlated with how sensitive players were to latency, and more experienced players tended to use more corrections. It should be noted that the experiment consisted of a simple target selection task that used a jog wheel input device with a button, where the only inputs were left-right rotations and clicks. Since it was previously shown that more visually dynamic environments are more reliant on low latency, the results may apply somewhat differently to typical cloud games.

CHAPTER 3

COMPARISON OF POINTING IN 2D VERSUS 3D

A large amount of research related to pointing has been performed in the context of two-dimensional environments in which users move a pointer around a screen in a fixed-background setting. This research covers a wide range of applications, including pointing assistance [95], pointing device comparison [60], and measuring the effects of latency [72]. However, it is unknown whether this research may be directly applied to three-dimensional first person shooter-like environments, because there may be substantial differences between the two settings which could interact with previous findings.

In order to find whether there are performance differences between the two and three dimensional environments, I performed a user study that compared participant performance metrics in aiming between the two environments in settings that are as similar as possible to each other except for dimensionality. In addition to providing a contribution by comparing performance in 2D versus 3D, I also performed the study with the intent of using the findings to inform my work on latency and aim assistance (Chapter 5)

3.1 Apparatus

The study was performed on a performance custom-built gaming PC (see Table 3.1 for specifications). A high performance gaming display with a high refresh rate of 120 Hz and 1 ms pixel response time was used in order to minimize latency – such a display is often used in FPS gaming competitions as well. The display was set to its native resolution of 1920x1080 and full screen mode was used. A high performance gaming mouse polled at 1000 Hz was also used for minimum latency and smooth, responsive tracking. The experimental game ran at a constant 120 frames per second when vertical sync was enabled, and in the range of

Table 3.1: Experimental PC specifications.

Component	Specification
Processor (CPU)	Intel Core i7 3.2 GHz
Video Card (GPU)	NVidia GeForce 460 GTX
Memory (RAM)	16GB PC1600 DDR3
Motherboard	Asus P8Z68-V LE
Storage	Intel 520 SSD, 120GB capacity
Mouse	Razer DeathAdder 3.5G, 1000 Hz sample rate
Display	BenQ XL2420TX, 120 Hz refresh rate
Operating System	Arch Linux 64-bit
Video Drivers	NVidia standard closed-source

hundreds of frames per second without vertical sync; thus, the PC was more than capable of ideal performance within the experiment.

Mouse sensitivity (gain) was set such that it felt subjectively appropriate and acceptable to most participants. Participants were allowed to adjust sensitivity if it was significantly different than they are normally used to, although no participants did so.

The experiment was performed in a small, closed, quiet room with minimal distractions and were asked to silence their mobile phones. Participants adjusted their chairs to an appropriate height and had ample space to move their mouse on a large mouse pad.

The experiment was executed within a custom-built application made to resemble a first person shooter game (while in the 3D condition), which guided users through each step of the study session (with assistance from the experiment administrator). It was developed in the C++ language using the Ogre3D open-source graphics library¹ (version 1.8). Ogre3D was used to simplify OpenGL 3D graphics rendering, keyboard and mouse input, sample 3D models, and Quake 3 BSP file format loading and rendering. The BSP file capability was used such that a simple game world could be created using a Quake 3 level editor and easily used within the experiment.

¹<http://www.ogre3d.org>

3.2 Tasks and Procedure

My experimental system implemented two primary tasks to be performed by participants: target tracking and target acquisition. These tasks are representative of primary activities in many 3D game environments; in FPS games, target acquisition is carried out when first encountering an enemy, and then a combination of acquisition and tracking is used until the enemy is eliminated. The entirety of the study consisted of these two tasks being performed in short and discrete trials with varying conditions.

All tasks and conditions were performed in a custom-made game world created using GtkRadiant.² This world consisted of one room divided into two sections, with one section used for the acquisition task and the other used for the tracking task (Figure 3.1). The player position was fixed to a location at the center, between the two rooms. In the 3D conditions, the world had a graphical appearance very similar to *Quake III Arena*, which was a popular first person shooter.

3.2.1 Acquisition

In first person shooters, aiming typically begins with target acquisition. The player performs acquisition by moving their aim to a direction corresponding to an intended target. In shooters, this target may be a specific entity such as an adversary combatant, or a location in the world environment chosen in order control space by making it hazardous for enemies to cross the targeted area. In this study, I focus on direct aiming at specific targets.

Each target acquisition trial within the study began with the participant's aim being set by the system to a base view initial direction, aiming down range to the middle of the cluster where targets appear. This aim direction was locked for a period of 500 ms at the beginning of each trial. During this time, the participant could not adjust their aim, ensuring that each trial began with a controlled initial aim direction, and the 500 ms pause allowed participants to prepare to aim at the target.

At the end of the 500 ms period, the trial timer started and a spherical target appeared.

²Level creation tool for creating Quake 3 compatible maps: <http://icculus.org/gtkradiant>

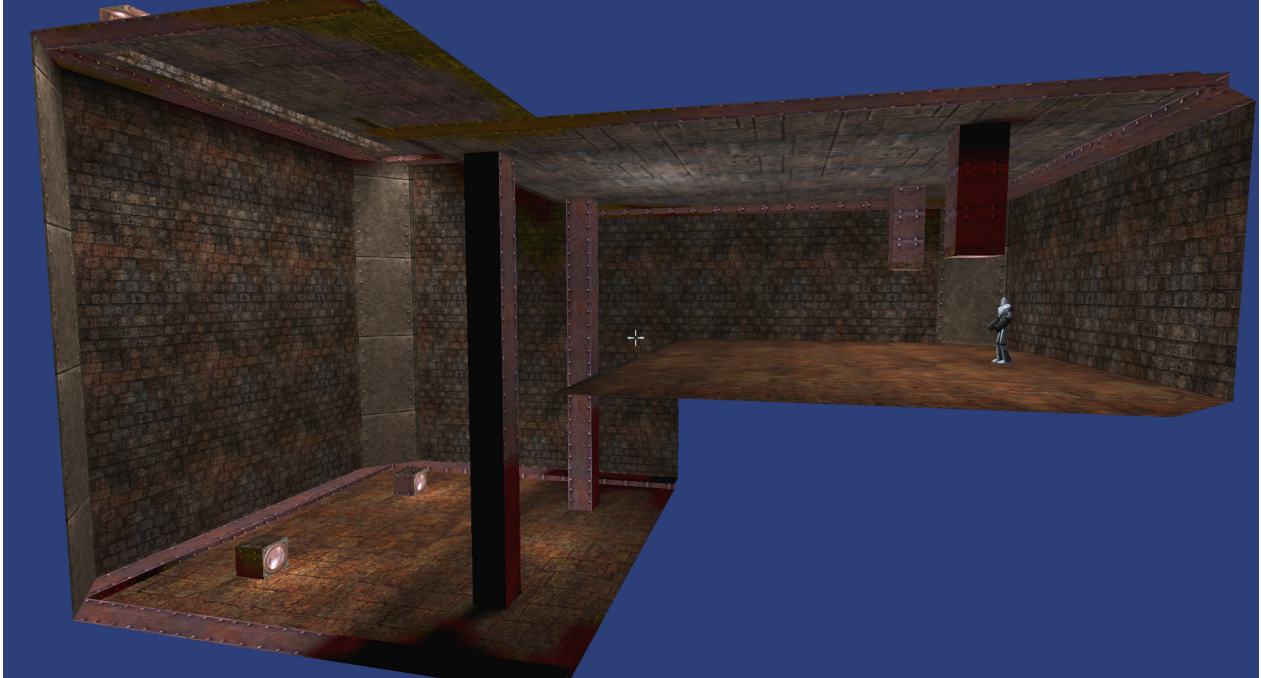


Figure 3.1: Side cut-out view of the world. The large room shown on the left is used for target acquisition, while the area on the right (tracking target shown) is used for tracking. The player location is fixed in the middle.

At this point, the aim lock was released, and participants attempted to aim at the target then click the left mouse button to shoot it. Participants were instructed to complete trials as quickly as possible while trying to be reasonably accurate. The main dependent measure was trial completion time, with lower times being better.

Participants were also told that the targets would start appearing red-tinted if accuracy dropped below a threshold, and to be more careful and accurate if this happened until targets stopped appearing red. The threshold was reached if accuracy fell below 80% at any point within a block of trials (see Section 3.6). If the participant shot but missed a target during a trial, the trial would continue until successfully completed, but it would be flagged as containing misses (errors) and repeated until completed without errors.

Each sub-block of trials consisted of 72 possible target positions within a virtual circle located in front of the player, perpendicular to the initial view direction (Figure 3.2). The position order was randomized, but the same order occurred within every block and participant. This was done by randomizing the order of the possible positions with a fixed random number generator seed. The targets were all visible on screen from the base view position



Figure 3.2: 2D Acquisition task showing a sample of various target distances, widths, and directions.

without having to rotate the view.

Units and Measurements

As required by standard game and graphics engines, my experimental world was modeled in 3D Euclidean space with Cartesian coordinates, with each object having positions and sizes given in coordinates along x , y , and z axes. I use the term *world units* to refer to distances within this Cartesian system. Fitts' Law calculations require amplitude (i.e. distance) and width measurements, but it is unsuitable to use Cartesian coordinates from within a 3D game environment for this purpose because they do not accurately reflect the input device movement amplitudes required to acquire a target – because the view rotates rather than translates, the movement magnitude is non-linear with respect to the Cartesian distance of

the target.

As discussed by Looser [54] and Kopper [48], angular measures are required for analysis within a 3D first person shooter-like setting. Therefore, for the 3D target acquisition task, I converted world unit measures into angular measures to use as inputs for distance and width in calculations. Here, distance (amplitude) is the amount of rotation required to bring the center of the object to the center of the screen, and width is the arc length occupied by the object in the player’s field of view.

Distance angle was calculated as follows:

$$D = \arctan\left(\frac{d}{d'}\right)$$

where d' is the depth of the target, away from the player, and d is the deflection distance in world units. Target width angle was calculated as follows:

$$W = 2 \arctan\left(\frac{w/2}{d'/\cos(D)}\right)$$

where w is target width in world units and D is angular target deflection distance.

Target conditions

The 72 target conditions were generated as a result of all the different combinations of target width, distance, and direction parameters. Discrete values (as opposed to random sampling from a continuous range) were used for the parameters in order to enable statistical analysis such as ANOVA. Angular units are used for the 3D task, while Cartesian world units are used in the 2D task. All targets were located at a depth of 448 world units away from the player. The following is a summary of factors in target positioning.

Direction Direction indicates the direction that the participant had to move the mouse to acquire the target, starting from the base view position. There were six different values of direction used: 0° , 45° , 135° , 180° , 225° , and 315° , where a direction of 0° corresponds with aiming directly to the right. This parameter was used in order to vary the direction of mouse movement, and because it has been shown that angle of approach affects the prediction ability of Fitts’ Law [91, 65], the directions chosen are balanced across axes. Directions corresponding to directly upward or downward

movement were omitted in order to decrease the number of possible combinations and thus the length of the experiment for purposes of minimizing fatigue. By including the angles at 45° angles from the horizontal axis, I still included the vertical axis in the experiment.

Distance Target distance determines how far a participant had to move the mouse to acquire the target. In 2D, it is the distance in world units away from the center, while in 3D, distance was the deflection angle away from the initial view direction. There were four values for distance, with the maximum value chosen as the largest distance that fits within the room and is visible from the base view direction. The remaining three values were chosen by subdividing the farthest position evenly into five values, with the smallest resulting value being discarded to avoid an overly easy condition. In 3D, this subdivision was performed in the angular unit system, after calculating the angular deflection for the farthest target. In Figure 3.2, distance is depicted by the four concentric circles centered around the reticle.

Width Target width is the diameter of the target sphere or circle. In 2D, it is the diameter of the circle in world units, with a maximum size chosen manually to result in reasonable sizes and resulting ID values. In 3D, it is the angular width (i.e., the angle that a participant would have to move their view by in order to move across the extents of the sphere). The maximum width in 3D was chosen by calculating the value that resulted in an equal width-to-distance ratio (in angular units of deflection) as the equivalent condition in the 2D version of the task. This ensured that the *Index of Difficulty* is equal in both dimensionalities. The remaining values were calculated by evenly subdividing the width of the largest target. Three values of width were used.

Target distance and width values were aggregated into an *Index of Difficulty* (ID) value for each combination, which was used in results analysis. See Section 3.4 below for the ID calculation, and Table 3.2 for a detailed breakdown of distance and width values for all possible Index of Difficulty values.

The distance and width parameters in the 3D task are similar to those in the 2D task, but the algorithm to calculate the positions and sizes of the target must be different, such

Table 3.2: Distance and width values for all possible values of ID in the target acquisition task.

ID	2D		3D (Cartesian)		3D (angular)	
	Distance	Width	Distance	Width	Distance	Width
2.949	68.32	10.17	101.8	15.28	12.8°	1.905°
2.124	68.32	20.33	101.8	30.56	12.8°	3.81°
1.696	68.32	30.5	101.8	45.87	12.8°	5.715°
3.47	102.5	10.17	156	15.78	19.2°	1.905°
2.595	102.5	20.33	156	31.56	19.2°	3.81°
2.124	102.5	30.5	156	47.36	19.2°	5.715°
3.852	136.6	10.17	214.7	16.52	25.6°	1.905°
2.949	136.6	20.33	214.7	33.05	25.6°	3.81°
2.454	136.6	30.5	214.7	49.6	25.6°	5.715°
4.154	170.8	10.17	280	17.57	32.01°	1.905°
3.233	170.8	20.33	280	35.15	32.01°	3.81°
2.722	170.8	30.5	280	52.74	32.01°	5.715°

that calculating the Index of Difficulty in each unit system results in the same ID values. Since game engines operate in a Cartesian unit system, ultimately the 3D target positions and sizes must be set in world units as well. However, because aiming in 3D involves rotating the view rather than panning, it is not possible to simply subdivide the distances and sizes of the 3D targets in world units because such an approach would result in ID values inconsistent with those in the 2D case. For example, the magnitude of the aiming movement required to acquire a target with a distance of 100 world units is less than twice as large as the movement required to acquire a target at a distance of 200 world units.

Furthermore, a constant scaling factor was applied to the target positions and sizes in the 2D task, with the purpose of making the effective distances and sizes of the target visually similar on the screen while not affecting the resulting ID value. Without the scaling, targets in the 2D task appeared much larger than in the 3D task. See Appendix B for the algorithm used to make consistent target parameters between 2D and 3D tasks.

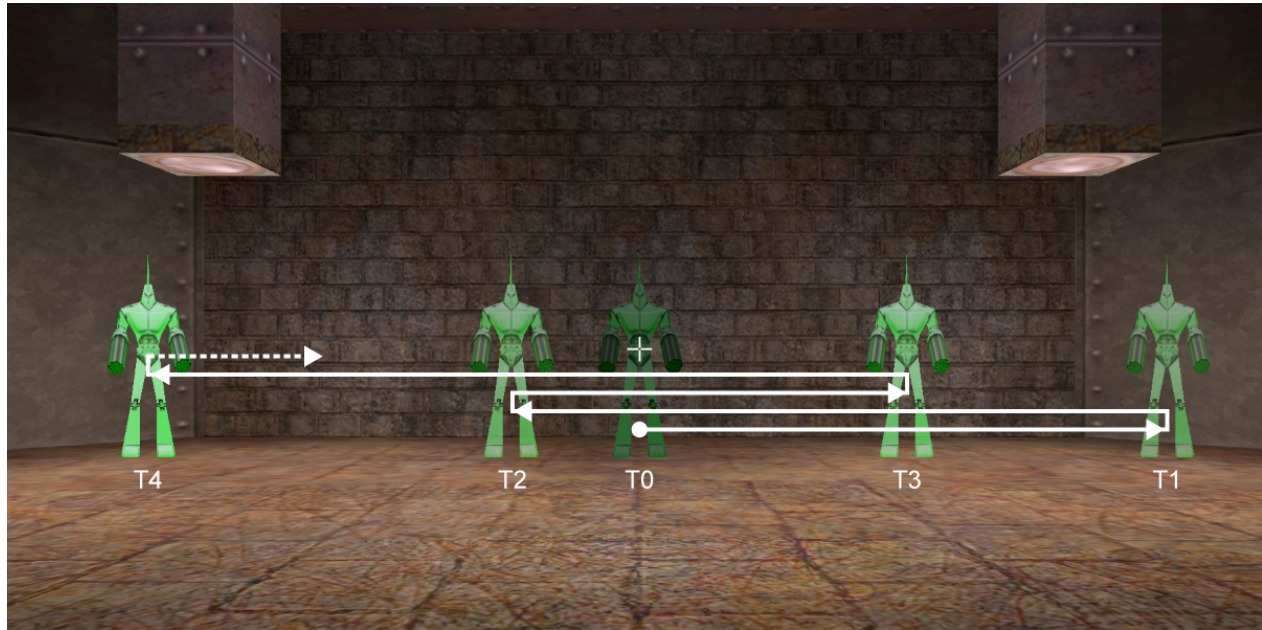


Figure 3.3: Target tracking task showing target movement from initial position (T0) through four direction changes (T1-4).

3.2.2 Tracking

In addition to the target acquisition task, a target tracking task was also performed by participants. Tracking a moving target is a core part of aiming in first person shooters. In a game, a player would typically acquire the target first, then track it while it potentially moves evasively until it is eliminated. Participants performed tracking separately in order to isolate this phase of aiming and identify the effects of lag on it alone.

In each tracking trial, a target appeared in the same initial position, directly in front of the player, centered in the area that the target had to move within. When the participant was ready to start a trial, they aimed at and clicked on the target to begin. The target immediately started moving side-to-side in a pseudo-random, evasive pattern (Figure 3.3). Each trial lasted for a period of four seconds, during which the participant was required to do their best to keep their aim within the target's bounding box. Whenever the player's aim was on the target, the target was tinted green to indicate successful tracking as feedback. Once a trial was completed, the target reset to the initial position, ready for the next trial. The dependent measure for evaluation was the amount of time that participants successfully aimed at the target during a trial.

The target was humanoid in appearance, with attributes similar to many first person shooters. Hit detection was done using a bounding box with a width of 32 units, depth of 24 units, and a height of 56 units. The target’s side-to-side evasive motion occurred along one axis, corresponding to the width of the room, orthogonal to the player’s base aim direction. Target motion was linear, with evasion happening as direction changes which are equivalent to the target’s velocity vector being reversed.

The target accelerates to a maximum velocity over a period of 140 ms after each direction change. The timing of direction changes is predetermined, sampled from a uniform random number generator with a period of 0.3 to 0.7 seconds. This results in an average of ten direction changes per trial. The seed for the generator is kept constant so that every sub-block and participant experiences the same evasive pattern.

The only independent variable in the tracking task was target speed, which had two possible values: 240 and 320 units per second. Each block started with seven trials at the slower speed, followed by seven trials at the higher speed.

Wherever possible, the parameters chosen were based on an existing first person shooter - Quake III Arena. The bounding box size matches that of the player’s avatar in Quake 3, the acceleration time and evasive motion period was similar to that of a typical player in Quake 3, and the higher target speed of 320 units per second matches that of a player in Quake 3. The lower speed is similar to what would be found in slower-paced FPS games.

3.3 Differences Between 2D and 3D Conditions

The experiment was kept as similar as possible between the 2D and 3D conditions in the study, and thus the same game world and target models were used, but changes to aiming, lighting, camera, and targets were performed.

Aiming While moving the mouse rotates the view (effectively panning the background) in the 3D case while leaving the aiming reticle fixed in the center, in 2D, moving the mouse moves the aiming reticle while the world and background remains fixed.

Lighting The graphics engine’s lighting system was disabled for the 2D conditions, leaving the environment fully lit in order to remove any 3D cues.



Figure 3.4: Cropped view of the tracking task in the 2D environment (top) and the 3D FPS environment (bottom).

Camera Instead of using perspective projection to show the appearance of a 3D world on the screen, orthogonal projection was used for the camera instead, removing the three-dimensional appearance.

Targets As previously mentioned, in the acquisition task, care was taken to preserve equal Index of Difficulty values between the 2D and 3D conditions of the task, which resulted in some differences between target positions and sizes. In the tracking task, no such changes were needed because no modeling or regression was performed.

Figure 3.4 shows the visual differences in the acquisition task, while Figure 3.5 shows the differences in the acquisition task.

3.4 Performance Metrics in Acquisition

The target acquisition task in the study is set up to be equivalent to an ISO 9241-9 [42] standard task for evaluating input devices. In both tasks, circular targets of varying sizes and distances appear around a central location in various directions. The ISO task is a formulation of Fitts' Law [57], which is widely used and recommended in human computer interaction studies [84].

In this task, targets are classified by *Index of Difficulty* (ID), which represents the difficulty of the pointing task conditions, in bits:

$$ID_e = \log_2 \left(\frac{D}{W_e} + 1 \right)$$



Figure 3.5: Cropped view of the acquisition task showing a sequence of frames from one trial (left to right) in the 2D environment (top) and the 3D FPS environment (bottom).

where D is the distance to the target, and W is the width of the target – both parameters are equivalent to those in my target acquisition task. W_e is the effective width, adjusted for accuracy. The range of ID covers those commonly encountered in real FPS games, with the exception of instances where targets are very close or very far from the player.

Although I am primarily interested in trial completion time (i.e., the time taken to aim at the target and click on it), I am also interested in *Throughput* (TP), as defined by the aforementioned ISO standard, which measures the bandwidth of the pointing condition:

$$TP = \frac{ID_e}{MT}$$

where MT is the movement time (i.e., trial completion time).

Throughput is immune to the speed-accuracy tradeoff [58] due to the adjustments for target width based on accuracy, represented by the adjusted width term, W_e :

$$W_e = \begin{cases} W * \frac{2.066}{z(1 - Err/2)} & \text{if } Err > 0.0049\%, \\ W * 0.5089 & \text{otherwise} \end{cases}$$

where W is the target width, Err is the error rate, and z is the normal distribution quantile function (qnorm in environments such as the R language).

Although I use the accuracy-adjusted width in the throughput analysis, the non-adjusted version is used when setting up target conditions, and is reported in Table 3.2. This was done because adjusted ID must be calculated after data is gathered, since it uses error rate or standard deviation, which precludes being able to have consistent, discrete levels for ID. Furthermore, ID_e must be calculated per condition, rather than aggregated for all participants.

3.5 Participants

Twelve participants were recruited from the local university (9 male, 3 female). All participants had previous experience at FPS games using a keyboard and mouse, and all were right handed. Mean weekly computer usage for participants was 42 hours. Seven participants rated themselves as being in the top 25th percentile of skill level in FPS games.

3.6 Study Design

Within each session, all conditions for one task (i.e., acquisition or tracking) were completed first, followed by all conditions for the other task. Participants were given up to a five minute break between the two tasks to rest, although no one took more than one minute.

Trials were grouped into blocks, with each block containing all possible conditions for each task (72 for acquisition, 14 for tracking). Blocks were used to increase the total number of trials, to monitor for practice effects, and to disperse the various conditions throughout the experiment. There were five blocks in each dimensionality condition, with the first two being used for training, during which every second trial was skipped in order to reduce the total training length. A short break was given to participants between blocks.

In order to give participants a chance to get accustomed to the tasks, procedure, mouse sensitivity, and the rest of the experiment, each dimensionality condition began with two blocks of training, at which time participant performance did not contribute to data analysis. Participants were also allowed to ask questions during this time, and they prepared for the non-training period.

All conditions were counterbalanced, with half the participants starting with the acquisition task and the other half starting with the tracking task. Also, half the participants started with the 2D tasks and the other half started with 3D tasks.

My hypotheses were as follows:

H1. Acquisition time would change with dimensionality.

H2. Tracking time would change with dimensionality.

3.6.1 Experimental Conditions Summary

The experimental factors were as follows:

Dimensionality. Represents whether the task was the 2D or 3D version.

Block. The block number, with three identical blocks being performed for each task and dimensionality combination, and one additional training block performed at the start of each.

Index of Difficulty (ID). The index of difficulty component of the Fitts' Law relationship as part of the acquisition task. With 4 different possible values for distance and 3 for width within each dimensionality condition, there are 12 resulting combinations of index of difficulty. However, there were only 10 unique values of ID due to some distance/width combinations equating to the same values of ID.

Speed. The target speed in world units per second as part of the tracking task. There were 2 levels used.

The target acquisition study used a 2x3x10 within-participants RM-ANOVA with factors *Dimensionality*, *Block*, and *ID*. The dependent measure was trial completion time.

The target tracking study used a 2x3x2 within-participants RM-ANOVA with factors *Dimensionality*, *Block*, and *Speed*. The dependent measure was time on target.

3.7 Results

Outliers were further removed by filtering out the worst 1% of trials for each participant within each condition. Outliers were only removed on the slow end of the distribution because it was not possible for a participant to get an unusually good trial time through methods such as anticipating a timer. The slow end was removed because participants were observed to sometimes pause briefly during a trial due to reasons such as talking or ergonomic adjustments.

3.7.1 Acquisition

Although I tracked error rate, trials with errors (target misses) were still included in the trial completion time analysis rather than being filtered out. This was done because, in real games, misses occur frequently during the acquisition process, and the total time taken to shoot at a target regardless of misses is typically the most important metric. That is, the primary concern is which player is eliminated first in an encounter.

Main effect of Dimensionality

Trial completion time included trials with misses. RM-ANOVA showed a significant effect of *Dimensionality* on trial completion time ($F_{1,11} = 15.8, p < 0.01$). This effect can be observed in Figure 3.6 as the overall performance gap between 2D and 3D, with 2D having a shorter trial completion time on average. Post-hoc paired t-tests showed a significant difference in task time ($p < 0.05$) between 2D and 3D at all ID levels except at $ID = 1.70$ and $ID = 2.45$. I therefore accept **H1**—acquisition time changed with dimensionality.

Interaction between Dimensionality and ID

RM-ANOVA also showed an interaction effect between *Dimensionality* and *ID* on trial completion time ($F_{9,99} = 2.65, p < 0.01$), suggesting that dimensionality has a different performance effect as ID changes. By looking at Figure 3.6, it can be observed that 3D acquisition performance degrades quicker as ID rises.

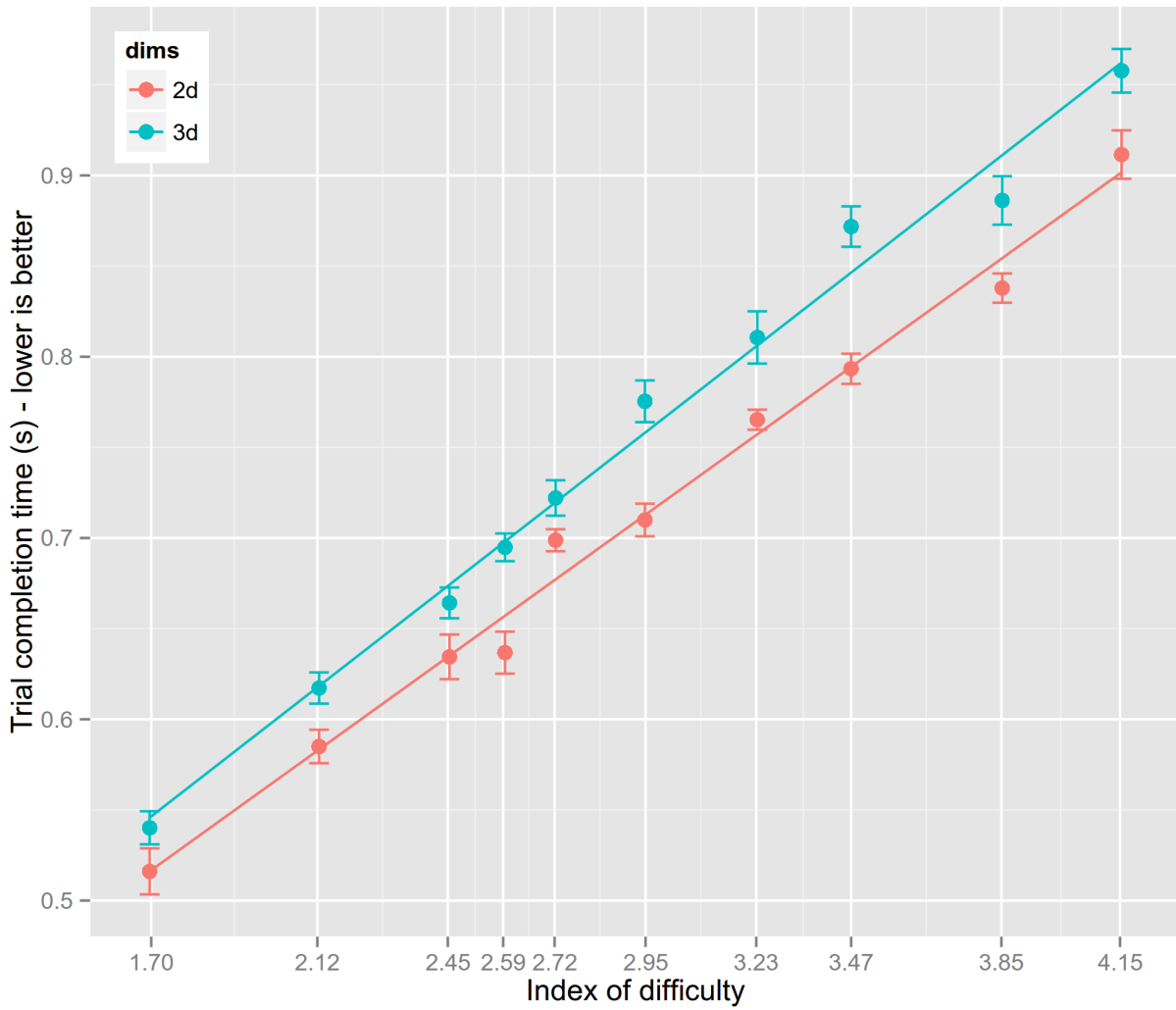


Figure 3.6: Index of difficulty versus trial completion time.

Table 3.3: Target acquisition ID versus trial completion time regression results.

Dimensions	y-intercept	slope	R^2
2D	0.250	0.157	0.99
3D	0.258	0.170	0.99

Regression analysis

I performed a linear regression to determine how task completion time changed over varying levels of *ID*. The results can be seen in Table 3.3.

Results show that the regression y-intercept is very similar between 2D and 3D, while task completion time grows approximately 8% faster in 3D compared to 2D. The regression has an excellent fit to the data with an R^2 of 0.99 in both dimensions. The fact that a linear regression fits well with the data confirms that Fitts' Law is well suited to 3D pointing when angular units are used.

Error Rate

There was a significant difference in number of errors per trial between the 2D and 3D conditions (t-test, $p < 0.05$). In the 2D conditions, the mean number of errors per trial was 0.74, while in the 3D conditions, the mean number of errors per trial was 0.65.

Throughput

The accuracy-adjusted throughput (or index of performance) was 4.14 in 3D and 4.26 in 2D. This is approximately a 3% difference, compared to the 8% difference found when looking at just trial completion time with errors rolled in. However, the difference of 3% was not statistically significant ($p \approx 0.24$).

Both values found for throughput are similar to those found previously [59] for pointing with a mouse.

Table 3.4: Summary of RM-ANOVA main effects and interactions on acquisition time for the acquisition task.

Effect	DFn	DFd	F	Significance
Dimensionality	1	11	15.8	Yes
ID	9	99	321	Yes
Block	1	11	0.737	-
Dimensionality : ID	9	99	2.86	Yes
Dimensionality : Block	1	11	1.44	-
ID : Block	9	99	1.57	-
Dimensionality : ID : Block	9	99	2.52	Yes

Summary of Effects

Table 3.4 shows a summary of main effects and interactions analyzed with RM-ANOVA, including both significant and non-significant effects on acquisition time.

3.7.2 Tracking

Performance in the tracking study was determined by the mean time successfully spent tracking the target per trial, with maximum possible time being 5 seconds.

Main effect of Dimensionality

RM-ANOVA showed no significant main effect of *Dimensionality* on tracking performance ($F_{1,11} = 0.136$, $p = 0.07$). I therefore reject **H2**. Figure 3.7 shows the relationship between target speed and time on target for both 2D and 3D environments.

Interaction effect between Dimensionality and Speed

RM-ANOVA showed a significant interaction effect of *Dimensionality* and *Speed* on time-on-target performance ($F_{1,11} = 10.8$, $p < 0.01$), although post-hoc t-tests show no significant differences between 2D and 3D performance at either target speed.

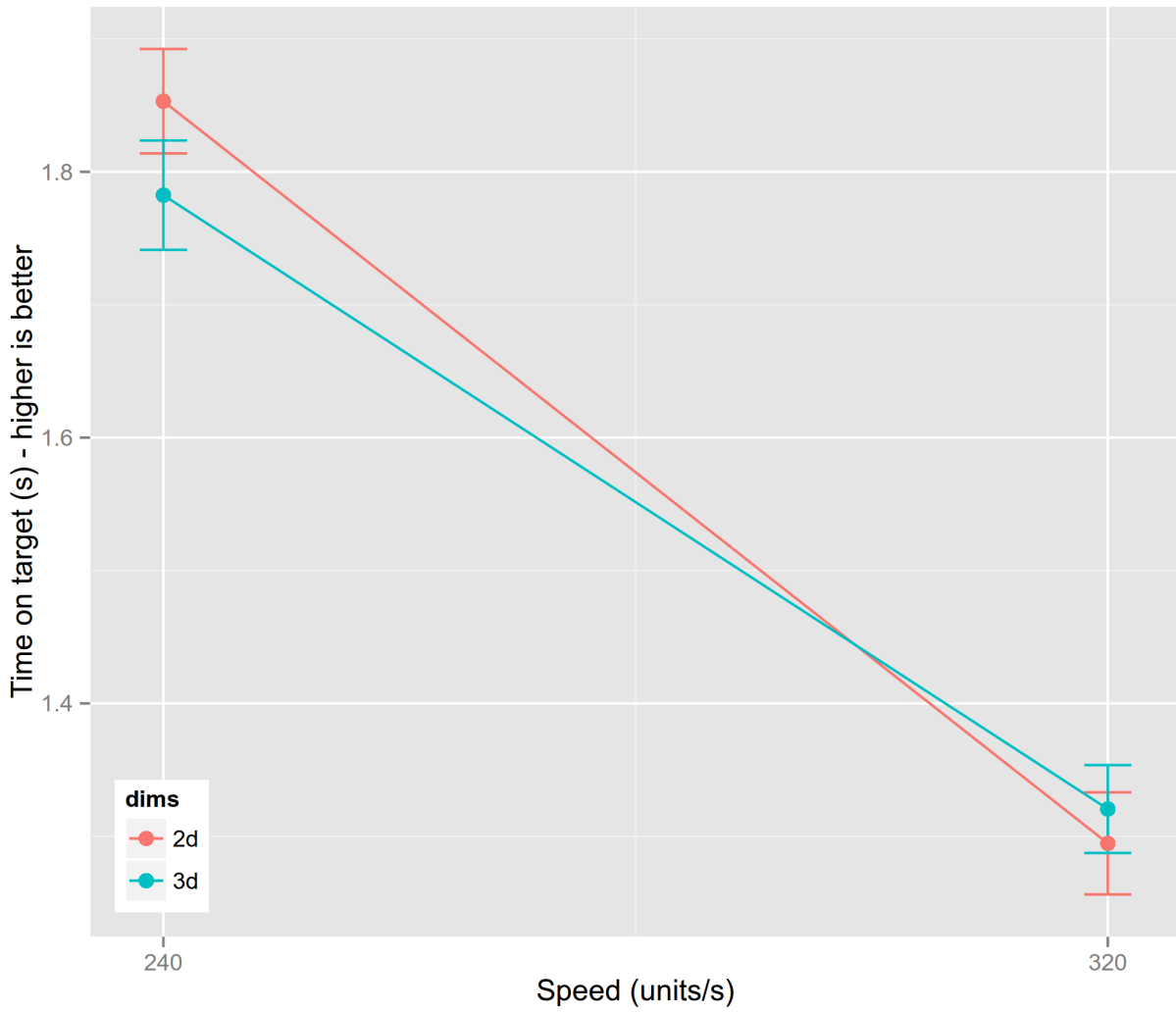


Figure 3.7: Target speed versus time on target.

Table 3.5: Summary of RM-ANOVA main effects and interactions for the tracking task.

Effect	DFn	DFd	F	Significance
Dimensionality	1	11	0.14	-
Speed	1	11	594	Yes
Block	1	11	2.37	-
Dimensionality : Speed	1	11	10.8	Yes
Dimensionality : Block	1	11	3.00	-
Speed : Block	1	11	0.02	-
Dimensionality : Speed : Block	1	11	4.85	Yes

The interaction suggests that although dimensionality does not affect tracking performance on average, it does have an effect once target speed is considered, with faster moving targets becoming more difficult to track in 2D environments compared to 3D ones. This interaction effect can be observed in Figure 3.7.

Summary of Effects

Table 3.5 shows a summary of main effects and interactions analyzed with RM-ANOVA, including both significant and non-significant effects.

3.8 Discussion

The experiment examined the performance differences between equivalent tasks in two and three dimensional settings, separated into target acquisition and target tracking tasks. To summarize the difference between the conditions, the 3D conditions used a perspective-transformed view that emulates how we see the real world, while the 2D conditions used orthographic projection (i.e., parallel lines are preserved). Also, mouse movement directly controlled the movement of the reticle in the 2D conditions, with the background staying fixed, while mouse movement rotated the view in the 3D conditions, which, in effect, moves the background while keeping the reticle fixed.

The performance metrics that I focused on do not in themselves explain the specific mechanical differences that give cause to the overall difference in performance, but the results do indicate that there is a significant difference in how aiming is performed between 2D and 3D conditions. In the acquisition task, the results show that movement time was a statistically significant 8% longer in the 3D condition compared to the 2D condition across a range of index of difficulty values. In the tracking task, there was no significant main effect, but an interaction effect with target speed was found between the 2D and 3D conditions.

In the acquisition task, throughput [84] was not statistically significantly different between 2D and 3D. This result does not invalidate the difference in task completion time between the conditions; instead, the low difference in throughput is explained by the difference in error rate between the two conditions (0.74 errors per trial in 2D versus 0.65 in 3D) since throughput makes use of error rate as a method of finding the effective width of targets. That is, it takes into account the effective spread of targeting locations used by participants relative to actual target width. In the 2D condition, the error rate was higher, meaning that participants were effectively aiming at a larger target (i.e., the area that their clicks covered was larger), since errors mean clicking in an area larger than the real target. This higher 2D error rate results in a lower throughput score for the condition, but the important outcome is that aiming in 2D behaves differently than in 3D, as evidenced by movement time and error rate differences.

My study was the first to use difficulty-matched targets comparing 2D and 3D settings in otherwise very similar conditions and environments. Although a study by Looser [54] was similar in the goals used, mine provided important additional changes (see below). My study shows that there are differences in aiming between 2D and 3D settings, and these differences are one of the motivating factors for the usefulness of my study described in Chapter 5. Although it would be useful to examine how latency affects first person shooter performance regardless of the work in this chapter, the 2D/3D differences I found indicate that there may also be reasons that latency could affect performance in FPS differently than it does in 2D pointing scenarios. Therefore, it should not be assumed that the effects of latency will be the same in FPS as in the existing work on latency in 2D, and it is worth evaluating the effects separately.

3.8.1 Performance Differences Between 2D and 3D

In the following sections I discuss possible explanations for the differences I found between 2D and 3D, and contrast my work with previous studies of targeting in 3D. While I do not currently have a concrete explanation for the differences, there is currently external work underway to examine these differences in detail and find the causes.

Target Acquisition

There are several possible reasons for the reduced performance in the 3D condition. The first, and perhaps most likely, reason to explain the difference is that visual differences between the two conditions may cause performance losses – that is, the dramatically-larger amount of optical flow (i.e., visual movement) in the 3D condition requires more perceptual processing. This explanation may also account for the increasing difference between 2D and 3D as the index of difficulty rises since smaller targets may be more likely to be lost in the visual movement.

As detailed in Section 2.1.2, controlled pointing movements are performed through means of one or more constituent submovements, beginning with a ballistic open-loop movement that is programmed to end at the target. Because stochastic neuromotor noise may cause the initial movement to miss, the initial movement may be followed by corrective movements that make use of visual feedback to correct for errors and arrive at the target [64]. Between each submovement, visual feedback is used to evaluate the error and plan a corrective action. Therefore, any mechanisms that delay the visual feedback or the amount of time necessary to process and react to it would also delay the total movement time. In the 2D condition, the target location is fixed on the screen, with user eye gaze normally being focused on the target throughout the motion. This allows for easy and rapid comparison of cursor location to the target location, since there is relatively little movement on the screen that requires neuro-sensory processing. In contrast, for the 3D condition, the entire screen changes in content when movement is performed, and the visual information must be processed by the brain. A visual search [93] must be performed to find the location of the target on the screen before a corrective movement can be planned. In 2D, visual search is less likely to be needed

because the cursor is unlikely to be lost during movement – users can perform continuous tracking of the cursor in the peripheral vision. Also, kinesthetic feedback can be used as well [46] which would complement the visual feedback.

The second reason that movement time was different may be found in the throughput results analysis. The finding that throughput was not significantly different between the two conditions while the movement time and error rates were suggests that users likely made use of a different level of speed/accuracy tradeoff between the two conditions. That is, participants were faster on average as a result of being more risky and making more errors. However, trials with errors were included in movement time analysis – if a trial had an error, it would take more time than normal to complete because users would have to realize their mistake, make another correction, and finish the task. Despite these slower error trials counting in movement time analysis, movement time was still quicker in 2D.

A third possibility is that the FPS environment I used (like most FPS games) is a perspective projection of a 3D world, as opposed to a true 3D visual experience. The lack of some 3D cues, such as stereopsis or parallax, may have caused perceptual conflicts that degraded performance. For example, participants may have misjudged the target location, distance, or depth due to cue conflicts, hindering their targeting ability. To explore this possibility, in future work I will test performance with a true 3D display that provides both stereo vision and parallax.

A fourth reason for the 2D/3D difference is the potential for visual-motor conflict in the 3D task. In traditional 2D cursor movement, the user’s hand and mouse movement is aligned with the direction of cursor movement on the screen. The optical flow information on the screen is therefore in the same direction as the movement of the hand that can be seen in one’s peripheral vision. In 3D FPS cursor movement, however, the 3D world pans behind the central reticle, and the user’s hand and mouse movement is opposite to the movement of the world movement on the screen. The optical flow information on the screen is in the opposite direction as the movement of the hand that can be seen in one’s peripheral vision. This is a version of the stimulus-response compatibility hypothesis [49], with lower levels of compatibility (or naturalness) being associated with slower reaction times.

Yet another reason for the difference could be that humans are used to direct, one-to-one

physical control of limb position in space. Moving the cursor through means of a mouse is similar to direct manipulation – the cursor directly reflects the physical motion of the mouse, albeit with a control-to-display ratio (i.e., gain) that is likely not at a one-to-one ratio. In the 3D condition, mouse movement is mapped non-linearly to aim control, because moving the mouse results in a rotation of the view – that is, the relationship between target distance from the cursor on the screen is non-linear with respect to the amount of input movement required. This could be seen as another form of stimulus-response incompatibility.

Finally, another potential contribution to the 3D condition being slower is due to perspective distortion of distal targets. As pointed out by Zaranek [97], targets that are close to the edge of the screen appear larger than they are due to perspective distortion. This illusion may interfere with planning the initial ballistic submovement in acquisition. The user’s movement would be planned with a seemingly easier to hit target in mind, therefore reducing the effectiveness of the ballistic submovement and requiring additional corrective followup motions.

Regarding the differences in error rate, a possible reason for the finding is that users are more familiar with 2D environments when using a computer, such as when experiencing standard desktop usage through web browsing or such. Rotational 3D environments are comparatively more rare. As a result, users may have been more confident in their aiming abilities in 2D and took more risks by prioritizing speed over error rate in the speed/accuracy trade-off. This more risky behavior could also explain the difference in movement time, since the priority would be placed on speed rather than accuracy.

As future work, there are a number of interesting possibilities for digging deeper into the potential visual differences between the 3D and 2D conditions that contribute to the differences in aiming performance that I found in this study. Individual 3D depth cues (such as parallax) from the 3D case could be isolated and evaluated for their effect on aiming performance. Likewise, the visual information in the background texture could be systematically reduced in order to lessen the effect of optical flow. Finally, in a wholly 2D environment, one could test shifting the background under a centered cursor, versus the traditional setup of moving the cursor over a fixed background. A key tool in finding the underlying differences in mechanics in either of these proposed conditions would be tracking the location of aim

at all times, in order to analyze where the performance difference is located. For example, 3D aiming might involve longer pauses between submovements due to the aforementioned differences in visual and processing load required. Another possibility would be that less time is spent in the ballistic portion of movement and more corrective actions are required due to perspective distortion.

Target Tracking

Tracking time on target was not significantly different between the 2D and 3D conditions. However, there was a significant interaction effect with target speed between 2D and 3D. It is likely that this interaction is a result of a combination of fast targets being very difficult to track and the inherent differences in tracking targets once the target has moved significantly away from the center position, especially for box-shaped targets.

The tracking task in the experiment was very difficult to perform well, particularly in the fast target condition which represents target movement in fast-paced first person shooters. Participants were successful in tracking the target for only approximately 1.5 seconds out of a 5 second trial. Participants often made frantic guessing movements trying to anticipate target movement rather than being able to properly track it. Consequently, anything that would help the participants stay on target longer despite their inability to do so would benefit the faster target case more than the slower one. It is likely that the nature of tracking this particular type of movement in 3D is what presented this aid in the 3D conditions.

When the target is centered at the start of a trial, motion is perpendicular to the vector between the player and the target. In effect, rotating the view in 3D to track the target in this position is similar to moving the cursor in 2D – approximately the same amount of input movement is required for a given amount of target motion. However, since the target moves along the same vector or "rail" the entire time, as the target approaches the periphery of its movement range, tracking in 3D requires comparatively less movement. This is because a significant portion of the target movement vector moves it toward or away from the player, which requires no compensatory tracking movement, and the portion of the vector that requires tracking compensation becomes less significant. The target does effectively appear smaller as it moves away from the center of its movement, which makes it harder to track and

offsets the benefit of requiring less movement. However, the target is normally box-shaped in an FPS game (and in the study), and the exposed profile of the box becomes larger as the target moves away from center since more of the side of the box is exposed. Therefore, the target size does not decrease as quickly as the magnitude of the required tracking motion does.

The effective growth of the target profile as it moves away from the center could be avoided in the future by using a cylindrical hit area for the target rather than a box, or by having the target constantly adjust its rotation so that the front face of it is directed toward the player. I did not do this in my study, partially because it is a common aspect of aiming in 3D and also because the fixed-rotation target is the approach used in a similar study [54].

3.8.2 Comparison with Previous 3D Studies

The most relevant previous study that can be compared to my work is that of Looser [54]. I see my study as building on this previous result, rather than conflicting with it. Looser used a similar approach to mine, in that regression was performed on the Fitts' Law index of difficulty versus movement time results in both the 2D and 3D studies. However, the results of Looser's study were different than my own; Looser found a very similar slope of regression between the two conditions, with the main difference being the y-intercept. This is unlike my results, where I found a very similar y-intercept but a different slope. See Table 3.6 for a comparison of results.

It is worth noting that my study confirmed that Fitts' Law is suitable for use in a 3D environment, even though I used a 2D version of the task as described by ISO 9241-9 [42], unlike Looser. The suitability of Fitts' Law is evidenced by the excellent fit of a linear regression to my results ($R^2 = 0.99$). The fit seems to hold both on the low and high end of ID values used in my study.

It is interesting and surprising that Looser's results show such a wide discrepancy in y-intercept between the two conditions. Since a trial starts as soon as a target appears, the intercept is a measure of reaction time including the time needed to prepare for the movement. My results show an intercept of 250 ms, which is in line with typical human visual reaction time [50], with the 3D case being very slightly higher, likely due to the additional complexity

Table 3.6: Target acquisition ID versus trial completion time regression for both my study and Looser’s.

Dimensions	Ivkovic		Looser	
	y-intercept	slope	y-intercept	slope
2D	0.250	0.157	0.19	0.18
3D	0.258	0.170	0.46	0.19

in evaluating the projected distance of the object. However, Looser’s results, while showing a reasonable 190 ms intercept for the 2D case, show a 3D intercept of 460 ms, which seems unreasonably long. It is possible that their 3D environment was unusually difficult to process or that the target was difficult to quickly perform a visual search on.

There were several differences in the studies that help to explain the different overall results. Looser used a one-dimensional Fitts’ Law test to compare a traditional 2D environment to a 3D FPS environment. In the 2D case, fixed-width targets were presented at different distances from the starting cursor along the horizontal dimension of the screen (the targets were centered vertically on the screen). In the FPS case, fixed size targets were presented at different locations along the horizontal x-axis of the 3D world at a fixed depth, effectively presenting a rectangular target to the user. In contrast, I evaluated a 2D Fitts’ Law task where targets varied in size and location in both horizontal and vertical dimensions on the screen.

There were also differences in the setup of my experimental conditions. I attempted to present a consistent visual environment in both conditions. For example, the same background texture was used in both 2D and 3D. In the 2D case, I used an orthographic camera projection (to remove all perspective depth cues) and removed shading cues from the spherical targets so that they appeared circular. In contrast, Looser used conditions that appeared entirely different visually – the 2D task was devoid of background texture and used desktop-like features such as a normal mouse pointer, while the 3D task used a fully textured box-shaped room with humanoid targets and other depth cues. Additionally, there may have been differences due to the apparatus used. My experiment used a high resolution, high refresh rate display which may have elicited a different response than the comparably low resolution and

refresh rate used by Looser.

Finally, Looser chose an index of difficulty range that spanned the same range as my study, but were inconsistent between conditions, therefore only regression could be compared. I designed targets with matched IDs between the conditions so that I could perform an ANOVA in addition to regression. This allowed us to look for interaction effects, which is an important extension.

CHAPTER 4

LATENCY IN REAL-WORLD SYSTEMS

Although gamers share evidence when discussing game system latency [51], and significant marketing is employed to promote "low latency" input devices and displays, little empirical evidence exists for the local latency associated with different gaming setups. Minimizing latency in VR headset displays has been a recent focus of attention for new products, such as the Oculus Rift,¹ which includes a device to measure latency from "motion-to-photon" while the device is in use. However, efforts to apply such measurement methodologies to typical FPS gaming systems have been uncommon.

In this chapter, I show a sample of the latency of existing real-world systems. In order to do this, I performed a study in which I sampled sixteen different systems comprised of various games and hardware. Due to limited time and resources, my goal in the study was to find a reasonable range of latencies that can be found in various classes of systems, rather than performing a large-scale study of many systems where each factor in the system could be analyzed separately.

4.1 Methodology

I built my own latency measurement apparatus in order to canvass a range of typical gaming setups and measure the amount of latency faced by gamers in real-world situations. This apparatus was also used to measure latency in my study reported in Chapter 5. my system measures the latency between the beginning of a physical input device movement and the resulting visible motion on the screen. I used simple descriptive statistics to report my findings.

¹<http://www.oculusvr.com>

4.1.1 System Selection

I chose the sixteen sampled conditions by measuring systems that were easily accessible to us. These included systems from the University's Interaction Lab, the author's own system, and systems of people known to the author. I attempted to find a representative variety of conditions by varying factors when possible, such as exchanging the input device or game in the same system. I did not make any special attempt to find combinations with the lowest or highest possible latency. Instead, I used the hardware and software that was already commonly used on those systems.

I selected systems from a variety of different types including gaming and office PCs, gaming consoles, and different types of input and output devices. Measurements were performed on a number of different games, using the same system settings that would be used during normal gameplay without modification. All displays had their "game mode" enabled in the on-screen settings.

4.1.2 Apparatus and Measurements

The measurement apparatus used a high speed video camera (Canon ELPH 300HS digital camera, recording video at 240 frames per second) mounted on a tripod to capture the motion of the input device. This allowed both the input device and the display to be captured clearly in the camera frame at once.

To obtain a measurement, I ran the game system with a highly visible textured scene to allow for easy observation of movement on the screen. With the camera recording, I tapped the input device sharply with a hammer to elicit a change in view direction with as little input transition time as possible (i.e., ramping up to a high velocity as quickly as possible to avoid ambiguity). The input device was elevated to the height of the display and kept steady by placing it on a flat wooden board located closely to the front of the display.

I measured the amount of latency by reviewing the recorded footage on a frame-by-frame basis, counting the number of recorded frames elapsed between the input device initially starting to move and the display changing, with each recorded frame having a period of 4.2 milliseconds. An LED light or highly contrasting marker mounted on the device helped to

show exactly when the movement started.

The reported latency period ended when any updates related to the input movement were first seen on the screen. Note that this could mean that only a small portion of the screen has updated at that point. For most systems I measured, the display operated at a refresh rate of 60 Hz, therefore it would take up to an additional 8 ms for half the screen to update and 16 ms for the rest of the screen to fully update, potentially slightly increasing the effective latency experienced by players.

my camera makes discrete measurements with a precision of 4.2 ms (i.e., the frame period length) and the displays used on the measured systems can have some jitter on their latency. For these reasons, I report the average latency from 4 measurements for each of the real-world systems, as well as the standard deviation of the measurements.

4.2 Results

I used my latency measurement apparatus to determine the latency of a number of real-world systems. Table 4.1 shows a summary of the results, with mean latency and standard deviation reported in milliseconds. Unless otherwise indicated (such as for the Overlord and BenQ displays), the displays receive video input at 60 frames per second. Systems, input devices, and displays are classified broadly as described in the legend at the bottom of the table.

The table is displayed with each system condition numbered and separated into groups. Systems 1-8 are PC desktops, while systems 9-16 are gaming consoles. Systems 1-5 are independent PC systems with no grouping. Numbers 6-8 are performed on the same desktop PC, with two different monitors and mice. Numbers 9-12 are the gaming consoles on the same TV, but with different games. Finally, systems 13-16 share a TV as well, but on different games.

Figure 4.1 shows the mean latency of the systems with points associated with gaming consoles shown as brown circles, while points for PC systems are shown as blue diamonds.

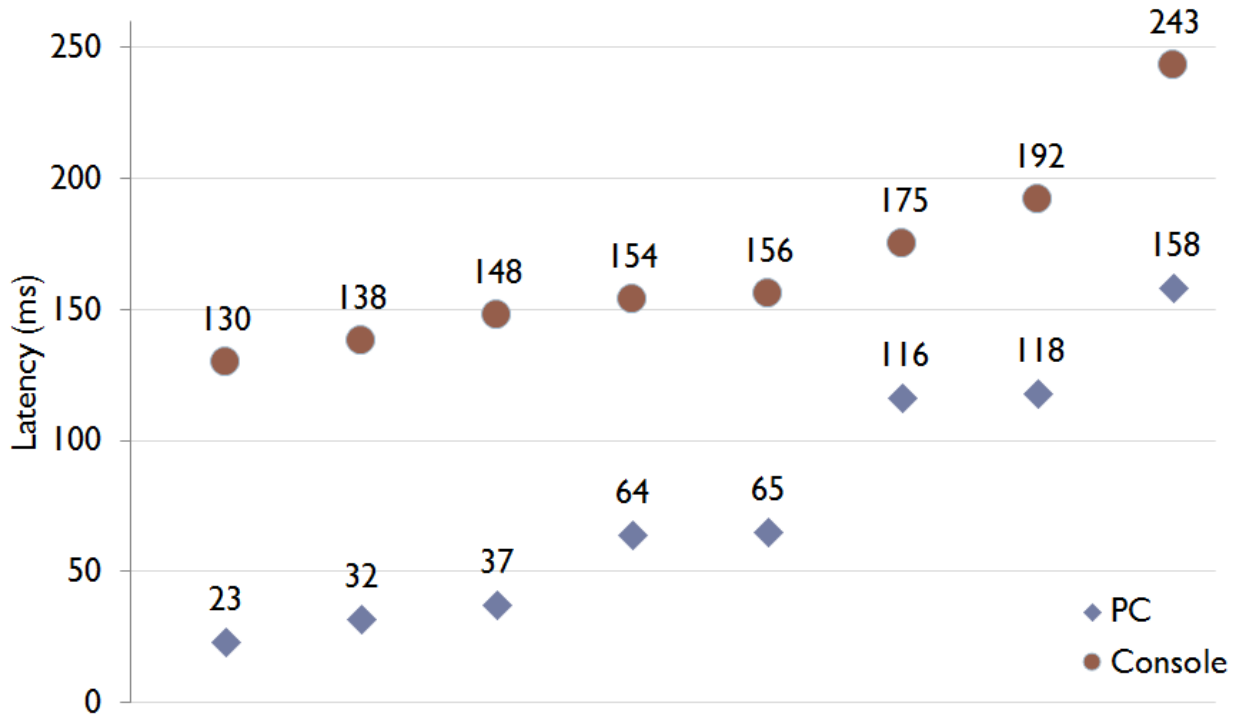


Figure 4.1: Scatter plot of latency results with a distinction between PC and console systems.

4.3 Discussion

Quantifying the results I found regarding the range of latency that can be found on various types of systems help to select the latency levels used in my study in Chapter 5, and together with the results of that study, I show that latency is a substantial problem in real-world gaming. Although there is a limited number of data points for each type of condition, some interesting observations can be made from the results in this study.

4.3.1 Latency Summary

I found a wide range of latencies from 23 ms on a PC with an ultra-fast gaming monitor, up to a surprising 243 ms for the best-selling game GTA5, even on a relatively low-latency TV. Although my measurements are not exhaustive, I believe they are a reasonably broad sample of real-world latency that covers a useful variety of conditions. I attempted to avoid bias by measuring a wide variety of systems that were of convenient availability to us, rather

than looking for systems that fit within certain desired parameters. The results match well with casual observations and measurements I observed in my experience with system latency before the study. PC systems ranged from 23 to 158 ms of latency, with a relatively even spread of measurements between them. Console systems tested at 130 ms at the low end of latency, up to 243 ms at the high end.

4.3.2 Attributing Latency to Components

Although data for the various types of system conditions is limited, in some cases it is possible to make observations within the pairs of conditions where only one factor changes between two measurements.

Displays

One of the largest sources of latency variance that I observed was due to the display used in the system configuration. In general, displays designed for gaming and low latency, such as those in systems 1 and 6, reached the lowest latency results. Standard computer monitors often have a low-to-moderate amount of latency, often in the range of 15 ms to 40 ms, although the latency can vary considerably [27]. Although my experimental conditions do not allow for isolating the latency contribution of HDTV television displays, latency databases such as DisplayLag [27] show that televisions often have a latency in the range of 40 ms to 80 ms, which is in line with my observations. This latency likely comes from the image processing algorithms and internal buffering mechanisms used by televisions, some of which are used even when "game mode" is enabled.

There were two cases in my conditions where direct comparisons between displays can be made:

- System 6 is a gaming desktop PC running the fast-paced Quake III Arena first person shooter using a gaming mouse and a ultra-high performance BenQ gaming display. The display boasts a low latency, low pixel response time, and high refresh rate as its performance features. This system achieved an average latency of 23 ms, which was the lowest observed in my study. The same system, with the display exchanged for

a standard LCD display (System 7), showed 64 ms of latency, which is a substantial increase.

- Systems 11 and 14 both ran on the Nintendo Wii U console, playing the Legend of Zelda: Windwaker HD game on two different televisions. In this case, the resulting latencies of 155 ms and 138 ms were not significantly different. The difference in latency is dwarfed by the high total latency in both systems.

Wired versus Wireless Mice

Wireless input devices can sometimes induce a non-negligible amount of latency into the system, although performance of different technologies varies. In my results, systems 6 and 8 shared the same conditions, except the wired gaming mouse in system 6 (Logitech G5) is swapped for a wireless mouse (Logitech MX 1100). This resulted in an increase of latency from 23 ms to 37 ms, which could make a difference in situations where low latency is critical. However, the difference is modest, and the MX 1100 is built on six year old wireless technologies. Modern wireless mice are likely to have less latency, which would result in an inconsequentially low difference in latency compared to that caused by many displays.

Game-Induced Latency

Although gamers tend to immediately think of hardware as the source of the local latency in a system, a surprisingly high amount of latency can be present as a result of the game itself, or the way that the game configures the system. There are a variety of reasons why this may happen. One very common way that games add latency is due to the necessary interpolation between frames that must be performed when game logic is decoupled from the game rendering system, which is a standard way to design game architecture. Other sources include input filtering/smoothing, rendering effects that require a buffer of multiple frames, the usage of vertical sync, and inappropriate timing of input sampling. See Section 2.3.2 in Chapter 2 for more information on these sources.

This game-induced source of latency can be clearly observed in the latency magnitudes of systems 15 and 16. The only difference in these systems is the game, where Battlefield 3

achieves a latency of 155 ms, and Grand Theft Auto 5 (GTA 5) achieves a surprisingly high 243 ms, which is almost 100 ms higher. Although the reasons for this difference cannot be discerned fully by an outside observer, one of the reasons is that GTA 5 runs at 30 frames per second, or less in some situations, while Battlefield 3 can achieve up to 60 frames per second.

PC versus Console

Although a clear and dramatic difference in latency can be observed between the PC and gaming console systems (Figure 4.1), my experimental conditions prevent us from making direct comparisons where all other factors are equal. However, some inferences can be made. One reason for consoles showing more latency is that they were all used with televisions as displays, which is the standard way they're used in practice, and televisions generally have more latency than computer monitors. Consoles also use wireless gamepads as input devices and likely use input filtering to achieve smooth aiming, which adds latency as well. Consoles almost always enable vertical sync in games, which is not necessarily the case in PC gaming, since players tend to be given the option of disabling it. Finally, PC systems can often run at high frame rates, which decrease the amount of local latency, while games tuned for consoles often target a rate of 30 frames per second.

Regardless of the reasons, a clear trend can be seen in real-world systems, with gaming console setups exhibiting a high amount of latency, in the range that can be perceived by many gamers. Since consoles are a popular platform for gaming, and because such setups are prone to having high latency, gamers would benefit if more attention was given to reducing latency.

Table 4.1: Results of field latency sample, with latency shown in milliseconds.

#	System	Game	Input	Display	Lag \pm SD
1	Linux ^(G)	Path of Exile	GM, W	Overlord X270OC 100Hz ^(I)	32 \pm 3
2	OS X 10 ^(S)	Zombie count	M, WL	Toshiba PA3769 (VGA) ^(T)	158 \pm 12
3	Win 7 ^(G)	Day Z	M, W	Asus VH242H ^(T)	117 \pm 7
4	Win 7 ^(S)	Cut the rope	M, W	Dell S2340Tt ^(I)	117 \pm 6
5	Win 8.1 ^(G)	CS: GO	GM, W	Dell UP2414Q ^(I)	65 \pm 4
6	Win 8 ^(G)	Quake 3	GM, W	BenQ XL2420T 120Hz ^(T)	23 \pm 2
7	Win 8 ^(G)	Quake 3	GM, W	LG L226WTX ^(T)	64 \pm 4
8	Win 8 ^(G)	Quake 3	GM, WL	BenQ XL2420T 120Hz ^(T)	37 \pm 2
9	PS4 ^(C)	Killzone SF	C, WL	Sony KDL55HX850 ^(V)	148 \pm 8
10	PS4 ^(C)	Watch Dogs	C, WL	Sony KDL55HX850 ^(V)	175 \pm 16
11	Wii U ^(C)	Windwaker HD	C, WL	Sony KDL55HX850 ^(V)	155 \pm 16
12	Wii U ^(C)	CoD Black Ops 2	C, WL	Sony KDL55HX850 ^(V)	130 \pm 8
13	Wii U ^(C)	Bioshock Infinite	C, WL	Sams. UN60EH6003F ^(V)	192 \pm 9
14	Wii U ^(C)	Windwaker HD	C, WL	Sams. UN60EH6003F ^(V)	138 \pm 17
15	Xb 360 ^(C)	Battlefield 3	C, WL	Sams. UN60EH6003F ^(V)	155 \pm 23
16	Xb 360 ^(C)	GTA 5	C, WL	Sams. UN60EH6003F ^(V)	243 \pm 60

Systems: G = gaming computer (performance CPU, discrete GPU); S = standard computer; C = gaming console.

Input: M = standard mouse; GM = gaming mouse; C = standard gamepad; W = wired, WL = wireless.

Displays: T = twisted-nematic; I = in-plane switching or equivalent; V = LCD televisions (IPS with game mode on).

CHAPTER 5

QUANTIFYING AND MITIGATING LOCAL LATENCY

In the previous chapter, I established that local latency is a pervasive aspect of real-world gaming systems, with it being observed in every tested system with widely varying levels. In this chapter, I examine and quantify how local latency affects aiming performance in first person shooters. I also demonstrate methods of mitigating latency to decrease its effect on performance. In order to accomplish these goals, I carried out a user study based on a modified version of my system used in Chapter 3, with the 2D condition removed and latency conditions added.

The study consisted of two separate sessions that were performed on consecutive days. In one session, participants performed the study without the mitigation assists in order to quantify how latency affects their aim. In the other session, the same tasks and procedure were performed, but with the mitigation assists being active. This two-session split allowed us to compare performance with and without the assists in order to ascertain whether the mitigation techniques are effective at reducing the performance impact of latency. Two sessions were used in order to keep the length of each condition reasonable and avoid excessive participant fatigue.

5.1 Controlling Local Latency

In order to find the effects of lag on participant performance, I devised a mechanism that allowed us to control the amount of local latency present in the system at any given time in a controlled manner. To accomplish this, I built a computer system with the lowest amount of latency that I could reasonably achieve, such that I could find a baseline level of performance when latency was as close to zero as possible. To reach higher levels of latency above baseline,

I purposely added latency to the system at the software layer, within the experimental game.

I chose to add latency purely on the output, delaying rendered frames from being shown on the display, rather than varying latency on both the input and output. This simplification is acceptable because there would be little difference between latency on the input versus the output within the experiment in terms of the effect on performance results. Also, in many real-world cases, the output side of the system is the dominant cause of latency.

Five levels of total system latency were used in the study: 11 ms, 41 ms, 74 ms, 114 ms, and 164 ms. See Section 5.6.1 regarding how latency levels changed throughout the experiment. The first level was the lowest achievable, while the highest was a reasonable high-end value based on my results shown in Chapter 4, although even higher levels can be found in real-world systems. The intermediate levels were not uniformly spaced because it is more valuable to find the effects of latency with finer granularity toward the lower end. Because it is already known that latency has a significant effect at the higher levels, it was useful to see if there were any threshold levels found and what the performance curve looks like at lower levels of latency.

Because the use of vertical sync (v-sync) on the display output causes additional latency, it was disabled for the baseline (lowest) level of latency, resulting in 11 ms of lag. However, vertical sync is commonly used in real-world gaming, and is necessary in order to avoid distracting screen tearing. Therefore, v-sync was used in the latency conditions above baseline. The second lowest level of latency in the experiment, 41 ms, was the result of enabling v-sync without purposely adding any more artificial latency on the output—it is the natural latency of the system with v-sync enabled. The final three levels of latency were the result of enabling v-sync, as well as adding latency artificially on the output.

The method used to artificially add latency was buffering rendered frames, then displaying them later. Each frame was rendered into a texture target contained in a ring buffer, where the buffer's size was one larger than the number of frames of latency desired. Instead of sending the current frame to the display after it was rendered, the appropriate previously rendered frame was retrieved from the buffer and displayed instead. For example, to add two frames of latency to the system (on top of the natural 41 ms with v-sync), the screen image rendered two frames in the past was displayed. Since the display's refresh rate in the

experiment was 120 Hz, this resulted in $41 + 2 * (1/120) = 74$ milliseconds of latency.

5.2 Tasks and Procedure

My experimental system implemented two primary tasks to be performed by participants: target acquisition and target tracking. These tasks are representative of primary activities in many 3D game environments; in FPS games, target acquisition is carried out when first encountering an enemy, and then a combination of acquisition and tracking is used until the enemy is eliminated. The entirety of the study consisted of these two tasks being performed in short and discrete trials with varying conditions. These tasks and procedures are similar to those found in Chapter 3. However, there are differences in design between the two studies and therefore the study in this chapter is described here in detail, rather than only the differences being contrasted.

All tasks and conditions were performed in a custom-made game world created using GtkRadiant¹. This world consisted of one room divided into two sections, with one section used for the acquisition task and the other used for the tracking task (see Figure 3.1). The player position was fixed to a location at the center, between the two rooms. The world had a graphical appearance very similar to *Quake III Arena*, which was a popular first person shooter.

5.2.1 Acquisition

Target acquisition is typically the initial phase of aim found in most first person shooters. It consists of the player moving their aim to a new location or direction corresponding to an intended target. In shooters, this target may be a specific entity such as an adversary combatant, or a ground location chosen in order to control space. In this study, I focus on the former type of target.

Each target acquisition trial within the study began with the participant's aim being set by the system to a base initial direction, aiming down range to the middle of the cluster

¹Level creation tool for creating Quake 3 compatible maps: <http://icculus.org/gtkradiant>

where targets appear. This aim direction was locked for a period of 500 ms at the beginning of each trial. During this time, the participant could not adjust their aim, ensuring that each trial began with a controlled initial aim direction, and the 500 ms pause allowed participants to prepare to aim at the target.

At the end of the 500 ms period, the trial timer started and the target appeared. The target was approximately spherical, in the shape of an ogre head (Figure 5.1). At this point, the aim lock was released, and participants attempted to aim at the target then click the left mouse button to shoot it. Participants were instructed to complete trials as quickly as possible while trying to be reasonably accurate. The main dependent measure was trial completion time, with lower times being better.

Participants were also told that the targets would start appearing red-tinted if accuracy dropped below a threshold, and to be more careful and accurate if this happened until targets stopped appearing red. The threshold was reached if accuracy fell below 80% at any point within a sub-block of trials (a group of trials with a constant latency level; see Section 5.6). If the participant shot but missed a target during a trial, the trial would continue until successfully completed, but it would be flagged as containing misses (errors) and repeated until completed without errors.

Each sub-block of trials consisted of 36 possible target positions within a virtual cone extending from the center of the base view position (Figure 5.1). The position order was randomized, but the same order occurred within every sub-block and participant. This was done by randomizing the order of the possible positions with a fixed random number generator seed. The targets were all visible on screen from the base view position without having to rotate the view.

Target conditions

The 36 target positions were generated as a result of different combinations of parameters related to the cone: angle, radius, and depth. Discrete, manually chosen values were used for the parameters in order to enable statistical analysis such as ANOVA.

Angle Angle indicates the direction that the participant had to move the mouse to acquire the target, starting from the base view position. There were six different values of

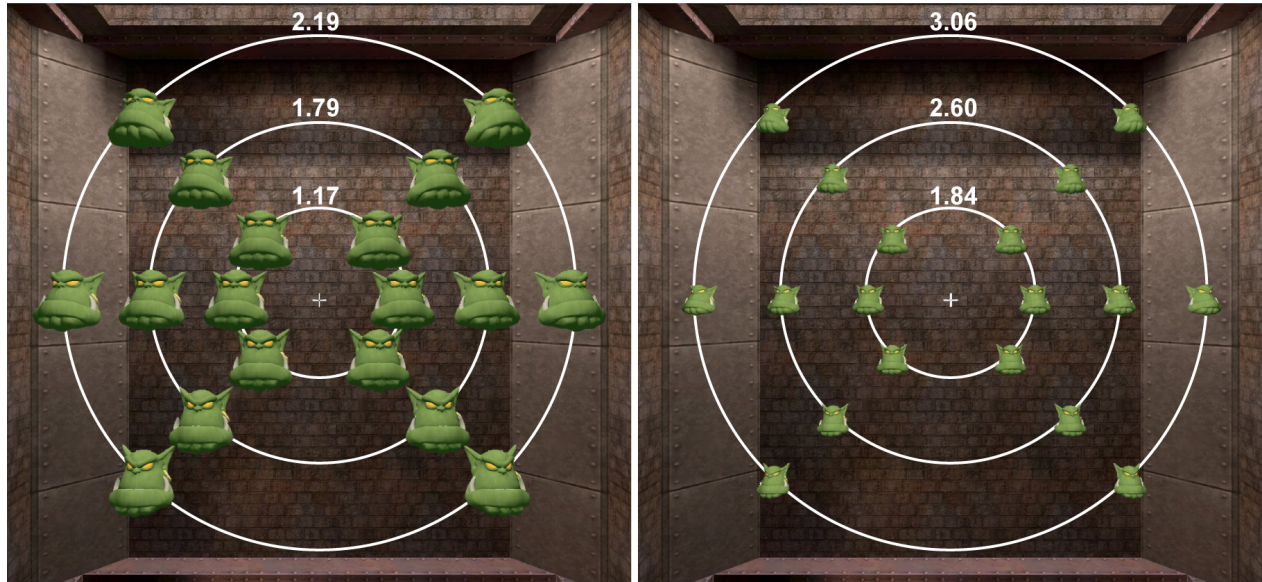


Figure 5.1: Target acquisition task showing all possible target locations and sizes. The six IDs (numeric labels) were chosen to be typical of targets in FPS games.

angle used: 0° , 45° , 135° , 180° , 225° , and 315° , where an angle of 0° corresponds with aiming directly to the right. This parameter was used in order to vary the direction of mouse movement, and because it has been shown that angle of approach affects the prediction ability of Fitts' Law [91, 65], the angles chosen are balanced across axes. Angles corresponding to directly upward or downward movement were omitted in order to decrease the number of possible combinations and thus the length of the experiment for purposes of minimizing fatigue.

Radius Radius indicates the *base radius* of the cone. It is the angular deflection away from the centered base view position, which determines how far a participant has to move the mouse to acquire the target. This parameter is similar to the distance factor in the Fitts' Law formula. Radius had three values: 74, 148, and 224 units. These were chosen by finding the largest suitable radius that is contained within the room and visible from the base position, then dividing it into three evenly spaced intervals (omitting the value of zero).

Depth Depth indicates the target's depth into the room, away from the player along the base view direction. Two values were used: 224 units and 448 units. Because the player position was fixed and there were no other moving objects in the world, depth could

also be taken as the size of the target, since a target that is farther away is effectively smaller from the player’s perspective. Thus, this parameter is analogous to the width parameter in the Fitts’ Law formula.

The various parameters for position and size were aggregated into an *Index of Difficulty* (ID) value, which was used in results analysis. The ID was calculated in a way similar to the procedure detailed in Chapter 3, using the Shannon formulation for ID [55]:

$$ID = \log_2 \left(\frac{D}{W} + 1 \right)$$

As explained in Section 3.2.1, D (distance) and W (width) were expressed in angles rather than Cartesian coordinates for the purposes of ID calculation.

5.2.2 Tracking

In addition to the target acquisition task, a target tracking task was also performed by participants. Tracking a moving target is a core part of the aiming task in first person shooters. In a game, a player would typically acquire the target first, then track it while it potentially moves evasively until it is eliminated. I performed tracking separately in order to isolate this phase of aiming and identify the effects of lag on it alone.

In each tracking trial, a target appeared in the same initial position, directly in front of the player, centered in the area that the target had to move within. When the player was ready to start a trial, they aimed at and clicked on the target to begin. The target immediately started moving side-to-side in a pseudo-random, evasive pattern (Figure 5.2). Each trial lasted for a period of five seconds, during which the participant was required to do their best to keep their aim within the target’s boundary box. Whenever the player’s aim was on the target, the target was tinted green to indicate successful tracking as feedback. Once a trial was completed, the target reset to the initial position, ready for the next trial. The dependent measure for evaluation was the amount of time that participants successfully aimed at the target during a trial.

The target was humanoid in appearance, with attributes similar to many first person shooters. Hit detection was done using a bounding box with a width and depth of 30 units,

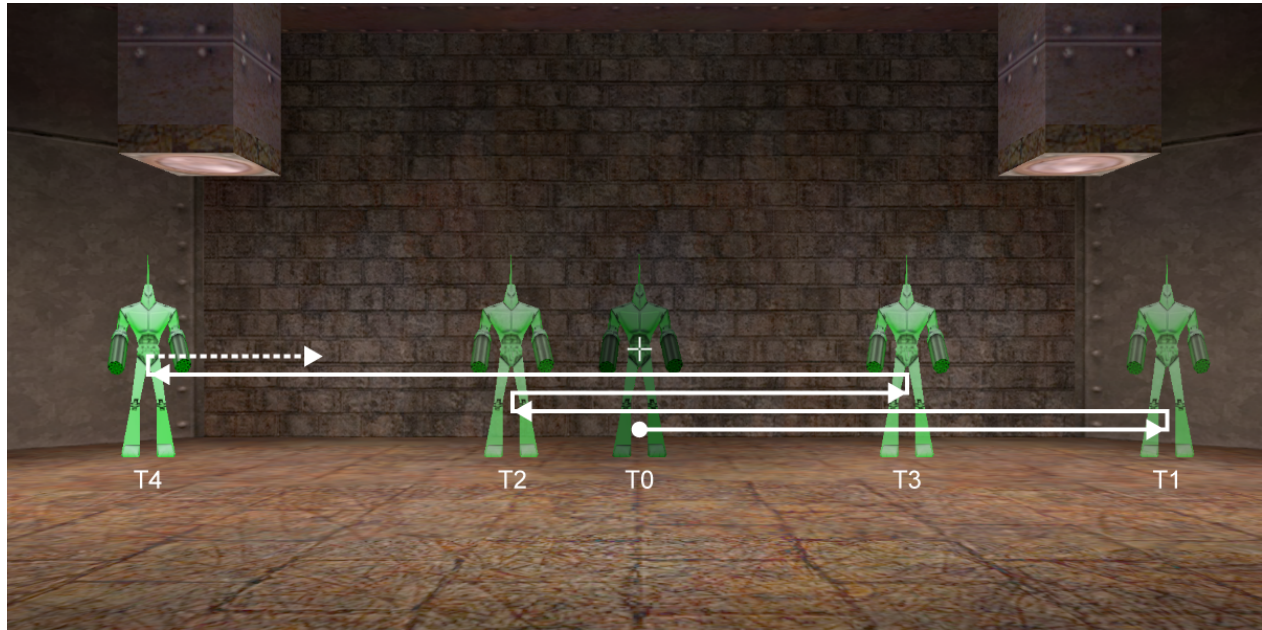


Figure 5.2: Target tracking task showing target movement from initial position (T0) through four direction changes (T1-4).

and a height of 56 units. These dimensions were chosen to match those of the player's character in the game Quake 3 Arena. Its side-to-side evasive motion occurred along one axis, corresponding to the width of the room, orthogonal to the player's base aim direction. Target motion is linear, with evasion happening as direction changes which are equivalent to the target's velocity vector being reversed.

The target accelerates to a maximum velocity over a period of 140 ms after each direction change. The timing of direction changes is predetermined, sampled from a uniform random number generator with a period of 0.3 to 0.75 seconds. This results in an average of ten direction changes per trial. The seed for the generator is kept constant so that every sub-block and participant experiences the same evasive pattern.

The only independent variable in the tracking task was target speed, which had two possible values: 240 and 320 units per second. Each sub-block started with two trials at the slower speed, followed by two trials at the higher speed.

Wherever possible, the parameters chosen were based on an existing first person shooter - Quake III Arena. The bounding box size matches that of the player's avatar in Quake 3, the acceleration time and evasive motion period was similar to that of a typical player in Quake

3, and the higher target speed of 320 units per second matches that of a player in Quake 3. The lower speed is similar to what would be found in slower-paced FPS games.

5.2.3 Questionnaire

At the beginning of their first session, participants filled out a computer-based questionnaire regarding their demographics information, including questions that have them self-assess their skill level in first person shooters, how much time they spend playing FPS games, and how much time they spend on their computer. See Appendix C for a full set of questions asked. The questionnaires were designed partly to get a sense of the demographics of the study, and partly to look for any correlations between participant performance or the effect of lag on each participant based on their experience level.

Additionally, between each change in latency levels (i.e., between sub-blocks) during the experiment, participants filled out a brief survey regarding their experience during the sub-block. Answers were given through a seven-point Likert scale [53] that ranged from "strongly agree" to "strongly disagree." The statements given were as follows (see Appendix D for the actual survey given to participants):

- "I performed well this round."
- "Lag affected my performance."
- "This round was frustrating."
- "The controls were laggy."

During the session in which latency mitigation aim assists were used, an additional statement was given along with the previous four (see Appendix E):

- "I noticed my aim being assisted."

Finally, a questionnaire was given at the end of each study session, with questions depending on the session. In the participant's first session, five questions were given, with the first and last question being asked as a yes or no answer, and the rest with an eleven-point Likert scale.

- "Were you aware of the existence of input lag before this study?"
- "How much do you feel that input lag has affected your performance in games in the past (apart from this study)?"
- "How much do you feel that input lag has affected your enjoyment of games in the past (apart from this study)?"
- "Has input lag affected your purchasing decisions in the past?"
- "Did you experience motion sickness during the study?"

During the second session, statements were as follows (see Appendix F), with answers in a seven-point scale:

- "Aim assists improved my performance."
- "I preferred having my aim assisted compared to the previous day's non-assisted experience."
- "I would want to see aim assists being used to mitigate lag in commercial games"

5.3 Compensating for Local Latency

Although quantifying the effects of latency on performance was the primary goal of the study, another important goal was to discover whether any methods could be used to effectively mitigate the effects of latency and restore performance to levels as close as possible to those of lag-free gameplay. Since aiming assistance is commonly used to improve performance on relatively low-performance or hard to use input devices such as gamepads (e.g., in many console FPS games such as *Call of Duty: Black Ops 2*²), and to balance gameplay between opponents of varying skill levels [89], it seemed suitable to use aim assists for the purpose of mitigating latency.

²<https://www.callofduty.com/blackops2> (Treyarch)

I developed a compensation technique to mitigate the effects of local latency on aiming using aiming assistance. In order to compensate for any reasonable amount of latency, the strength of the assistance was based on the current latency level of the system. Since effective forms of general aiming assistance already exist, I based my techniques on known methods, rather than inventing novel methods. However, finding a combination of suitable assists to use, tuning the weight of their input parameters, and evaluating their effectiveness is a valuable and novel contribution.

I devised my form of assistance so that it directly mitigates local latency by targeting the mechanisms through which latency affects player performance, rather than serving as a general form of player performance boosting. In order to create targeted compensation schemes, I identified two primary ways that local latency affects the two tasks.

Target overshoot during target acquisition.

During target acquisition, players tend to keep moving the crosshair toward the target until they observe that they have reached the target. If lag is present, players can overshoot the target (i.e., move aim too far past it) because the state observed on the display is behind the true game state. If the player fired as soon as they thought they were aiming at the target, they would actually miss due to overshoot.

Sticky targets[95] was chosen to compensate for latency in targeting because it directly combats the target overshoot effect by increasing the width of the target in motor space. Sticky targets reduces the mouse gain (also called *C:D ratio* [62]) while the player's aim is on the target, and thus it should decrease overshoot due to latency.

Direction changes during target tracking.

Each time a target changes direction while in motion, players have to change their aiming movement in order to continue to track it. Perceiving direction changes under local latency can take substantially longer; and once players do react and attempt to change their aim, the target has moved even farther in the other direction. Another aspect to tracking is predicting future target positions and when it may change direction, if a pattern is apparent. Latency interferes with this process by delaying feedback regarding incorrect predictions.

Aim dragging, an assist found in the *Call of Duty: Black Ops 2* game and many other first person shooters, was chosen to compensate for latency during tracking because it provides assistance in tracking these direction changes, reducing the amount of time spent off target. Aim dragging causes the player’s aim to be partially dragged along the direction of target movement. As in Call of Duty, aim dragging is active even if aiming near the target but not exactly on it. The decreased-gain aspect of sticky targets is also applied, which compensates for the fact that the tracking assistance could otherwise move the aim too quickly once added to the player’s own input, and also provides overshoot protection.

In order to develop a model of compensation and tune the relevant parameters, I performed an initial informal pilot study using two participants that did not participate in the full study. I began by implementing familiar mathematical functions that would result in an approximate curve shape that seemed appropriate, then I iterated on the function parameters experimentally. I did not iterate on my initial tuning of the weights of the parameters. If I had iterated, the compensation effectiveness would quite likely be improved for future studies. Table 5.1 shows the resulting assistance values at the latencies used in my experiment. The formulas used for compensation are as follows:

Sticky targets

$$s = \frac{1}{1 + 10l^{1.2}}$$

where s is the resulting multiplier to mouse gain (with 1 being no change in gain) and l is the amount of local latency, given in seconds. The effect is a roughly linear decrease in sensitivity up to about 100 ms, after which the effect decreases gradually to prevent the gain from becoming too low at high amounts of latency.

Aim dragging

$$d = 1 - \frac{1}{1 + 2.75l^{1.3}}$$

where d is the resulting strength of the drag, with 0 corresponding to no assistance and 1 being perfect assistance (auto-tracking), and l is the amount of latency in seconds.

Table 5.1: Assistance strength at the experimental lag levels.

Factor	41 ms	74 ms	114 ms	164 ms
<i>s</i>	0.82	0.69	0.57	0.47
<i>d</i>	0.10	0.17	0.24	0.31

Again, the effect is approximately linear at lower latency, with a gradually decreasing slope at high latency.

5.4 Participants

Eighteen participants were recruited from a local university (15 male, 3 female). All participants had at least some experience playing first person shooter games with a mouse as the input device. Every participant except for one was right-handed. The median number of hours spent using a computer per week was 52. There were 5 participants that did not play FPS games at the time of the study, but have played in the past. 7 participants rated themselves as being within the top 25th percentile of skill, while 4 participants rated themselves in the bottom 25%. The average self-assessed skill rating between participants, on a scale of 0 to 10, was 6.5. 8 participants primarily played shooters offline, while the most common type of multiplayer gameplay reported was in realistic, fast-paced shooters (such as the Call of Duty series). 4 participants had more experience playing games with a gamepad than a mouse.

Participants were awarded with a \$5 honorarium for their first session, followed by \$20 for the second session, in order to encourage them to attend both sessions.

5.5 Apparatus

The study was performed on a custom-built gaming PC (see Table 5.2 for specifications). A high performance gaming display with a high refresh rate of 120 Hz and 1 ms pixel response time was used in order to minimize baseline latency—such a display is often used in FPS

gaming competitions as well. A high performance gaming mouse polled at 1000 Hz was also used for minimum latency and smooth, responsive tracking. The experimental game ran at a constant 120 frames per second when vertical sync was enabled, and in the range of hundreds of frames per second without vertical sync; thus, the PC was more than capable of ideal performance within the experiment.

Mouse sensitivity (gain) was set such that it felt subjectively appropriate and acceptable to most participants. Participants were allowed to adjust sensitivity if it was significantly different than what they were used to, and two participants did decrease their sensitivity.

The experiment was performed in a small, closed, quiet room with minimal distractions and participants were asked to silence their mobile phones. Participants adjusted their chairs to an appropriate height and had ample space to move their mouse on a large mouse pad.

Other than surveys, the experiment was executed within a custom-built application made to resemble a first person shooter game, which guided users through each step of the study session (with assistance from the experiment administrator). It was developed in the C++ language using the Ogre3D open-source graphics library³ (version 1.8). Ogre3D was used to simplify OpenGL 3D graphics rendering, keyboard and mouse input, sample 3D models, and Quake 3 BSP file format loading and rendering. The BSP file capability was used such that a simple game world could be created using a Quake 3 level editor and easily used within the experiment.

5.5.1 Latency Measurement

I used the measurement apparatus as described in Chapter 3 to measure the amount of lag in each experimental condition. To assess the latency jitter and ensure that my measurements were accurate, I made 18 measurements at each of the lowest two latency levels and reported the mean and standard deviation of the samples. I also verified that the higher latency conditions with artificially added latency have the expected amount of additional latency. The mean baseline latency (no v-sync) was 11.4 ms (s.d. 2.28), and the first latency condition (v-sync on) was 40.8 ms (s.d. 3.13). While some frame-to-frame variation was present, it

³<http://www.ogre3d.org>

Table 5.2: Experimental PC specifications.

Component	Specification
Processor (CPU)	Intel Core i7 3.2 GHz
Video Card (GPU)	NVidia GeForce 460 GTX
Memory (RAM)	16GB PC1600 DDR3
Motherboard	Asus P8Z68-V LE
Storage	Intel 520 SSD, 120GB capacity
Mouse	Razer DeathAdder 3.5G, 1000 Hz sample rate
Display	BenQ XL2420TX, 120 Hz refresh rate
Operating System	Arch Linux 64-bit
Video Drivers	NVidia standard closed-source

was minimal.

5.6 Study Design

Within each session, all conditions for one task (i.e., acquisition or tracking) were completed first, followed by all conditions for the other task. Participants were given a several minute break between the two tasks to rest.

Since the entire experiment had to be completed both with and without latency compensation enabled, and each case required significant time to complete, participants performed the experiment in *two separate sessions* on two consecutive days, at approximately the same time each day. This was done to reduce participant fatigue and to keep sessions to a reasonable length of time. The total experiment length was approximately one hour per session, for a total of two hours.

5.6.1 Latency Levels and Blocking

In order to evaluate how participants perform under different levels, it was necessary to change latency as a factor over time during the experiment. Such a factor would normally

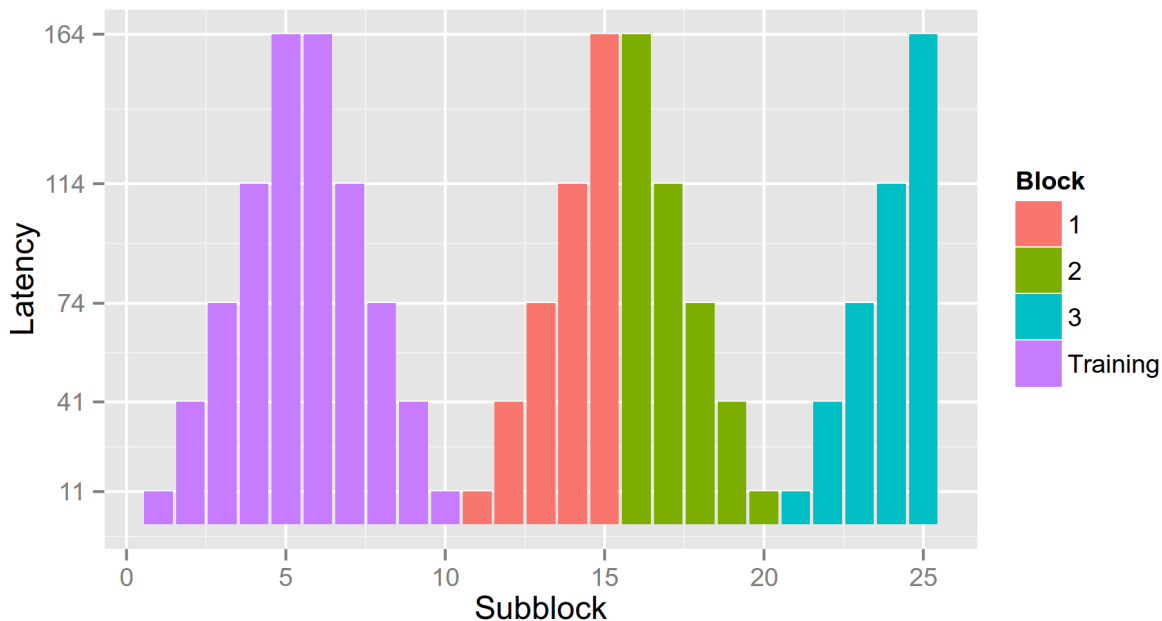


Figure 5.3: Latency levels over the course of a session, grouped into blocks.

be randomized and counterbalanced throughout the session. However, it was not reasonable to do so with latency, because an important aspect of latency in real-world systems is that it stays relatively constant, allowing players to adapt and to compensate for it to some degree. The study design was made with an attempt to account for this adaptation; therefore, latency levels in the experiment only ever changed to an adjacent level (e.g., 11 ms to 41 ms, or 114 ms to 74 ms).

In order to satisfy both counterbalancing (minimizing practice effects) and allow for participants to gradually adjust to latency, I grouped sequences of all five levels of latency into *blocks*, where each block contains a linear progression of latency level, either from minimum to maximum, or from maximum to minimum. Blocks alternated between increasing and decreasing progression, so that latency in each block started at the level that the previous block ended. See Figure 5.3 for a chart depicting the change of latency throughout the course of an experiment session.

Additionally, I used the term *sub-block* to refer to all the trials within a single, continuous latency condition. Therefore, since there were five latency groups per block, there were also five sub-blocks per block. A sub-block contained all 36 possible trials in the acquisition task,

and 8 trials in the tracking task (4 at each speed).

5.6.2 Training

In order to give participants a chance to get accustomed to the experiment and adjust to latency, each session began with two blocks of training, at which time their performance did not contribute to data analysis, and participants were allowed to ask questions. Latency during training began at the minimum level, progressed to a maximum, and then back to a minimum. There were no surveys given during training. During these training blocks, each level of latency only included half the normal number of trials (e.g., only two trials for each speed during tracking, rather than four) in order to decrease the total time taken for training while still allowing latency to gradually decrease to a minimum for the start of the non-training period. Additionally, each time latency changed, the first two trials in the acquisition task and one trial in the tracking task were used as training (unknowingly to the participant), giving participants a chance to briefly adapt to the new level of latency.

5.6.3 Counterbalancing

In order to counterbalance the compensation condition, approximately half of the participants started without compensation for their first session, and the other half started with compensation enabled. Due to a mistake in participant scheduling, counterbalancing was not performed perfectly: 10 participants started without compensation, while 8 started with compensation. Additionally, the order of tasks was fully counterbalanced, with half the participants starting with the acquisition task, and the other half starting with tracking.

5.6.4 Experimental Conditions Summary

Three factors were the same for acquisition and tracking tasks: *Lag*, *Block*, and *Compensation*. The tracking task also had a *Speed* factor for target speed, and the acquisition task had an *ID* factor for index of difficulty.

Lag. Represents the level of local latency experienced within a condition.

Block. As described in Section 5.6.1, block is a group of all possible conditions, including latency, within a session. Since latency compensation was varied across sessions, a block did not include both values for *Compensation*.

Compensation. This factor indicated whether latency compensation aiming assists were in use.

Speed. Used only in the tracking task, this factor represented the speed of the target.

ID. For target acquisition, six *IDs* were tested, consisting of combinations of target distance and cone radius.

5.6.5 Design and Hypotheses

The acquisition study used a 2x3x5x6 within-participants repeated measures analysis of variance (RM-ANOVA) with factors *Compensation* (off, on), *Block* (1-3), *Lag* (11, 41, 74, 114, 164 ms), and *ID* (1.15, 1.76, 1.81, 2.15, 2.54, 3.02). Dependent measures were trial completion time and number of errors.

The tracking study used a 2x3x5x2 within-participants RM-ANOVA with factors *Compensation* (off, on), *Block* (1-3), *Lag* (11, 41, 74, 114, 164 ms), and target *Speed* (240, 320 u/s). The dependent measure was time on target.

My hypotheses for the studies were:

H1. Tracking time on target would decrease with lag.

H2. Acquisition time would increase with lag.

H3. Acquisition errors would increase with lag.

H4. Lag compensation would reduce the effect of lag on tracking time on target.

H5. Lag compensation would reduce the effect of lag on acquisition time.

H6. Lag compensation would reduce the effect of lag on acquisition errors.

5.7 Results

5.7.1 Pre-processing

Before the data generated by the study was analyzed, several steps were taken to clean the data and remove outliers:

- All results for the tracking task for one participant were removed because it was discovered after that participant’s session that the system was incorrectly configured in such a way that it would invalidate those results – tracking latency mitigation was inadvertently changed to be much stronger preceding the session.
- Several participants stopped for an extended time during a trial, either to take a break or ask questions. These trials were removed by removing acquisition task trials that took longer than 3 seconds to complete, which removed a total of 36 trials.
- Outliers were further removed by filtering out the slowest 1% of trials for each participant within each latency level. Outliers were only removed on the slow end of the distribution because it was not possible for a participant to get an unusually good trial time through methods such as anticipating a timer. The slow end was removed because participants were observed to sometimes pause briefly during a trial due to reasons such as talking or ergonomic adjustments.

In total, 268 trials out of 21694 total trials were removed.

5.7.2 Acquisition

Although I tracked error rate, trials with errors (target misses) were still included in the trial completion time analysis rather than being filtered out. This was done because, in real games, misses occur frequently during the acquisition process, and the total time taken to shoot at a target regardless of misses is typically the most important metric.

Table 5.3: Trial completion time increase due to latency.

Latency	Movement time	Δ Movement Time	Significance
11 ms	620 ms	-	-
41 ms	630 ms	1.5%	-
74 ms	689 ms	11.0%	Yes
114 ms	798 ms	28.5%	Yes
164 ms	944 ms	52.1%	Yes

Effect of Lag on trial completion time

RM-ANOVA showed a significant main effect of *Lag* ($F_{4,60} = 246$, $p < 0.0001$), as well as a significant interaction effect between *Lag* and *Compensation* ($F_{4,60} = 18.7$, $p < 0.001$). In the non-compensated case, planned pairwise t-tests (Holm corrected) showed a significant difference between all levels of *Lag* ($p < 0.0001$), with the exception of the 11 ms and 41 ms pair ($p = 0.25$). I therefore accept **H2**.

Figure 5.4 shows the effect of latency on trial completion time (black lines), both with and without compensation. The effect of 41 ms of latency was negligible compared to baseline, although it becomes substantial and approximately linear at higher levels. Trial completion time increased by 1.5%, 11.0%, 28.5%, and 52.1% compared to baseline at 41 ms, 74 ms, 114 ms, and 164 ms, respectively (Table 5.3).

Effect of Compensation

The interaction between *Lag* and *Compensation* also suggests that compensation was effective at reducing the effect of latency. Figure 5.4 shows that compensation did decrease the effect of latency on performance, and follow-up paired t-tests (Holm corrected) confirm a statistically significant improvement at 74 ms, 114 ms, and 164 ms ($p < 0.05$). This effect was found both when comparing the compensated curve to baseline at each latency level, and when comparing within the compensated curve between the 11 ms level. I therefore accept **H5**.

However, planned pairwise t-tests showed that trial completion time was still significantly higher ($p < 0.05$) at 74 ms or more latency, even when compensation was enabled – this was

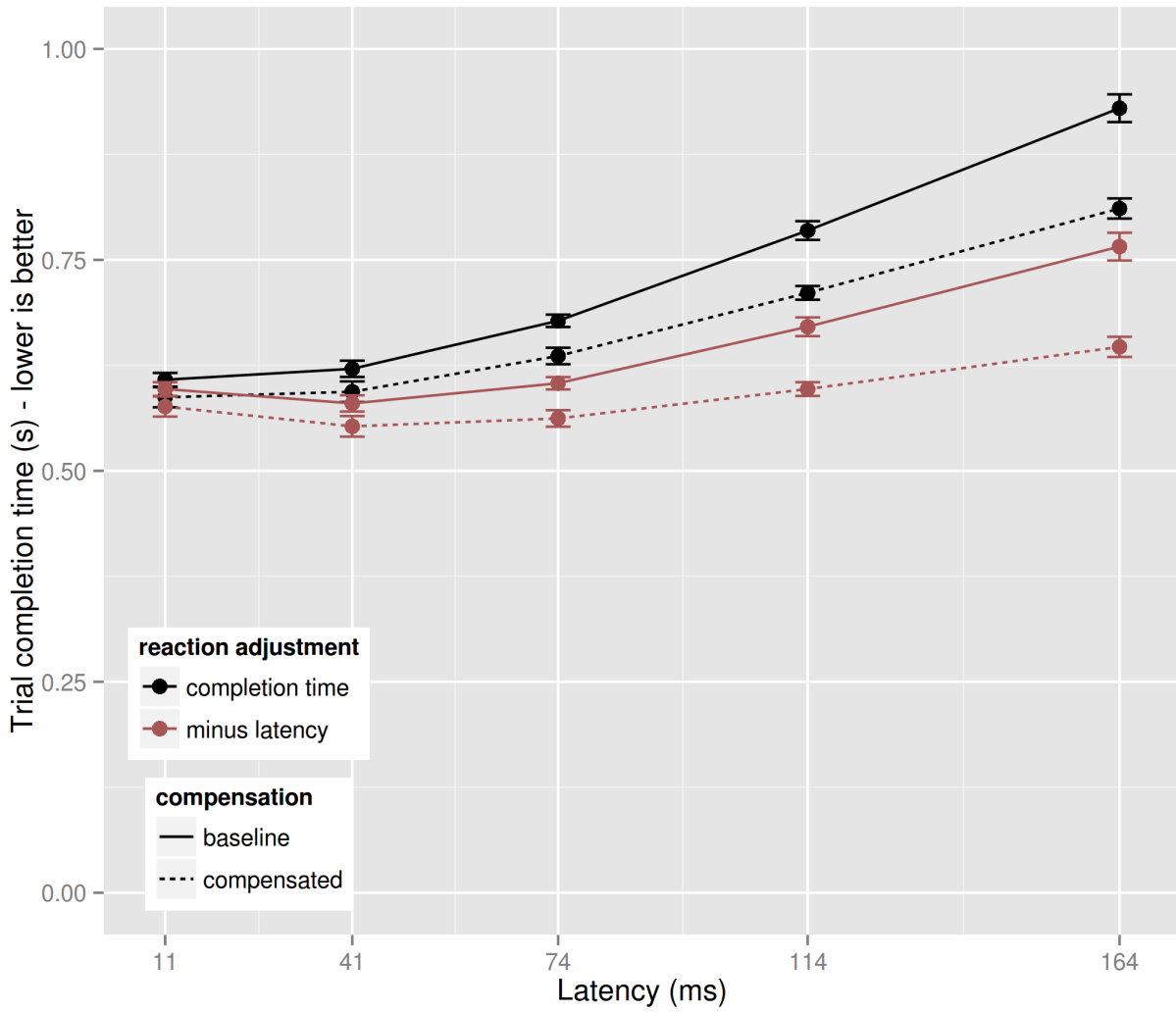


Figure 5.4: Effect of latency on targeting performance.

the same result found in the non-compensated condition. This means that my compensation technique was not able to fully eliminate the effect of latency, but it did decrease the performance effect.

Figure 5.4 also contains dark red lines depicting trial completion time when the amount of latency is subtracted from completion time. With lag, the participant will see the target appear later than when it actually gets created within the game state, and therefore reaction time will be delayed by the amount of latency present. All subsequent figures will display non-adjusted performance, (black lines).

Effect of ID

RM-ANOVA showed a significant two-way interaction between *Lag* and *ID* ($F_{20,300} = 8.87$, $p < 0.0001$), as well as a three-way interaction between *Lag*, *ID*, and *Compensation* ($F_{20,300} = 2.15$, $p < 0.001$). This three-way interaction suggests that lag has a different effect on acquisition performance depending on the index of difficulty of the acquisition task. Figure 5.5 shows that trial completion time actually did degrade at 41 ms of latency at the lower three levels of *ID*, even though (as I observed previously) there was no significant effect when not considering *ID*. This degradation effect was confirmed by a followup paired t-test ($p < 0.001$) at the three lower difficulties, although not at higher difficulties ($p > 0.5$). The figure also shows that compensation worked for all *ID* levels. The curves, which represent different levels of ID, all show less performance degradation with increasing lag in the compensated conditions when compared to the non-compensated conditions.

Figure 5.6 shows the same *Lag*, *ID*, and *Compensation* factors, but with *ID* on the x-axis. Since the curves are approximately linear, it appears that the acquisition task in the FPS environment follows Fitts' Law when angular distances are used. Note however the sharp increase in completion time between the very close second and third ID at higher latencies. These points are associated with a change in both radius and distance, reinforcing results found in previous studies [83].

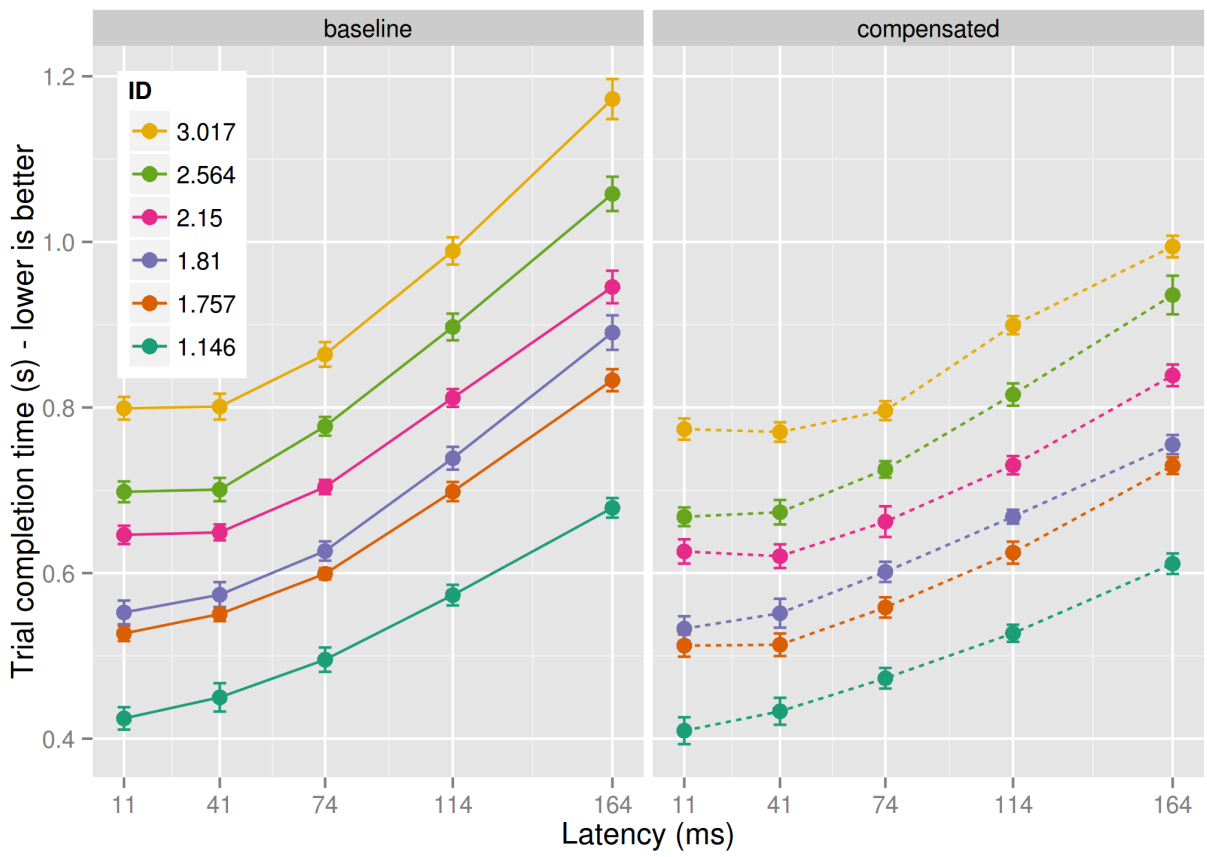


Figure 5.5: Effect of latency on targeting performance by ID and compensation.

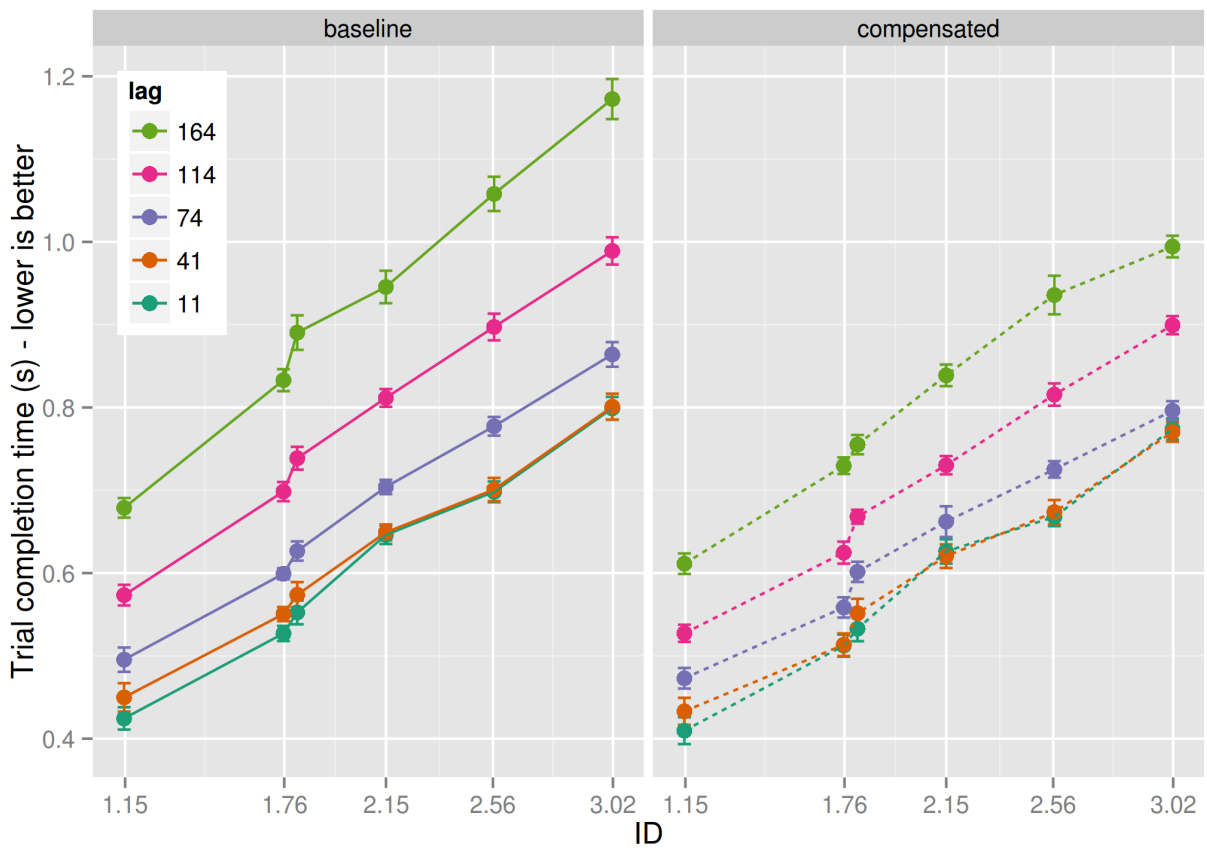


Figure 5.6: Effect of difficulty on targeting performance by latency and compensation.

Effect of Block

RM-ANOVA showed a significant main effect of *Block* ($F_{2,30} = 6.76, p < 0.05$), indicating that a practice effect was present. Participants did perform better as the session went on: trial completion time decreased by 1.5% in block 2 and 3.6% in block 3 compared to block 1. These effects are small, and because the occurrence of each latency level throughout the session is fairly balanced, there is likely no significant confound. Since there was no significant interaction between *Block* and *Lag*, this confirms that there was no significance of practice on the effect of latency on performance during the experiment.

Summary of Effects

Table 5.4 shows a summary of main effects and interactions analyzed with RM-ANOVA, including both significant and non-significant effects.

Error rate

I also examined the effect of *Lag* on target acquisition error rate (misses on targets). RM-ANOVA showed a significant interaction between *Lag* and *Compensation* ($F_{4,60} = 6.32, p < 0.001$) on error rate. See Figure 5.7 for a chart showing the change in error rate. Without compensation, paired t-tests show that error rate is lower at 41 ms than at 11 ms ($p < 0.01$), and that there is no significant difference between 11 ms and the other levels, so I must reject **H3**. With compensation enabled, paired t-tests show a significant decrease in error rate at 114 ms and 164 ms compared to 11 ms of lag, and no significant difference at 41 ms and 74 ms. I therefore accept **H6**. The decrease in error rate compared to the baseline suggests that my compensation technique could benefit from further tuning of the parameters to the assistance strength functions, since improving aspects of performance beyond the expected performance with no latency is undesirable.

5.7.3 Tracking

Performance in the tracking study was determined by the mean amount of time successfully spent tracking the target per trial, with maximum possible time being 5 seconds.

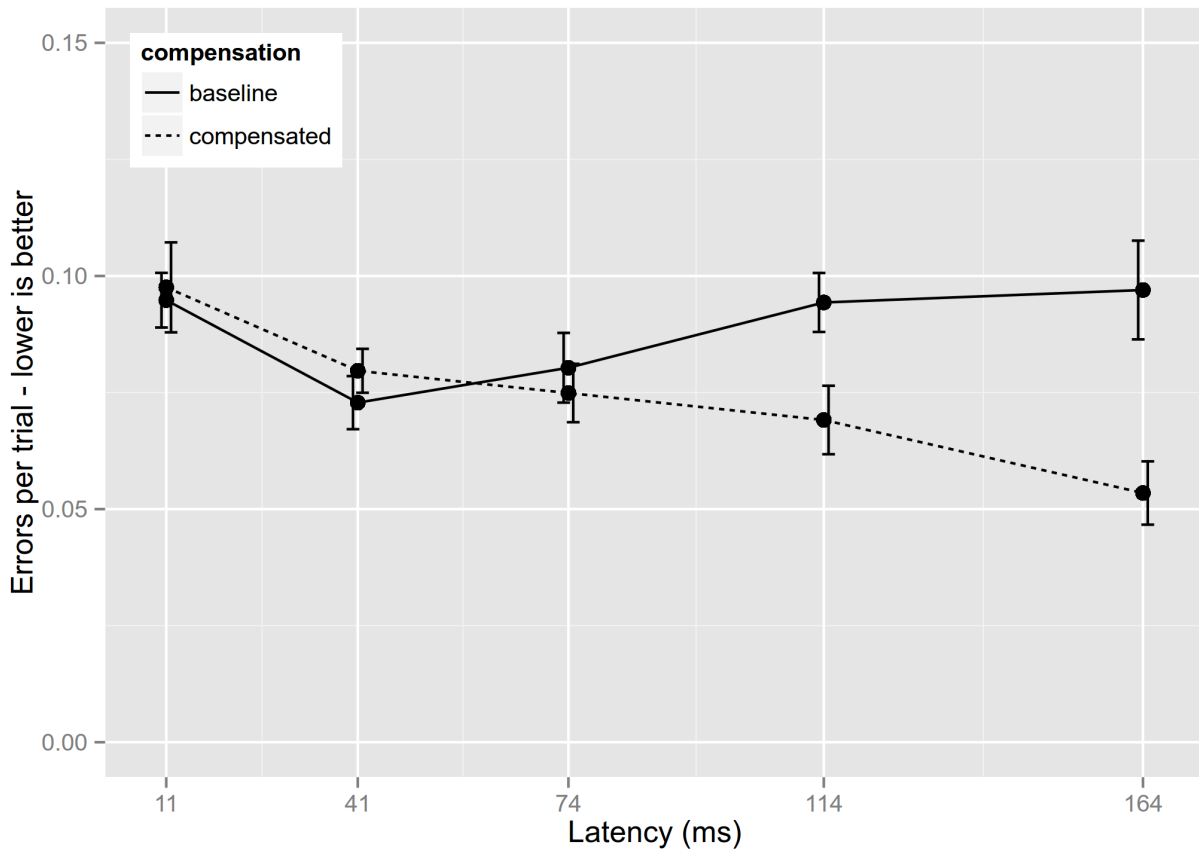


Figure 5.7: Effect of latency on error rate in target acquisition.

Table 5.4: Summary of RM-ANOVA main effects and interactions for the acquisition task.

Effect	DFn	DFd	F	Significance
Lag	4	60	246.1	Yes
ID	5	75	383.6	Yes
Block	2	30	6.8	Yes
Compensation	1	15	16.5	Yes
Lag : ID	20	300	8.9	Yes
Lag : Block	8	120	1.8	-
ID : Block	10	150	2.2	Yes
Lag : Compensation	4	60	18.7	Yes
ID : Compensation	5	75	5.5	Yes
Block : Compensation	2	30	0.2	-
Lag : ID : Block	40	600	1.4	-
Lag : ID : Compensation	20	300	2.2	Yes
Lag : Block : Compensation	8	120	1.7	-
ID : Block : Compensation	10	150	0.4	-
Lag : ID : Block : Compensation	40	600	1.0	-

Effect of Latency on tracking time

RM-ANOVA showed a significant main effect of *Lag* ($F_{4,56} = 39.7$, $p < 0.0001$), as well as a significant interaction between *Lag* and *Compensation* ($F_{4,56} = 46.2$, $p < 0.0001$). With no latency compensation, planned pairwise t-tests (Holm corrected) showed that there was a significant decrease in tracking time between each level of lag ($p < 0.01$). I therefore accept **H1**.

As shown in Figure 5.8, the performance decrease is approximately linear with increasing lag: 94.2%, 87.8%, 77.7%, and 67.3% of baseline latency performance at 41 ms, 74 ms, 114 ms, and 164 ms of latency (Table 5.5).

Table 5.5: Trial completion time increase due to latency.

Latency	Change in tracking time	Significance
41 ms	-5.8%	Yes
74 ms	-12.2%	Yes
114 ms	-22.3%	Yes
164 ms	-32.7%	Yes

Effect of Compensation

RM-ANOVA showed a significant main effect of *Compensation* ($F_{1,14} = 22.6, p < 0.001$) and an interaction between *Lag* and *Compensation* ($F_{4,56} = 46.2, p < 0.0001$), meaning that compensation was effective at reducing the effect of latency on tracking time. With compensation enabled, planned pairwise t-tests showed no significant difference in tracking time between any levels of *Lag* ($p > 0.2$). Planned t-tests also showed a significant difference between the baseline and compensated tracking times at all levels above 11 ms ($p < 0.01$), and although it appears that the 11 ms level may differ as well, t-tests show no significant difference. I therefore accept **H4**.

Effect of Speed

An interaction was found by RM-ANOVA between *Lag* and target *Speed* ($F_{4,56} = 4.40, p < 0.01$), suggesting that latency affects performance in tracking differently depending on target speed. Figure 5.9 shows that the performance effect of latency is greater at lower target speeds than at higher speeds.

A significant three-way interaction between *Lag*, *Speed*, and *Compensation* was also shown by RM-ANOVA ($F_{4,56} = 3.38, p < 0.05$). This indicates that the effectiveness of my compensation technique varies depending on the amount of latency and target speed. Compensation resulted in a slight performance boost compared to baseline at lower target speeds when latency was 74 ms and 114 ms.

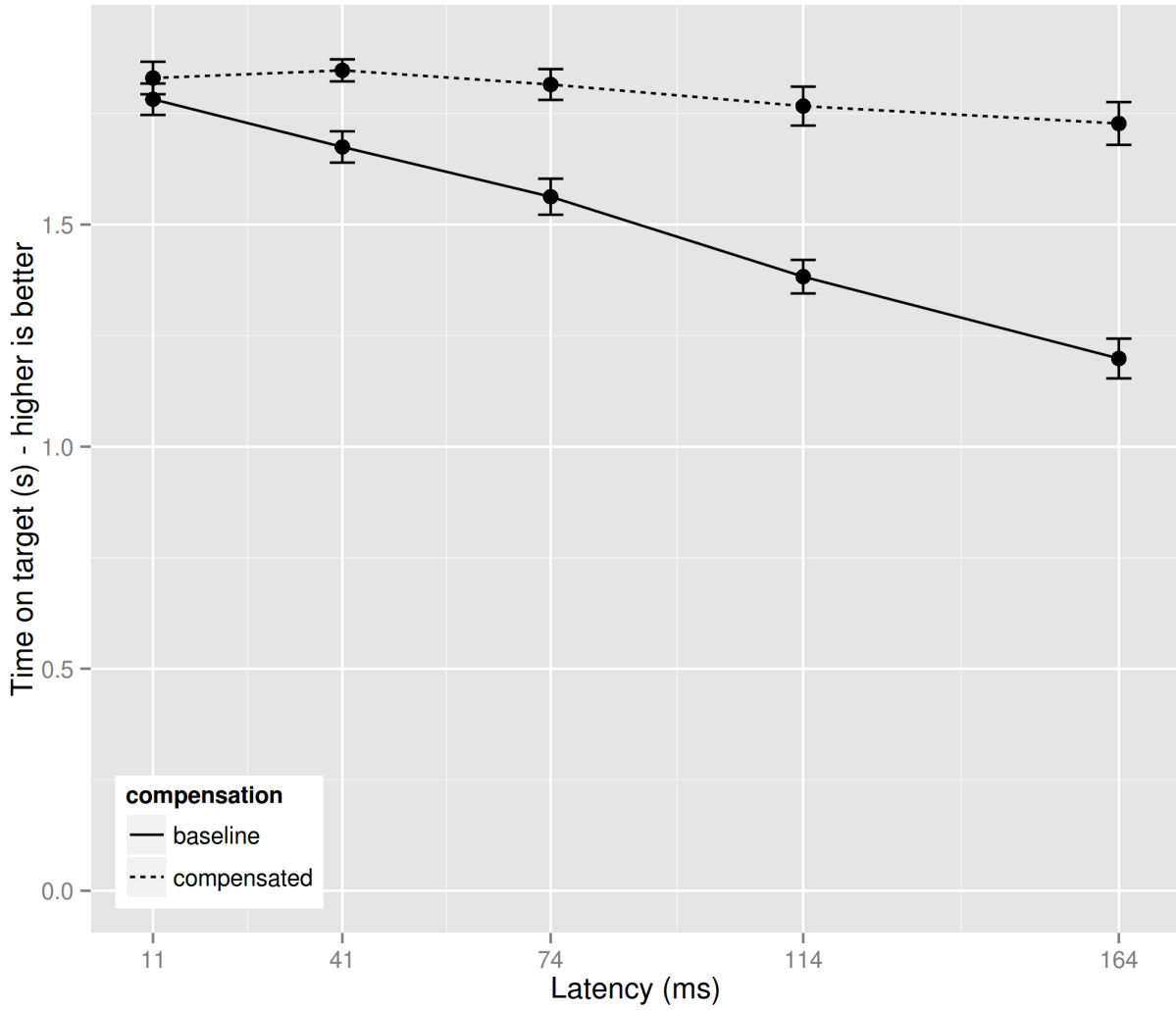


Figure 5.8: Effect of latency on tracking performance.

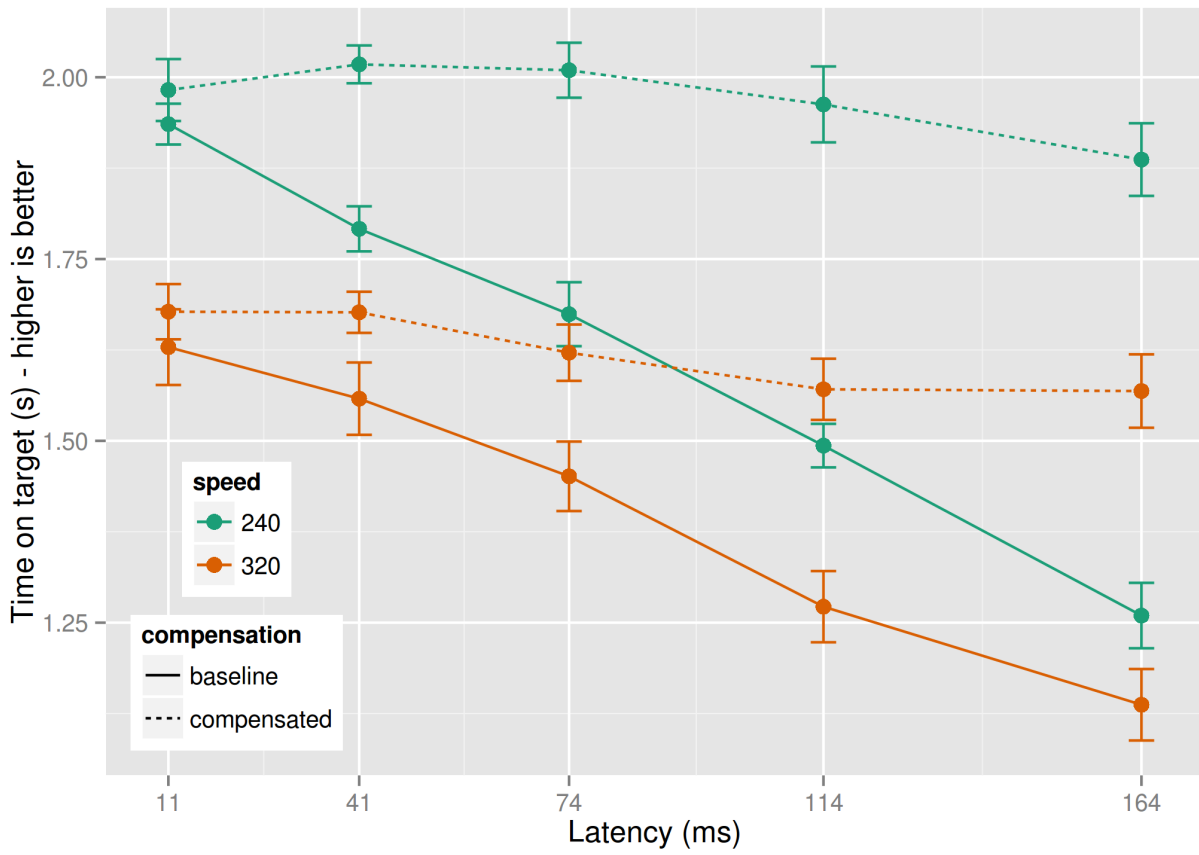


Figure 5.9: Effect of latency on tracking performance, based on target speed.

Table 5.6: Summary of RM-ANOVA main effects and interactions for the tracking task.

Effect	DFn	DFd	F	Significance
Lag	4	56	39.66	Yes
Speed	1	14	166.40	Yes
Block	2	28	8.98	Yes
Compensation	1	14	22.60	Yes
Lag : Speed	4	56	4.39	Yes
Lag : Block	8	112	2.20	Yes
Speed : Block	2	28	0.55	-
Lag : Compensation	4	56	46.16	Yes
Speed : Compensation	1	14	14.50	Yes
Block : Compensation	2	28	0.04	-
Lag : Speed : Block	8	112	0.82	-
Lag : Speed : Compensation	4	56	3.38	Yes
Lag : Block : Compensation	8	112	0.75	-
Speed : Block : Compensation	2	28	0.19	-
Lag : Speed : Block : Compensation	8	112	1.30	-

Effect of Block

RM-ANOVA showed a significant main effect of *Block* ($F_{2,28} = 8.98$, $p < 0.001$), indicating that there was a learning effect present. Tracking time increased by 4.1% in block 2 and 6.5% in block 3, as compared to block 1. As in the acquisition task, there was no significant interaction between *Block* and *Lag*, therefore there was no significant effect of practice on the impact of latency on performance.

Summary of Effects

Table 5.6 shows a summary of main effects and interactions analyzed with RM-ANOVA, including both significant and non-significant effects.

5.7.4 Survey Results

The survey results showed some interesting secondary findings.

Between-latency survey responses

Figure 5.10 shows a summary of the survey results given to participants between each sub-block, upon change of latency level. Answers were given as a seven-point Likert scale. The labels on the chart correspond to the survey questions as follows:

Performance "I performed well this round."

Lag Effect "Lag affected my performance."

Frustration "This round was frustrating."

Lagginess "The controls were laggy."

Surveys showed that participants could identify the presence of latency and its effect on performance at 114 ms of lag and higher (post-hoc u-test, $p < 0.05$), and could not distinguish between different levels of latency at the lower levels. With compensation enabled, they felt that they performed equally well at all levels of latency.

During the session in which latency compensation was enabled, participants were given an additional statement: "I noticed my aim being assisted." Based on the responses that were given, participants were not able to differentiate between the different levels of assistance given, even when compared to the baseline latency condition.

End of first session survey

At the end of a participant's first session, they were asked to fill out a brief five question survey. When asked before the experiment if they were aware of local latency (stated as "input lag" for familiarity) before the study, as a yes or no question, only 4 participants answered that they were not aware. 4 participants indicated that they experienced "quite a bit" of motion sickness during the experiment, with the choices being "not at all," "a little bit," and "quite a bit."

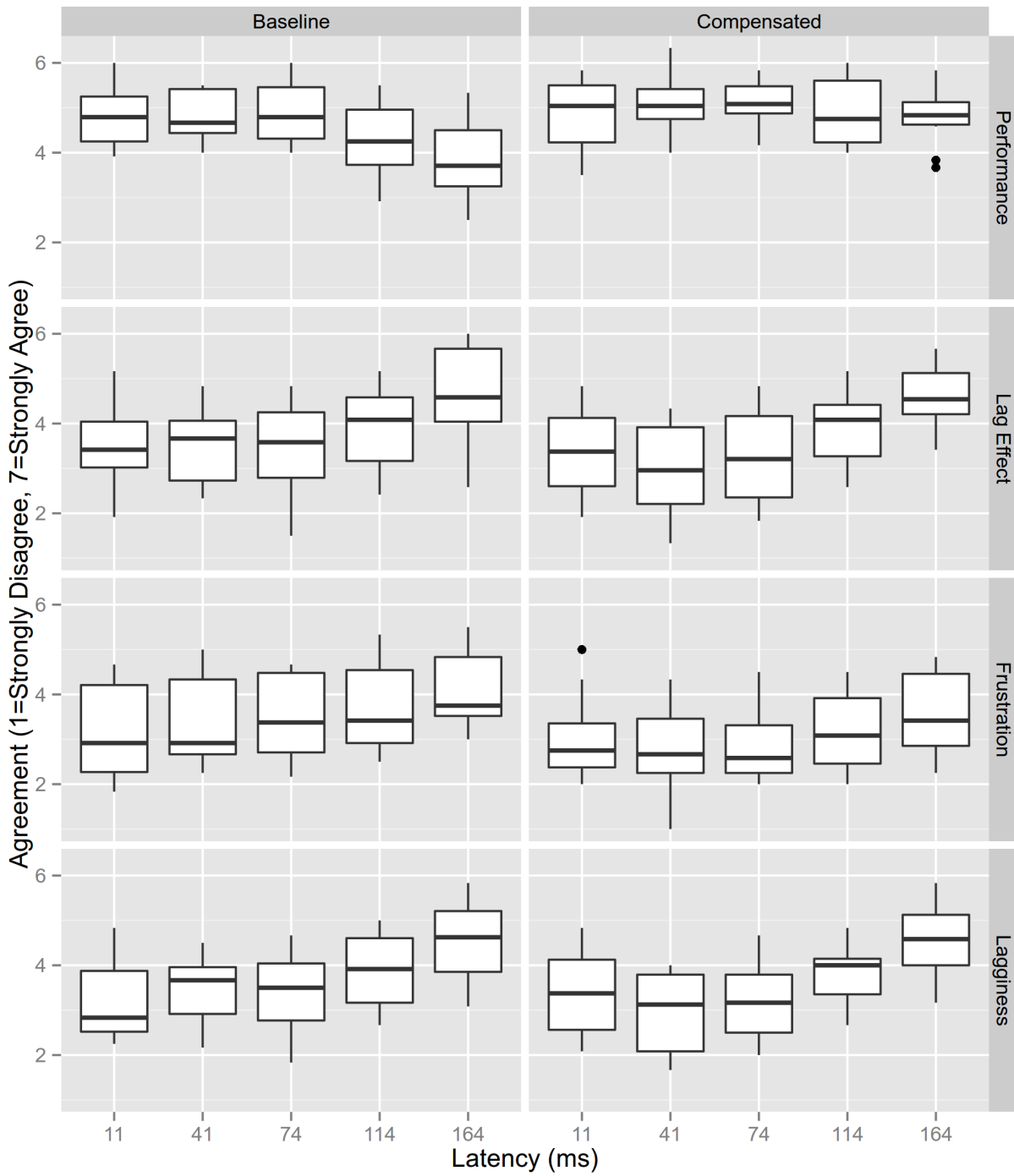


Figure 5.10: Summary of between-latency survey results.

The results for the remaining three questions are shown in Figure 5.11, presented as a histogram of responses. The given questions are indicated in the chart as follows, and answers were given as an eleven-point scale using semantic anchors at the ends of the scale, ranging from "not at all" to "very much:"

Performance "How much do you feel that input lag has affected your performance in games in the past (apart from this study)?"

Enjoyment "How much do you feel that input lag has affected your enjoyment of games in the past (apart from this study)?"

Purchasing "Has input lag affected your purchasing decisions in the past?"

End of second session survey

At the end of a participant's second session, they were asked to which extent they agreed with three statements, answered within a eleven-point Likert scale. A summary of answers is shown in Figure 5.12 as a histogram of responses.

Chart labels correspond with statements given as follows:

Performance "Aim assists improved my performance."

Preference "I preferred having my aim assisted compared to the previous day's non-assisted experience."

Desire "I would want to see aim assists being used to mitigate lag in commercial games"

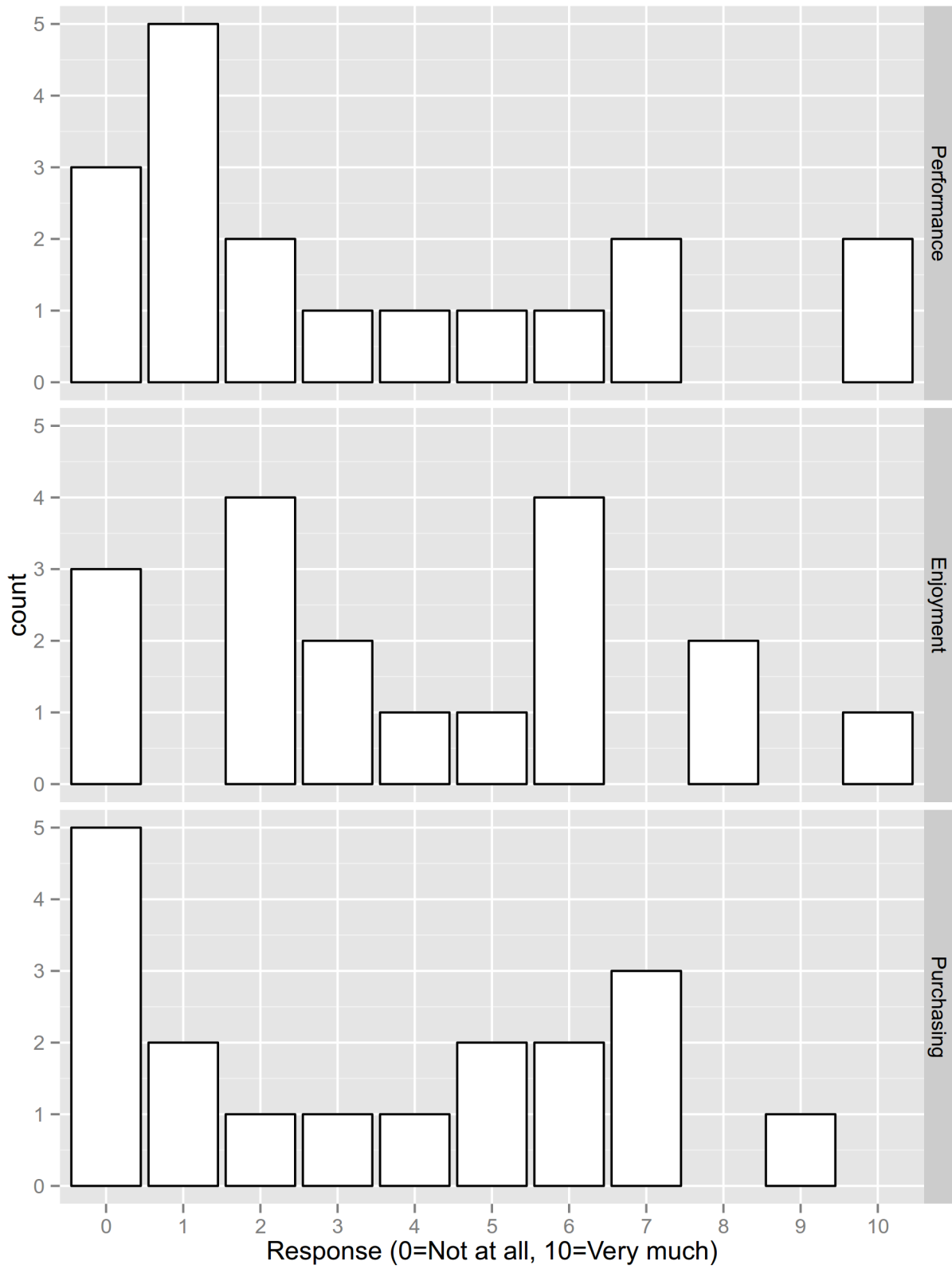


Figure 5.11: Summary of post-first session survey results.

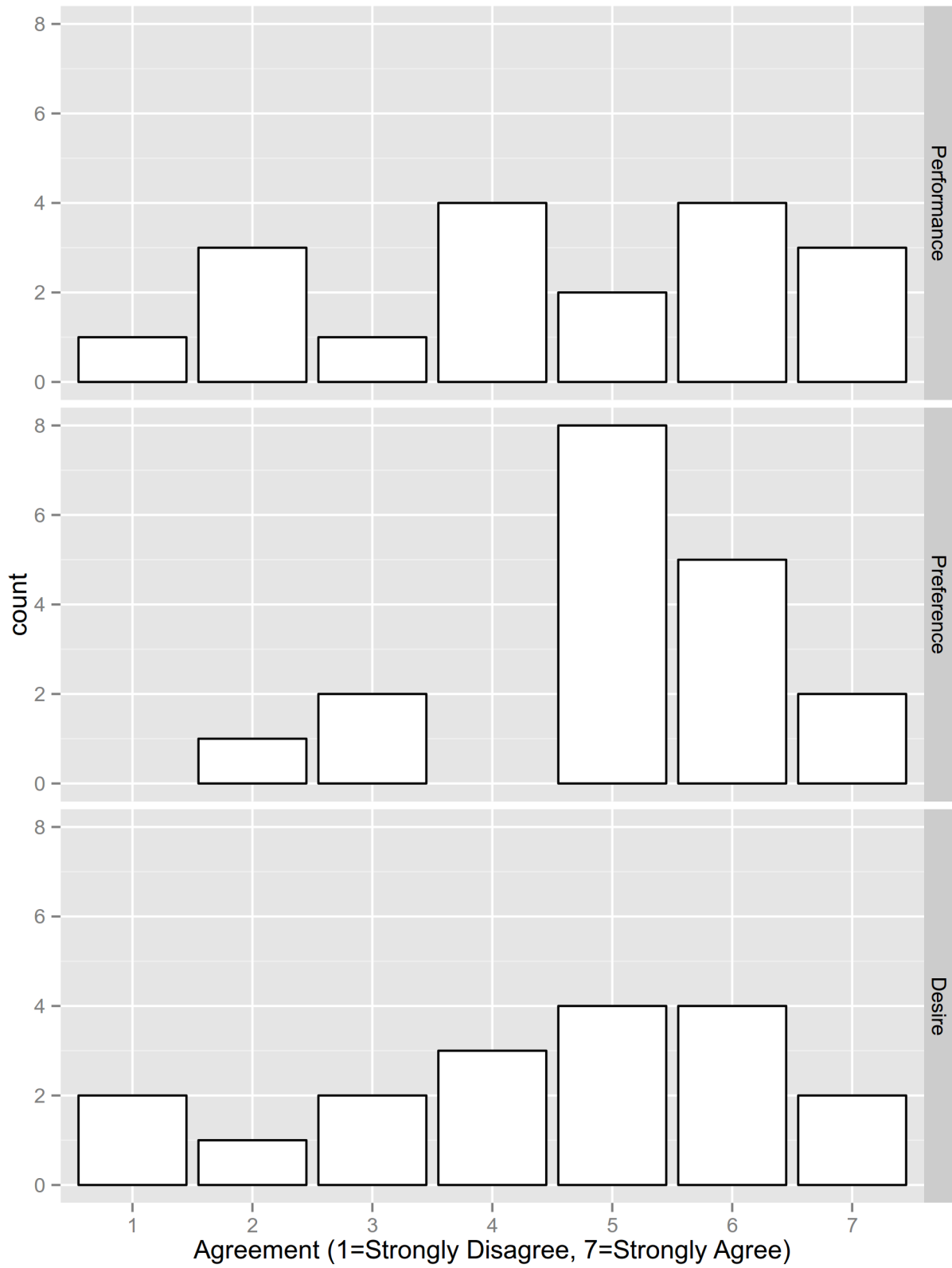


Figure 5.12: Summary of post-second session survey results.

CHAPTER 6

DISCUSSION

In this chapter, I discuss the results of the studies, with an emphasis on the findings from the previous chapter. I begin by summarizing and interpreting the results, followed by an explanation of them. I compare these results to previous work and discuss how they generalize to the real world. I also look at implications of the results for first person shooter game design, discuss the limitations of the work, and what could have been done differently for potentially better results.

6.1 Summary of Results

The primary goals of the research were to quantify the negative effects of local latency on aiming performance in first person shooters, as well as to see whether mitigation techniques could be effectively used to decrease the performance penalties of lag. I have clearly answered these questions in the studies through the research process.

Local latency has a substantial negative effect on aiming performance. For target acquisition, latency started having a significant effect on performance between 41 and 74 ms of lag, which is a common range encountered in PC systems. For targets with a lower index of difficulty, lag had significant effects even earlier, between 11 to 41 ms of lag. By 114 ms of lag (commonly seen on PC systems with games not optimized for low latency) acquisition time increased by 29%, and by 164 ms (common in console gaming on televisions), acquisition time was 52% above baseline performance.

For target tracking, latency levels above the baseline 11 ms all had a significant effect on tracking performance, with an approximately linear decrease in time on target as

latency increases. Time on target decreased by 22% at 114 ms of lag and 33% at 164 ms of lag.

Compensation reduced or eliminated the effects of lag on performance. For target acquisition, latency compensation in the form of sticky targets [95] and aim assistance reduced the negative performance effects of lag at all significant levels of latency. Assistance reduced the penalty of latency by up to 40%.

For target tracking, the compensation technique, in the form of sticky targets and aim dragging (similar to force fields [70]), was able to fully eliminate the performance effects of lag – that is, there was no significant difference in tracking time on target between any levels of latency compared to baseline.

Additionally, I had secondary research questions which had the goal of supporting the motivation or providing information toward the main questions – are there differences in aiming between 2D and 3D? How much local latency is seen in real world systems? What are users’ qualitative experiences with lag? I achieved these secondary goals as well:

Aiming in 3D is significantly different than in 2D. Target acquisition movement time was 8% longer in 3D conditions compared to equivalent 2D conditions. However, throughput [42] was similar between conditions, and error rate was higher in 2D, so the difference may be due to a different emphasis in the speed/accuracy trade-off. Regardless, these differences show that aiming is performed differently in 3D than in 2D.

Local latency is found in substantial levels in real-world systems. While purpose-built gaming PC systems can have very low latency (11 ms in the experimental setup in Chapter 5 and 23 ms in the samples in Chapter 4), I observed a wide and fairly evenly distributed range of latency in PC systems, spanning up to 158 ms of lag. Console systems are far more laggy, with sampled latency beginning at 130 ms and going up to as high as 243 ms in one system.

Many people are aware of lag, but experiences with it vary. In the study, 12 out of 16 people reported being aware of local latency before the study. Users could not detect lag until the 114 ms level or higher. Most participants thought that lag did not impede

their performance in past gameplay experience, although most thought that it did affect their enjoyment moderately. 10 people considered lag at least moderate amount when deciding their hardware purchasing decisions. Regarding aim assists, most participants thought that aim assists helped their performance, and, on average, participants would like to see aim assists being used in commercial games to mitigate lag. All but 3 participants preferred the assisted conditions in the study over the non-assisted ones.

Discussion for the findings on differences in aiming between 2D and 3D can be found in Chapter 3, while discussion on the local latency distribution in real-world systems is located in Chapter 4.

6.2 Explanation of Results

In this section, I attempt to explain why I found the results that were observed. Many of these explanations are hypotheses, and further examination would make for valuable future work.

6.2.1 Effects of Latency

The reasons why local latency causes performance to decrease are likely the same as those discussed for other types of latency. Both target tracking and acquisition (particularly for more difficult targets) rely heavily on visual feedback to perform the task [46]. As detailed in Section 2.1.2, controlled pointing movements are performed through means of one or more constituent submovements [28], beginning with a ballistic open-loop movement that is programmed to end at the target. Because stochastic neuromotor noise may cause the initial movement to miss, the initial movement may be followed by corrective movements that make use of visual feedback to correct for errors and arrive at the target [64]. Between each submovement, visual feedback is used to evaluate the error and plan a corrective action – in target tracking, this feedback takes on the order of 200 to 300 ms to adjust to a target direction change [46], and in acquisition, reports of 135 to 160 ms needed for visual feedback are common [17, 77]. Therefore, any mechanisms that delay the visual feedback or the amount of time necessary to process and react to it would also delay the total movement time.

Latency has a direct effect on these submovement-feedback cycles. Each time movement pauses between submovements while awaiting feedback, latency adds to the amount of time required to perform the feedback process because the previous submovement effectively ends later than normal by the length of the latency time. For example, if there were three submovements, pauses for feedback would take place twice, and so if latency magnitude was 100 ms, the total movement would require an additional 200 ms.

Also, in target acquisition in the present study, a trial begins as soon as a target appears, and therefore simple reaction time is part of the measured movement time. Latency has the effect of delaying the user's ability to see the target by the length of the latency duration. Therefore, even if this was the only factor of latency affecting movement time (i.e., ignoring the feedback interference), movement time would increase by the amount of latency present.

Based on these explanations, I can hypothesize a simple model for predicting acquisition movement time under the presence of latency:

$$t_l = t + l + (s - 1)l$$

where t_l is the new movement time, t is the original movement time, l is the amount of latency, and s is the number of submovements used to complete the task. However, by comparing the predictions of this model on my own results while assuming two submovements per trial (the best current model for movement time prediction assumes exactly two submovements [64]), the prediction overstates the effect of lag for lower latencies while predicting correctly at the high end of latency. One possible explanation for this is that the number of submovements used changes depending on the amount of latency – testing this would be an interesting question for future work. Also, there is evidence that visual feedback is used throughout fast reaching movement [77] rather than just between submovements. Finally, it is possible that users are accustomed to pointing under relatively low amounts of latency and so they perform better when some latency is present. Evidence for this can be seen when latency is subtracted from movement time to compensate for reaction time effects (red lines in Figure 5.4). In this case, movement time seems slightly faster at 41 ms compared to 11 ms.

Another reason that latency may have disruptive effects is that it causes a mismatch between visual feedback and kinesthetic feedback. In real-world aiming tasks, humans use

visual feedback to make adjustments for noise, but kinesthetic feedback (i.e., the body sensing the positions of one's own limbs) is used as well [36]. Local latency causes a mismatch between the two types of feedback due to the visual feedback arriving later than other kinds of feedback.

The difference in sensitivity between targeting and tracking may arise from the added requirements of the tracking task – during tracking, changes in the direction of target movement were observed to cause significant disruption, and direction changes occurred numerous times during each trial. Direction changes cause a sort of double penalty in tracking – not only does one take longer to react to a direction change with lag, but the target has moved farther in the opposite direction relative to the crosshair by the time corrective action is taken (particularly at higher target speeds [46]). This requires a greater magnitude of corrective action, and this magnitude approaches its peak at approximately two direction changes per second, at which point users perform only one corrective action per direction change [36]. Additionally, when latency is very high and direction changes happen frequently, there becomes a point when the amount of time taken to react to a direction change and execute the lagged movement is as high or greater than the time between direction changes. In this case, tracking is a sequence of chaotic guesses because by the time the user has attempted to react to a direction change, another direction change is already underway.

6.2.2 Compensation for Latency

As with the effects of latency, the reasons for the success of the compensation mechanisms also align with previous work on aim assistance. That is, the compensation techniques worked well because they provided a kind of bridge across the delay-caused gap between the initiation of an action and the display of that action. For targeting, compensation enlarged the effective width of targets by the amount needed to make up for the overshooting that occurred due to latency; for tracking, aim dragging kept the crosshair on the target for the amount of time in which the target's direction change was invisible due to delay.

The difference in compensation effectiveness between acquisition and tracking (compensation did not work as well in acquisition) arises from the fact that I only compensated for overshooting errors, rather than lateral errors. I initially thought that latency would primar-

ily cause overshooting, but I observed that lateral aiming errors were also common, and the presence of lag hampered the player’s ability to adjust their aim through controlled feedback. Future work will examine techniques that take these transverse errors into account (e.g., bullet magnetism [89]).

The surprising finding in the study of compensation techniques was the degree of effectiveness of the technique, even at the highest levels of local latency that are likely to be experienced in the real world. This means that the problem of local latency is one that could potentially be solved if compensation techniques can be successfully deployed.

6.3 Comparison to Similar Studies

With regards to the effects of latency on aiming performance, the surprising finding in my study, is that even latencies as low as between 11 and 41 ms can cause significant performance problems for both target acquisition and tracking. Few studies have suggested that performance in aiming tasks can be affected by latencies this small – although one recent experiment on direct-touch input with ultra-low latency touch screen hardware also found that latency above 10 ms had a negative effect on touch performance [44].

A study on the effects of network latency and packet loss in Unreal Tournament 2003 (a previously popular, fast paced first person shooter) examined the effect on a variety of metrics, including hit fraction in a precision shooting test and effects on kill and death scores [13]. In the hit fraction test, accuracy was similar up until 100 ms of latency, at which point performance dropped sharply. Similar results were found in a restricted deathmatch scenario where a human was made to fight against an AI opponent in a controlled environment with restricted weapons. I found that local latency has effects well before this 100 ms threshold, and this is likely because of the differences between the effects of network lag and local latency (see Section 2.3.1), such as network lag not affecting local aim feedback, and the presence of network latency mitigation techniques in the game.

A 1993 study by MacKenzie examines the effects of local latency on pointing performance in a Fitts’ Law target acquisition task [61], finding effects at levels potentially as low as 25 ms, although statistical significance was not examined at specific levels of latency.

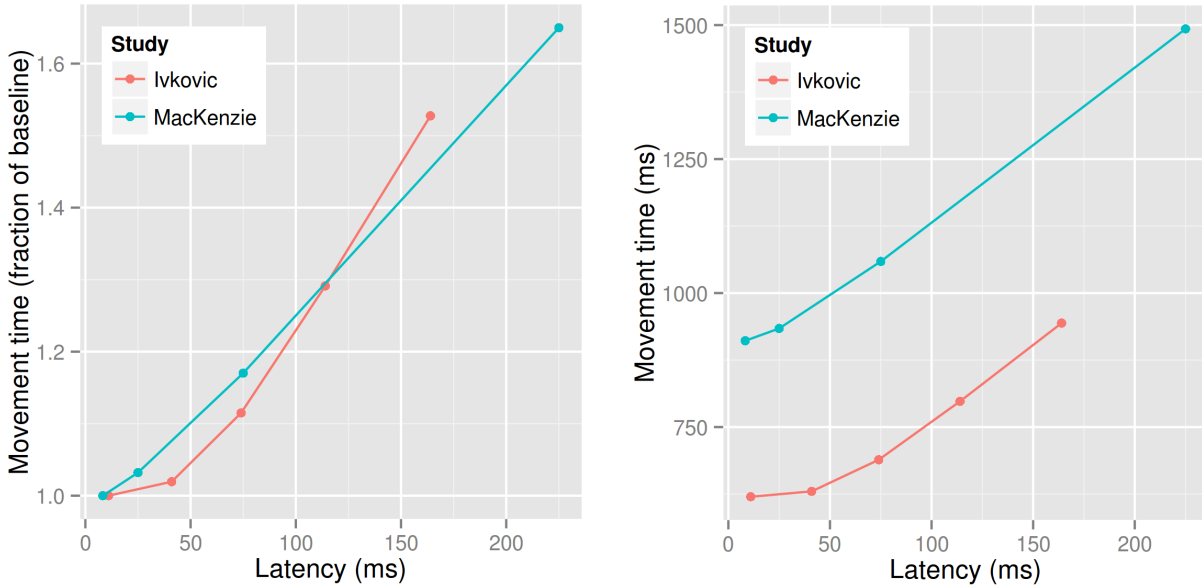


Figure 6.1: Comparison of results of my study versus MacKenzie. Left side shows performance relative to baseline, while right side shows absolute movement time.

Figure 6.1 shows a comparison of MacKenzie’s results versus mine. In the left pane, movement time is compared as a comparison to time at baseline latency (8.3 ms for MacKenzie, 11 ms for us). MacKenzie’s results show that latency has an approximately linear effect on performance, while in comparison, my results show a smaller effect at low levels of latency but a greater effect at higher levels. The penalty growth is lower up until my 41 ms, at which point growth becomes equal and then surpasses that observed by MacKenzie. However, it is important to note that this comparison ignores index of difficulty values since raw data based on ID was not made available by MacKenzie, and ID values in his study range up to approximately 6, while mine only reach a value of 3. This can be observed in the right pane of the figure, where MacKenzie’s movement time is much higher than mine. This makes a direct comparison somewhat limited in utility, particularly since both MacKenzie and I found that ID interacts with lag. However, it is interesting to observe that absolute movement time rises approximately parallel between the two studies after the initial few levels of latency, meaning that lag might be additive with movement time at higher levels.

In contrast, results of a 2D-based study by Pavlovych [72] that is seemingly equivalent to MacKenzie’s show quite different results: latency only starts showing significant effects

on performance at 108 ms of lag, and not at 83 ms or below. This discrepancy is noted and discussed by Pavlovych who concludes that the only viable hypothesis regarding the reason is that MacKenzie's system had an additional hidden 60 ms of lag compared to stated levels, which, when added, would make the results similar. If this hypothesis is true, then my study is the only target acquisition study that I'm aware of that shows sensitivity to latency at low levels. If the hypothesis is false, then my study is valuable in clarifying the lack of consensus in literature regarding the threshold of effects of latency.

Network latency in cloud gaming has very similar effects to local latency since both types of lag delay all feedback including local feedback to aiming movements. Therefore, network latency when playing an FPS through cloud gaming should be comparable to the effects of local latency in my study. While there do not seem to be cloud gaming studies that examine effects on player performance in situations similar to mine, studies have been performed regarding the latency levels needed for them to be perceivable [52, 75]. These studies show users noticing effects at levels of approximately 50 ms, which is lower than my findings that only show users noticing lag at 114 ms and higher. This discrepancy could be because higher sensitivity methodology is used (e.g., an ECG device [52]) rather than relying on users self-reporting), or because the addition of the remaining elements of FPS gameplay that I removed could make latency easier to notice.

FPS environments have unique properties that make latency particularly disruptive [22, 52], as can be observed by the faster growth of relative performance decrease in my study compared to MacKenzie's 2D study [61] and the higher sensitivity to latency at low levels compared to Pavlovych [72]. First, shooter games involve fast-paced game events with tight deadline requirements, meaning that quick reflexes and precise aiming are paramount to performance. Second, the amount of precision required in first person shooters is greater than in most other genres. Third, the first-person perspective means that a player's movements cause the entire view of the 3D scene to shift behind a fixed cursor in the center of the screen. This causes much stronger visual changes (e.g., optical flow and motion parallax) as compared to 2D environments where a small cursor is moved across a fixed background. Also, the first-person perspective requires more frequent and precise aim adjustments and therefore relies more heavily on feedback.

6.4 Generalization to Real FPS Games

The research in this thesis was (necessarily) a controlled experiment rather than a real-world test in a real FPS game, there are several reasons why my results about the effects of latency will hold in realistic scenarios:

- I tested levels of latency that are common in gaming systems, and covered a wide range of systems with different input and display devices in various games.
- I tested realistic task elements (targeting and tracking) that are ubiquitous in 3D shooter games. Although there are many differences between my study system and an actual FPS game (e.g., moving while shooting, other visual information), the core mechanics of moving the mouse to acquire or track a target do not change.
- The ways that targeting and tracking work in most FPS games is similar to our experimental tasks. FPS players typically aim for fixed targets in a single aim-and-shoot action, and moving targets typically move along a horizontal plane (e.g., the ground or a ledge) with most evasive action occurring as horizontal direction changes. Movement that lies along a projection toward or away from the player's aim does not significantly affect aiming requirements, since the only effect is a slight change in target size.
- I used an experimental setup that closely mimics a real gaming PC (fast mouse, graphics card, and display), and recruited participants with FPS gaming experience.

There are some differences that should be studied further in a real FPS environment. For example, in the study system the player could not move the character. While player movement is often used as part of tracking in FPS games, this restriction allowed us to focus specifically on mouse input for tracking tasks (although many FPS games encourage the player to stay stationary while firing in the form of reduced accuracy when firing while moving).

However, I believe that differences between my testing environment and a real FPS environment will most likely increase the effects of lag. Player movement adds additional complexity when aiming and provides further opportunity for lag to interfere with the feedback

processes. Likewise, the combination of movement and pressure to fire accurately encourages players to use the closed-loop feedback phase of aiming which is highly affected by lag due to the need for multiple adjustments with feedback delay between each one. Unlike real games, my experiment involved repeated simple aiming motions where participants could easily focus on the open-loop ballistic phase of motion.

Given enough time, players tend to get used to the latency of a particular system and notice it less. Players often claim that this adaptation negates the performance effect of lag, and although that may be true to a small extent, I believe the adaptation effect is limited. Lag reduces the amount of time available to respond to game events. Planned acquisition motions such as open-loop ballistic motion with correctly predicted target locations may become latency-adapted to some extent, but errors are common and would negate adaptation. Likewise, tracking predictable targets may suffer less from lag, but any unexpected path deviations or errors in tracking would suffer from the closed-loop feedback-delay penalty imposed by latency. When players become highly practiced at aiming tasks, they do not reduce the need for visual feedback, but rather they become even more reliant on it [74], and therefore become potentially more affected by latency. my experiment did attempt to give players a chance to adapt to lag by making smaller, sequential changes to it in small steps rather than randomizing latency levels, as well as giving a large amount of time to training at the start of the experiment and removing early trials in each block from results.

My study setup also supports generalization of the effects of compensation to real FPS games. The compensation technique was designed to be general enough to apply to many common aiming scenarios. Two issues were not studied in The experiment: first, the effects of large required movements to acquire a target (e.g., real gameplay might require the player to perform a rapid 180 degree turn to aim at an enemy); second, the potential effects of distractor targets. Distractors that can appear near the intended target can make aiming assistance techniques less effective [89], even though the technique is designed to only affect a small angle around the targeting cursor. In real FPS scenarios, distractors may be less of an issue because players typically shoot at the nearest target. In addition, modifications to the compensation algorithm – such as determining the size of the effect based on cursor speed – may help to avoid errors.

There are some important challenges in deploying a compensation scheme in a real-world FPS. One main issue is the need to know the amount of local latency in order to set the amount of compensation. If the estimated latency is incorrect, compensation will either work ineffectively or provide an unfair advantage to the player. To accurately determine local latency, measurement tools should be used. These tools are already in use for latency-sensitive devices such as the Rock Band¹ game controller and the Oculus Rift² virtual reality headset. However, to ensure fairness it would be better if latency estimates could be made directly from the hardware components that make up a game system. I envision that the required information could be archived in a database of component latency specifications, ideally created by manufacturers and from crowd-sourced gaming communities (which already has similar sites.³ The game could then require that players establish their latency before allowing compensation to be enabled.

The study that compares aiming in 2D to 3D also generalizes well because it uses established standards in the human computer interaction field [84], such as using the ISO 9241-9 [42] task for evaluating input devices. It also makes use of standard comparison methods such as movement time, error rate, and throughput. Additionally, every effort was made to create a 3D version of the task that was as similar as possible to the 2D version for direct comparison, including matching target index of difficulty levels between the two conditions in such a way that the amount of mouse movement required in both is equal.

6.5 Implications of Results

The findings on effects of latency have a number of implications for game designers, hardware manufacturers, and consumers. I have shown that even a small amount of latency has negative performance effects, yet game designers do not seem to treat minimizing latency as a priority. There are numerous ways to make games suffer from innate latency, such as filtering input, interpolating the game state within the rendering subsystem, or post-processing effects such as motion blur. Although I have not investigated exactly which game engine techniques cause

¹<http://www.harmonixmusic.com/games/rock-band/>

²<http://www.oculus.com>

³<http://www.displaylag.com>

lag, it is clear from the results in Chapter 4 that games do often induce lag due to their design. Additionally, developers can examine whether advanced techniques such as scheduling input and logic processing to happen as late as possible in the time between frames to reduce latency is suitable for their games.

If it is not possible to minimize latency, game designers can potentially reduce the reliance of game mechanics on low latency by understanding studies such as [23, 24]. FPS games are inherently one of the most prone to latency for reasons that I explained previously. Reducing the deadline requirements of actions or the amount of precision needed to have useful effects would help toward making the game less latency-sensitive, but such approaches should be weighed carefully due to potentially negative gameplay implications.

These results will hopefully also motivate hardware manufacturers to make hardware with low latency more readily available. Expensive devices marketed toward gamers exist, but these are out of reach of many gamers or have significant downsides (e.g., gaming monitors generally have twisted-nematic panels which offer inferior image quality to some other types of panels). In reality, there are devices that are both inexpensive and of reasonable quality that do not market themselves as high-end gaming devices, but they achieve low latency by not using design decisions in areas such as on-screen displays and image processing methods that add latency. A broad focus on decreasing latency in devices can likely decrease lag across devices without adding substantial cost.

Another possible consequence of this research is that consumers may be motivated to make decisions regarding hardware and games that take latency into consideration. Currently, many consumers are either not aware of local latency at all, or they underestimate the magnitude of its effects (as evidenced by the survey results), either by not realizing that there are performance effects at latencies that they do not perceive, or by not being aware of how high the performance penalties are at high levels of lag such as those present in console games on TV displays. It is interesting though that the survey results show that most participants did acknowledge experiencing lower enjoyment of games as a result of latency, yet they many did not previously think it affected their performance.

Finally, game developers should consider including latency mitigation by means of aim assists in games where it is appropriate to do so. I showed that assists are effective at

reducing the negative effects of lag on performance. Also, the survey results show that almost all participants preferred the conditions that made use of assists, and about half of them were positive toward including such assists in real games.

The understanding of Fitts' Law in one and two dimensions is taken as the ground truth in human computer interaction studies, so understanding how 3D situations differ from this baseline is a valuable result. It is important to know whether actions in 3D environments can be modeled using the known 2D relationships and equations. If this were the case, it would allow access to an extensive body of research for FPS game designers and researchers to make use of. However, by directly comparing 2D versus 3D FPS environments, I have shown that there are important differences in aiming performance between 2D and 3D contexts. Therefore, further work is needed to translate the body of previous work on 2D pointing to 3D FPS contexts. Once this translation is performed the detailed analysis of aiming can support empirically based design decisions. For example, an FPS game designer may choose to use specific target sizes relative to the overall map size in order to elicit a particular level of player performance.

Augmentations to FPS game mechanics can also benefit from the detailed analysis of aiming in the FPS context. For example, such adjustments are needed for player balancing. Recent work has shown that 2D aim assists do not translate directly to 3D [89]; therefore by incorporating assists based on aiming data relevant to the 3D FPS environment, better player balancing techniques may be achieved.

6.6 Limitations

Although this work is valuable toward understanding the effects of latency and aiming in 3D, there are some limitations and things I would change in retrospect. One significant change I would make is to perform the study presented in Chapter 3 first. Although the 2D/3D study is presented first, in reality, the first major body of work I performed was the latency study in Chapter 5, because the latency study is the one that examined the primary questions I set out to answer from the beginning.

As a consequence, I learned some lessons too late to apply them to the latency study. In

particular, the range of index of difficulty values I used in the latency study was fairly small, with the maximum value being only 3.02, as opposed to an ID of approximately 6 in similar 2D studies on latency. While I were restricted in the distance values used for targets so that they can stay within the player's field of view and room boundaries, I could have made the minimum target size smaller to create more difficult conditions. Another change I would have made is to use spherical targets in target acquisition in the latency study. The targets I did use (ogre heads) appear approximately spherical, but their bounding area was modeled as a box. This results in effective target widths that change based on distance angle, which I did not account for. This makes the plots involving index of difficulty slightly inaccurate, but none of my significant results are affected by the oversight.

Another limitation of my work is that I did not spend enough time on creating and tuning the aiming assists used for latency mitigation. Despite this, they worked remarkably well in target tracking, and although they were useful in acquisition, they could have been more effective. For optimal performance, I should have piloted other approaches to the aim assists and tried making use of other techniques. Also, tuning the parameters for the current aim assist formulas would have likely resulted in significantly better improvements in acquisition.

Although the field samples of latency that I performed in Chapter 4 were not meant to be thorough and exhaustive, it would have been beneficial to measure more setups than I did. Time and access constraints limited the number of sample points I had, but these were still sufficient to establish a useful range of latency values and make observations regarding patterns such as what factors have a significant effect on total system latency.

While I were very successful in making equivalent conditions between 2D and 3D in target acquisition in Chapter 3, the target tracking comparison would have been more useful if I either used cylindrical targets or made the target turn to face the player while it is moving. Either option would avoid the problem of the target effectively changing width when looked at from an angle due to its box-shaped hit area.

Finally, it would have been potentially valuable to have more participants in the studies, particularly for the study in Chapter 3, which only had twelve participants. When I analyzed the results with only eight participants, the magnitude of the movement time differences between 2D and 3D was 12% as opposed to the 8% I found in the end. This indicates some

instability in the results, and having more participants would help create confidence in the magnitude of the effects.

CHAPTER 7

CONCLUSION

The primary problem I intended to solve with this research was the lack of knowledge of precise, quantitative information regarding the effects of local latency on aiming performance in first person shooters, as well as a lack of mitigation techniques that can be used to compensate for the latency. A secondary problem I identified was that there was limited knowledge of how pointing in 2D settings compares to aiming in 3D FPS environments, which leaves doubt regarding whether knowledge of latency in 2D can be directly applied to first person shooters. I solved these problems through the studies that I performed and documented the approach and results in this thesis.

Although target acquisition and tracking have been previously characterized in the context of 2D graphical environments, little research has evaluated how those results translate to a 3D FPS-like graphical environment. I performed such an evaluation within 2D and 3D environments that were visually similar and had targets with matched indexes of difficulty. My results indicate differences between 2D and 3D that are in some cases substantial. The main lesson from this comparison for FPS games, therefore, is that previous targeting results from traditional 2D pointing studies may not be directly transferable to 3D FPS games. Past design work that is based on 2D models should be re-evaluated given the additional difficulty presented by these more complex 3D environments.

My work shows that local latency is a real and substantial problem – but that games can mitigate the problem with appropriate compensation methods. My studies provide the first empirical evidence quantifying the negative effects of local system latency on aiming in 3D game environments. My results show that local latency levels as low as 41 ms (which is lower than the lag found in most gaming systems) causing significant degradation in aiming performance, and that performance degradation increased to substantial levels at higher

latency – movement time increased by 52% in target acquisition, and target tracking time on target fell by 32% at 164 ms of latency, which is not uncommon in gaming systems. The results also show that significant amounts of latency prevail in real-world systems and that aim assistance can effectively mitigate local latency. My compensation scheme reduced the problems caused by lag in the case of targeting and removed the problem altogether in the case of tracking.

Local latency is pervasive and substantial in magnitude in gaming systems, and because aiming in 3D first person shooters has important differences to pointing in 2D settings that may make studies of latency in 2D to be inapplicable to FPS, I quantified the effects of latency on aiming in FPS games and also demonstrated effective techniques to compensate for the latency.

7.1 Questions for Further Study

My research also raised other questions and issues that would be valuable to investigate in future work. To begin with, the limitations I described in Section 6.6 would be useful to overcome in follow-up studies. I also present some other interesting possibilities for research in this section.

7.1.1 Effects of Lag on Score in Real Games

My studies provide valuable evidence on the effects of latency in specific, restricted aiming tasks. However, real games make use of many elements that mine do not, such as movement, targets firing back at the player, level knowledge, and so on. Likely the most important future work to perform would be to examine how local latency affects overall player scores in real-world competitive first person shooters.

7.1.2 Conflicting Results on Effects of Latency

Although existing studies of latency in 2D pointing tasks use different methodologies, conditions, and latency levels than my experiment, it seems that latency may have effects of

similar magnitude in 2D pointing as in 3D first person shooters according to some studies (e.g., [61]). However, previous studies conflict with each other [61, 72], and therefore there is no clear consensus on how latency affects simple target acquisition behavior. Further studies are needed to resolve these conflicting results.

Additionally, studies on 2D target tracking under latency show results very different than the results I found on target tracking in FPS. In one study [71], tracking errors were only significantly greater at 170 ms of latency than at baseline latency, as opposed to my results which show degradation as low as 41 ms. It would be valuable to investigate the reasons for this wide discrepancy in threshold of latency effects.

7.1.3 Changes to Aim Mechanics Due to Latency

In Section 6.2.1, I hypothesized a simple model to predict acquisition time under latency. Because the model (which relied on a measure of number of submovements performed in aim) did not properly predict actual performance, I proposed that latency might change the number of submovements performed during aim, or that it might shift the weight placed on the initial ballistic movement compared to the feedback-corrected final movements. It would be useful to examine these ideas in detail.

Additionally, while I examined the effects of latency on aim performance and differences in aim performance between 2D and 3D settings, I did not delve deep into the mechanical reasons regarding why these effects appear. It would be valuable to examine in detail how aiming might change under latency, or which differences in the environment or aiming mechanics between 2D and 3D cause the differences in performance.

These studies should also consider using a 3D virtual reality headset as part of the examination since the devices offer true 3D cues that are not available while using a standard screen-projected world. Also, VR is likely even more sensitive to lag than standard screens are [2], so studying the effects of lag in this context, where it is difficult to achieve sufficiently low latency, could be useful.

7.1.4 Index of Difficulty under Latency

Are target size and target distance independent factors for index of difficulty in the presence of latency? So and Chung [83] suggest that target size and distance should not be considered a single effect of ID for tasks involving latency. In the study regarding the effects of latency on aim, I included two target conditions with similar IDs, but with different combinations of distance and size. With lag present, the closer and smaller targets had consistently worse performance than the targets with greater distance and size (but similar ID). The performance discrepancy with these similar IDs also increased with lag (Figure 5.6). This would suggest that local latency can cause greater problems for smaller targets. However, this preliminary finding will need to be verified by a more targeted study as future work.

REFERENCES

- [1] Richard A Abrams and Jay Pratt. “Rapid aimed limb movements: Differential effects of practice on component submovements”. In: *Journal of Motor Behavior* 25.4 (1993), pp. 288–298.
- [2] Michael Abrash. *Latency – the sine qua non of AR and VR*. 2012. URL: <http://blogs.valvesoftware.com/abrash/latency-the-sine-qua-non-of-ar-and-vr/>.
- [3] Johnny Accot and Shumin Zhai. “Beyond Fitts’ law: models for trajectory-based HCI tasks”. In: *Proceedings of the ACM SIGCHI Conference*. ACM. 1997, pp. 295–302.
- [4] Jacob Agar, Jean-Pierre Corriveau, and Wei Shi. “Play patterns for path prediction in multiplayer online games”. In: *Communications and Networking in China (CHINA-COM) 2012*. IEEE. 2012, pp. 12–17.
- [5] Sudhir Aggarwal et al. “Accuracy in dead-reckoning based distributed multi-player games”. In: *Proceedings of 3rd ACM SIGCOMM workshop*. ACM. 2004, pp. 161–165.
- [6] David Ahlström, Martin Hitz, and Gerhard Leitner. “An evaluation of sticky and force enhanced targets in multi target situations”. In: *Proceedings of the 4th Nordic Conference on Human-computer Interaction: Changing Roles*. ACM. 2006, pp. 58–67.
- [7] Rahul Amin et al. “Assessing the impact of latency and jitter on the perceived quality of Call of Duty Modern Warfare 2”. In: *Human-Computer Interaction. Users and Contexts of Use*. Springer, 2013, pp. 97–106.
- [8] Michael Antonov. *Asynchronous Timewarp Examined*. Oculus VR. 2015. URL: <https://developer.oculus.com/blog/asynchronous-timewarp-examined/>.
- [9] Entertainment Software Association. *Essential Facts About the Computer and Video Game Industry*. 2013. URL: http://www.theesa.com/facts/pdfs/ESA_EF_2013.pdf.
- [10] Ravin Balakrishnan. ““Beating” Fitts’ law: virtual enhancements for pointing facilitation”. In: *International Journal of Human-Computer Studies* 61.6 (2004), pp. 857–874.
- [11] Scott Bateman et al. “Target assistance for subtly balancing competitive play”. In: *Proceedings of the ACM SIGCHI Conference*. ACM. 2011, pp. 2355–2364.
- [12] WDA Beggs and CI Howarth. “The accuracy of aiming at a target: Some further evidence for a theory of intermittent control”. In: *Acta Psychologica* 36.3 (1972), pp. 171–177.
- [13] Tom Beigbeder et al. “The effects of loss and latency on user performance in unreal tournament 2003”. In: *Proceedings of 3rd ACM SIGCOMM workshop*. ACM. 2004, pp. 144–151.

- [14] Yahn W Bernier. “Latency compensating methods in client/server in-game protocol design and optimization”. In: *Game Developers Conference*. Vol. 98033. 425. 2001.
- [15] Renaud Blanch, Yves Guiard, and Michel Beaudouin-Lafon. “Semantic pointing: improving target acquisition with control-display ratio adaptation”. In: *Proceedings of the SIGCHI Conference*. ACM. 2004, pp. 519–526.
- [16] Ugo Braga Sangiorgi et al. “Assessing lag perception in electronic sketching”. In: *Proceedings of the 7th Nordic Conference on Human-Computer Interaction: Making Sense Through Design*. ACM. 2012, pp. 153–161.
- [17] Les G Carlton. “Processing visual feedback information for movement control”. In: *Journal of Experimental Psychology: Human Perception and Performance* 7.5 (1981), p. 1019.
- [18] Alison N Cernich et al. “Sources of error in computerized neuropsychological assessment”. In: *Archives of Clinical Neuropsychology* 22 (2007), pp. 39–48.
- [19] Olivier Chapuis, Jean-Baptiste Labrune, and Emmanuel Pietriga. “DynaSpot: speed-dependent area cursor”. In: *Proceedings of the ACM SIGCHI Conference*. ACM. 2009, pp. 1391–1400.
- [20] Kuan-Ta Chen, Polly Huang, and Chin-Laung Lei. “How sensitive are online gamers to network quality?” In: *Communications of the ACM* 49.11 (2006), pp. 34–38.
- [21] Romeo Chua and Digby Elliott. “Visual regulation of manual aiming”. In: *Human Movement Science* 12.4 (1993), pp. 365–401.
- [22] Mark Claypool and Kajal Claypool. “Latency and player actions in online games”. In: *Communications of the ACM* 49.11 (2006), pp. 40–45.
- [23] Mark Claypool and Kajal Claypool. “Latency can kill: precision and deadline in online games”. In: *Proceedings of the first annual ACM SIGMM Conference*. ACM. 2010, pp. 215–222.
- [24] Mark Claypool, Tianhe Wang, and McIntyre Watts. “A taxonomy for player actions with latency in network games”. In: *Proceedings of the 25th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*. ACM. 2015, pp. 67–72.
- [25] ERFW Crossman and PJ Goodeve. “Feedback control of hand-movement and Fitts’ law”. In: *The Quarterly Journal of Experimental Psychology* 35.2 (1983), pp. 251–278.
- [26] Christophe Diot and Laurent Gautier. “A distributed architecture for multiplayer interactive applications on the Internet”. In: *Network* 13.4 (1999), pp. 6–15.
- [27] DisplayLag. *Input Lag Database*. 2015. URL: <http://www.displaylag.com/display-database>.
- [28] Digby Elliott, Werner F Helsen, and Romeo Chua. “A century later: Woodworth’s (1899) two-component model of goal-directed aiming”. In: *Psychological bulletin* 127.3 (2001), p. 342.
- [29] Digby Elliott et al. “Discrete vs. continuous visual control of manual aiming”. In: *Human Movement Science* 10.4 (1991), pp. 393–418.

- [30] Digby Elliott et al. “Goal-directed aiming: Correcting a force-specification error with the right and left hands”. In: *Journal of Motor Behavior* 31.4 (1999), pp. 309–324.
- [31] Digby Elliott et al. “Optimizing the use of vision in manual aiming: The role of practice”. In: *The Quarterly Journal of Experimental Psychology* 48.1 (1995), pp. 72–83.
- [32] Leah Findlater et al. “Enhanced area cursors: reducing fine pointing demands for people with motor impairments”. In: *Proceedings of the 23rd annual ACM symposium on User interface software and technology*. ACM. 2010, pp. 153–162.
- [33] Paul M Fitts. “The information capacity of the human motor system in controlling the amplitude of movement”. In: *Journal of Experimental Psychology* 47.6 (1954), p. 381.
- [34] *Full-screen DirectX desktop apps using the Flip Presentation Model*. Intel Developer Blogs. 2013. URL: <https://software.intel.com/en-us/blogs/2013/06/03/full-screen-direct3d-games-using-borderless-windowed-mode>.
- [35] Claude Ghez and John Krakauer. “The organization of movement”. In: *Principles of Neural Science* 4 (2000), pp. 653–73.
- [36] ERFW Grossman. “The information-capacity of the human motor-system in pursuit tracking”. In: *Quarterly Journal of Experimental Psychology* 12.1 (1960), pp. 01–16.
- [37] Tovi Grossman and Ravin Balakrishnan. “Pointing at trivariate targets in 3D environments”. In: *Proceedings of the ACM SIGCHI Conference*. ACM. 2004, pp. 447–454.
- [38] Yves Guiard. “On Fitts’s and Hooke’s laws: Simple harmonic movement in upper-limb cyclical aiming”. In: *Acta Psychologica* 82.1 (1993), pp. 139–159.
- [39] Yves Guiard, Renaud Blanch, and Michel Beaudouin-Lafon. “Object pointing: a complement to bitmap pointing in GUIs”. In: *Proceedings of Graphics Interface 2004*. Canadian Human-Computer Communications Society. 2004, pp. 9–16.
- [40] Carl Gutwin. “Traces: Visualizing the immediate past to support group interaction”. In: *Graphics interface*. 2002, pp. 43–50.
- [41] Aatif M Husain et al. “Visual evoked potentials with CRT and LCD monitors: When newer is not better”. In: *Neurology* 72.2 (2009), pp. 162–164.
- [42] ISO ISO. “ISO 9241-9 Ergonomic requirements for office work with visual display terminals (VDTs) - Part 9”. In: *International Organization for Standardization* ().
- [43] Michael Jarschel et al. “An evaluation of QoE in cloud gaming based on subjective tests”. In: *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2011 Fifth International Conference*. IEEE. 2011, pp. 330–335.
- [44] Ricardo Jota et al. “How fast is fast enough?: a study of the effects of latency in direct-touch pointing tasks”. In: *Proceedings of the ACM SIGCHI Conference*. ACM. 2013, pp. 2291–2300.
- [45] Paul Kabbash and William AS Buxton. “The “prince” technique: Fitts’ law and selection using area cursors”. In: *Proceedings of the ACM SIGCHI Conference*. ACM. 1995, pp. 273–279.
- [46] Steven W Keele. “Movement control in skilled motor performance”. In: *Psychological Bulletin* 70.6p1 (1968), p. 387.

- [47] Keiji Kikuchi. *Video game with latency compensation for delay caused by frame buffering*. US Patent 8,317,580. 2012.
- [48] Regis Kopper et al. “A human motor behavior model for distal pointing tasks”. In: *International Journal of Human-Computer Studies* 68.10 (2010), pp. 603–615.
- [49] Sylvan Kornblum, Thierry Hasbroucq, and Allen Osman. “Dimensional overlap: cognitive basis for stimulus-response compatibility—a model and taxonomy”. In: *Psychological Review* 97.2 (1990), p. 253.
- [50] Robert J Kosinski. “A literature review on reaction time”. In: *Clemson University* 10 (2008).
- [51] R Leadbetter. *Console Gaming: the Lag Factor*. 2009. URL: <http://eurogamer.net/articles/digitalfoundry-lag-factor-article>.
- [52] Yeng-Ting Lee et al. “Are all games equally cloud-gaming-friendly? an electromyographic approach”. In: *Network and Systems Support for Games (NetGames) 2012*. IEEE. 2012, pp. 1–6.
- [53] Rensis Likert. “A technique for the measurement of attitudes”. In: *Archives of Psychology* (1932).
- [54] Julian Looser, Andy Cockburn, and Joshua Savage. “On the validity of using first-person shooters for Fitts’ law studies”. In: *People and Computers XIX* 2 (2005), pp. 33–36.
- [55] I Scott MacKenzie. “A note on the information-theoretic basis for Fitts’ law”. In: *Journal of Motor Behavior* 21.3 (1989), pp. 323–330.
- [56] I Scott MacKenzie. “Fitts’ law as a research and design tool in human-computer interaction”. In: *Human-computer interaction* 7.1 (1992), pp. 91–139.
- [57] I Scott MacKenzie and William Buxton. “Extending Fitts’ law to two-dimensional tasks”. In: *Proceedings of the ACM SIGCHI Conference*. ACM. 1992, pp. 219–226.
- [58] I Scott MacKenzie and Poika Isokoski. “Fitts’ throughput and the speed-accuracy trade-off”. In: *Proceedings of the ACM SIGCHI Conference*. ACM. 2008, pp. 1633–1636.
- [59] I Scott MacKenzie, Tatu Kauppinen, and Miika Silfverberg. “Accuracy measures for evaluating computer pointing devices”. In: *Proceedings of the ACM SIGCHI Conference*. ACM. 2001, pp. 9–16.
- [60] I Scott MacKenzie, Abigail Sellen, and William AS Buxton. “A comparison of input devices in element pointing and dragging tasks”. In: *Proceedings of the ACM SIGCHI Conference*. ACM. 1991, pp. 161–166.
- [61] I Scott MacKenzie and Colin Ware. “Lag as a determinant of human performance in interactive systems”. In: *Proceedings of the INTERACT’93 and CHI’93 conference on Human factors in computing systems*. ACM. 1993, pp. 488–493.
- [62] Regan L Mandryk and Carl Gutwin. “Perceptibility and utility of sticky targets”. In: *Proceedings of Graphics Interface 2008*. Canadian Information Processing Society. 2008, pp. 65–72.

- [63] Justin Matejka, Tovi Grossman, and George Fitzmaurice. “Swift: reducing the effects of latency in online video scrubbing”. In: *Proceedings of the ACM SIGCHI Conference*. ACM. 2012, pp. 637–646.
- [64] David E Meyer et al. “Optimality in human motor performance: ideal control of rapid aimed movements”. In: *Psychological Review* 95.3 (1988), p. 340.
- [65] Atsuo Murata and Hirokazu Iwase. “Extending Fitts’ law to a three-dimensional pointing task”. In: *Human Movement Science* 20.6 (2001), pp. 791–805.
- [66] Daniel Natapov, Steven J Castellucci, and I Scott MacKenzie. “ISO 9241-9 evaluation of video game controllers”. In: *Proceedings of Graphics Interface 2009*. Canadian Information Processing Society. 2009, pp. 223–230.
- [67] Joe Padre. *Understanding touch responsiveness – Touchscreen technology series 2*. Sony Research. 2014. URL: <http://developer.sonymobile.com/2014/07/02/understanding-touch-responsiveness-touchscreen-technology-series-2/>.
- [68] Lothar Pantel and Lars C Wolf. “On the impact of delay on real-time multiplayer games”. In: *Proceedings of the 12th international workshop on Network and operating systems support for digital audio and video*. ACM. 2002, pp. 23–29.
- [69] Lothar Pantel and Lars C Wolf. “On the suitability of dead reckoning schemes for games”. In: *Proceedings of the 1st Workshop on Network and System Support for Games*. ACM. 2002, pp. 79–84.
- [70] S Pasalar, AV Roitman, and TJ Ebner. “Effects of speeds and force fields on submovements during circular manual tracking in humans”. In: *Experimental Brain Research* 163.2 (2005), pp. 214–225.
- [71] Andriy Pavlovyh and Wolfgang Stuerzlinger. “Target following performance in the presence of latency, jitter, and signal dropouts”. In: *Proceedings of Graphics Interface 2011*. Canadian Human-Computer Communications Society. 2011, pp. 33–40.
- [72] Andriy Pavlovyh and Wolfgang Stuerzlinger. “The tradeoff between spatial jitter and latency in pointing tasks”. In: *Proceedings of the 1st ACM SIGCHI Symposium on Engineering Interactive Computing Systems*. ACM. 2009, pp. 187–196.
- [73] Nicholas Lindman Port et al. “Manual interception of moving targets I. Performance and movement initiation”. In: *Experimental Brain Research* 116.3 (1997), pp. 406–420.
- [74] Luc Proteau. “On the specificity of learning and the role of visual information for movement control”. In: (1992).
- [75] Kjetil Raaen, Ragnhild Eg, and Carsten Griwodz. “Can gamers detect cloud delay?” In: *Network and Systems Support for Games (NetGames) 2014*. IEEE. 2014, pp. 1–3.
- [76] Kjetil Raaen and Andreas Petlund. “How much delay is there really in current games?” In: *Proceedings of the 6th ACM Multimedia Systems Conference*. ACM. 2015, pp. 89–92.
- [77] Jeffrey A Saunders and David C Knill. “Humans use continuous visual feedback from the hand to control fast reaching movements”. In: *Experimental Brain Research* 152.3 (2003), pp. 341–352.

- [78] Cheryl Savery and Nicholas Graham. “Reducing the negative effects of inconsistencies in networked games”. In: *Proceedings of the First ACM SIGCHI Annual Symposium on Computer-Human Interaction in Play*. ACM. 2014, pp. 237–246.
- [79] Cheryl Savery and TC Graham. “It’s about time: confronting latency in the development of groupware systems”. In: *Proceedings of the ACM 2011 Conference on Computer Supported Cooperative Work*. ACM. 2011, pp. 177–186.
- [80] Richard A Schmidt. “Control processes in motor skills”. In: *Exercise and Sport Sciences Reviews* 4.1 (1976), pp. 229–262.
- [81] Luc PJ Selen, Jaap H van Dieën, and Peter J Beek. “Impedance modulation and feedback corrections in tracking targets of variable size and frequency”. In: *Journal of Neurophysiology* 96.5 (2006), pp. 2750–2759.
- [82] Claude E Shannon and Warren Weaver. “The mathematical theory of information”. In: (1949), pp. 100–103.
- [83] Richard HY So and German KM Chung. “Sensory motor responses in virtual environments: Studying the effects of image latencies for target-directed hand movement”. In: *Engineering in Medicine and Biology Society, 2005*. IEEE. 2005, pp. 5006–5008.
- [84] R William Soukoreff and I Scott MacKenzie. “Towards a standard for pointing device evaluation, perspectives on 27 years of Fitts’ law research in HCP”. In: *International Journal of Human-Computer Studies* 61.6 (2004), pp. 751–789.
- [85] Dane Stuckel and Carl Gutwin. “The effects of local lag on tightly-coupled interaction in distributed groupware”. In: *Proceedings of the 2008 ACM Conference on Computer Supported Cooperative Work*. ACM. 2008, pp. 447–456.
- [86] Robert J Teather and Wolfgang Stuerzlinger. “Pointing at 3d target projections with one-eyed and stereo cursors”. In: *Proceedings of the ACM SIGCHI Conference*. ACM. 2013, pp. 159–168.
- [87] Namal Thibbotuwawa, Errol R Hoffmann, and Ravindra S Goonetilleke. “Open-loop and feedback-controlled mouse cursor movements in linear paths”. In: *Ergonomics* 55.4 (2012), pp. 476–488.
- [88] Luis Valente, Aura Conci, and Bruno Feijó. “Real time game loop models for single-player computer games”. In: *Proceedings of the IV Brazilian Symposium on Computer Games and Digital Entertainment*. Vol. 89. 2005, p. 99.
- [89] Rodrigo Vicencio-Moreira et al. “The effectiveness (or lack thereof) of aim-assist techniques in first-person shooter games”. In: *Proceedings of the ACM SIGCHI Conference*. ACM. 2014, pp. 937–946.
- [90] Alan Traviss Welford. In: (1968).
- [91] Thomas G Whisenand and Henry H Emurian. “Effects of angle of approach on cursor movement with a mouse: Consideration of Fitt’s law”. In: *Computers in Human Behavior* 12.3 (1996), pp. 481–495.
- [92] D Wilson. *Triple buffering: why we love it*. Anandtech. 2009. URL: <http://www.anandtech.com/show/2794/4>.

- [93] Jeremy M Wolfe. “Guided search 2.0 a revised model of visual search”. In: *Psychonomic Bulletin & Review* 1.2 (1994), pp. 202–238.
- [94] Robert Sessions Woodworth. “Accuracy of voluntary movement”. In: *The Psychological Review: Monograph Supplements* 3.3 (1899), p. i.
- [95] Aileen Worden et al. “Making computers easier for older adults to use: area cursors and sticky icons”. In: *Proceedings of the ACM SIGCHI Conference*. ACM. 1997, pp. 266–271.
- [96] Sebastian Zander, Ian Leeder, and Grenville Armitage. “Achieving fairness in multi-player network games through automated latency balancing”. In: *Proceedings of the ACM SIGCHI Conference*. ACM. 2005, pp. 117–124.
- [97] Alexander Zaranek et al. “Performance of modern gaming input devices in first-person shooter target acquisition”. In: *ACM SIGCHI Extended Abstracts on Human Factors in Computing Systems*. ACM. 2014, pp. 1495–1500.

APPENDIX A

CONSENT FORM

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF SASKATCHEWAN
INFORMED CONSENT FORM



Research Project: **Effect of input lag on first person shooter performance**
Investigators: Dr. Carl Gutwin, Department of Computer Science (966-8646)
Zenja Ivkovic, Department of Computer Science

This consent form, a copy of which has been given to you, is only part of the process of informed consent. It should give you the basic idea of what the research is about and what your participation will involve. If you would like more detail about something mentioned here, or information not included here, please ask. Please take the time to read this form carefully and to understand any accompanying information.

This study is concerned with detecting **the effect of input lag on first person shooter task performance**.

The goal of the research is to **find the effects of input lag and ways to mitigate it**.

The study will require **two sessions** of **60** minutes, during which you will be asked to perform first person shooter-like tasks in the Human-Computer Interaction Lab at the University of Saskatchewan.

At the end of the session, you will be given more information about the purpose and goals of the study, and there will be time for you to ask questions about the research. As a way of thanking you for your participation and to help compensate you for your time and any travel costs you may have incurred, you will receive a **\$5** honorarium at the end of the first session and **\$20** after the second session.

The data collected from this study will be used in articles for publication in journals and conference proceedings.

As one way of thanking you for your time, we will be pleased to make available to you a summary of the results of this study once they have been compiled (usually within two months). This summary will outline the research and discuss our findings and recommendations. This summary will be available on the HCI lab's website: <http://www.hci.usask.ca/>

All personal and identifying data will be kept confidential. Confidentiality will be preserved by using pseudonyms in any presentation of textual data in journals or at conferences. The informed consent form and all research data will be kept in a secure location under confidentiality in accordance with University policy for 5 years post publication. Do you have any questions about this aspect of the study?

You are free to withdraw from the study at any time without penalty and without losing any advertised benefits. Withdrawal from the study will not affect your academic status or your access to services at the university. If you withdraw, your data will be deleted from the study and destroyed. Your right to withdraw data from the study will apply until results have been disseminated, data has been pooled, etc. After this, it is possible that some form of research dissemination will have already occurred and it may not be possible to withdraw your data.

Your continued participation should be as informed as your initial consent, so you should feel free to ask for clarification or new information throughout your participation. If you have further questions concerning matters related to this research, please contact:

- Dr. Carl Gutwin, Professor, Dept. of Computer Science, (306) 966-8646, gutwin@cs.usask.ca

Your signature on this form indicates that you have understood to your satisfaction the information regarding participation in the research project and agree to participate as a participant. In no way does this waive your legal rights nor release the investigators, sponsors, or involved institutions from their legal and professional responsibilities. If you have further questions about this study or your rights as a participant, please contact:

- Dr. Carl Gutwin, Professor, Dept. of Computer Science, (306) 966-8646, gutwin@cs.usask.ca
- Research Ethics Office, University of Saskatchewan, (306) 966-2975 or toll free at 888-966-2975.

Participant's signature: _____

Date: _____

Investigator's signature: _____

Date: _____

A copy of this consent form has been given to you to keep for your records and reference. This research has the ethical approval of the Research Ethics Office at the University of Saskatchewan.

APPENDIX B

INDEX OF DIFFICULTY CALCULATIONS (2D AND 3D)

```
void AcqTargetMaker::makeDiffs() {
    const int circleRadiusSteps = 5;
    const int distanceSteps = 3;

    float maxDist = maxDeflection;
    float maxWidth = maxRadius;
    float minDist = maxDist / (float)circleRadiusSteps;
    float minWidth = maxWidth / (float)distanceSteps;
    float maxDwRatio = maxDist / maxWidth;

    // For 3D
    float maxDistAngle = atan(maxDist / targetDepth);
    float minDistAngle = maxDistAngle / (float)circleRadiusSteps;
    float maxWidthAngle = maxDistAngle / maxDwRatio;
    float minWidthAngle = maxDwRatio * minDistAngle;

    for (int distStep = 2; distStep <= circleRadiusSteps; distStep++) {
        for (int widthStep = 1; widthStep <= distanceSteps; widthStep++) {
            Difficulty diff;
            if (isFlat) { // current condition is 2D
                // Scaling to get appearance in pixels about same as 3D
                float visualScaleFactor = 0.61f;

                diff.distance = minDist * distStep * visualScaleFactor;
                diff.width = minWidth * widthStep * visualScaleFactor;
                diff.id = calcId(diff.distance, diff.width);
            } else {
                // Find the desired angle then work back to world coordinates
                float distAngle = minDistAngle * distStep;
                float widthAngle = minWidthAngle * widthStep;

                diff.distance = targetDepth * tan(distAngle);
                float distToTarget = targetDepth / cos(distAngle);
                diff.width = 2.0f * distToTarget * tan(widthAngle / 2.0f);
                diff.id = calcId(diff.distance, diff.width);
            }
            difficulties.push_back(diff);
        }
    }
}

float AcqTargetMaker::calcId(float distance, float width) {
    if (isFlat) {
        return log2((distance / width) + 1.0f);
    } else {
        float distanceAngle = atan(distance / targetDepth);
```



```
// Actual distance, not just along y axis
float distToTarget = targetDepth / cos(distanceAngle);
float widthAngle = 2.0f * atan(width/2.0f / distToTarget);

return log2((distanceAngle / widthAngle) + 1.0f);
}
}
```

APPENDIX C

DEMOGRAPHIC SURVEY



Lag study

0%

User ID

Gender

- Male
 Female

Handedness 1

Which hand do you use to maneuver the mouse when you play FPS games that require a mouse?

- Left
 Right

Primary Language

What is the language you speak at home (to parents/kids/etc)?

Computer Use

How many hours per week do you typically use a computer?

Not at all 80 hours

First Person Shooter (FPS) game validation

Do you play or have you played first person shooter (FPS) games that require the use of a mouse?

- Yes
 No

Current FPS Usage

How often do you currently play FPS games that require the use of a mouse?

- I never played FPS games
 I don't play FPS games anymore
 less than an hour a week

- 1-5 hours a week
- 6-10 hours a week
- 10-20 hours a week
- 20-30 hours a week
- more than 30 hours a week

Former FPS Usage

If you used to play FPS games more regularly in the past that required the use of the mouse, how often would you play those games?

- I have never played FPS games
- less than an hour a week
- 1-5 hours a week
- 6-10 hours a week
- 10-20 hours a week
- 20-30 hours a week
- more than 30 hours a week

Time since peak

How long has it been since your peak days of FPS gaming?

- I'm currently at my peak
- Less than half a year
- Less than a year
- Less than two years
- Less than five years
- Less than eight years
- Longer than eight years

FPS Skill Level

In your peak days of online FPS gaming, how well did you place in the match scoreboard, on average?

- top 10%
- top 25%
- top 50%
- top 75%
- below top 75%

What kind of FPS games do you primarily play?

- Offline
- Unrealistic, fast paced (like Quake, UT)
- Unrealistic, slow paced
- Realistic, fast paced (like Call of Duty)
- Realistic, slow paced (like ARMA)

Rate your skill level in FPS games

Extremely Unskilled Extremely Skilled

Do you have more experience with gaming with a gamepad or gaming with a mouse?

- A little more experience with a gamepad
- A lot more experience with a gamepad
- A little more experience with a mouse
- A lot more experience with a mouse

Do you or have you previously spent a significant amount of time playing non-FPS games with a mouse?


- Yes
- No

Next

APPENDIX D

BETWEEN-BLOCK SURVEY (NON-MITIGATED)





3%

Rate your experiences and agreements with the following statements

	strongly disagree	moderately disagree	slightly disagree	neutral	slightly agree	moderately agree	strongly agree
I performed well this round	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Lag affected my performance	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
This round was frustrating	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The controls were laggy	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>



APPENDIX E

BETWEEN-BLOCK SURVEY (MITIGATED)



3%

Rate your experiences and agreements with the following statements

	strongly disagree	moderately disagree	slightly disagree	neutral	slightly agree	moderately agree	strongly agree
I performed well this round	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Lag affected my performance	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
This round was frustrating	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The controls were laggy	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
I noticed my aim being assisted	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Back

Next



© University of Saskatchewan [Contact](#) [Maps](#) [Search](#) [Disclaimer](#) [Privacy](#)



APPENDIX F

END OF SESSION SURVEY



Baserate Sequence Survey

96%

Rate your experiences and agreements with the following statements

	strongly disagree	moderately disagree	slightly disagree	neutral	slightly agree	moderately agree	strongly agree
Aim assists improved my performance	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
I preferred having my aim assisted compared to today's non-assisted experience	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
I would want to see aim assists being used to mitigate lag in commercial games	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>



© University of Saskatchewan [Contact](#) [Maps](#) [Search](#) [Disclaimer](#) [Privacy](#)

