Supporting Aim Assistance Algorithms through a Rapidly Trainable, Personalized Model of Players' Spatial and Temporal Aiming Ability

Adrian L. Jessup Schneider
Queen's University
Kingston, Ontario, Canada
adrian.schneider@queensu.ca

T. C. Nicholas Graham
Queen's University
Kingston, Ontario, Canada
nicholas.graham@queensu.ca

ABSTRACT

Multiplayer digital games can use *aim assistance* to help people with different levels of aiming ability to play together.

To dynamically provide each player with the right amount of assistance, an aim assistance algorithm needs a model of the player's ability that can be measured and updated during gameplay. The model must be based on difficulty parameters such as target speed, size, and duration, that can be adjusted in-game to change aiming difficulty, and must account for player's spatial and temporal aiming abilities.

To satisfy these requirements, we present the novel dynamic spatiotemporal model of a player's aiming ability, based on difficulty parameters that can be manipulated in a game. In a crowdsourced experiment with 72 participants, the model was found to accurately predict how close to a target a player can aim and to converge rapidly with a small set of observations of aiming tasks.

CCS CONCEPTS

• **Human-centered computing** → **HCI theory, concepts and models**.

KEYWORDS

game balancing, dynamic difficulty adjustment, dynamic player balancing, aim assistance, personalization

ACM Reference Format:
Adrian L. Jessup Schneider and T. C. Nicholas Graham. 2023. Supporting Aim Assistance Algorithms through a Rapidly Trainable, Personalized Model of Players' Spatial and Temporal Aiming Ability. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23), April 23–28, 2023, Hamburg, Germany*. ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3544548.3581293

1 INTRODUCTION

Players of digital games typically differ in their success with game tasks such as aiming. This can be a result of differences in players' motor ability [65], familiarity with the type of game being played [56], or their investment in "dedicated practice and determination" [31]. For example, children with cerebral palsy may have deficits in fine-motor control that make it difficult to aim with a joystick [48], while other players may be confused by a new game's complex displays [56]. In this paper, we focus specifically on differences in aiming ability, which is influenced by all of the factors listed above, including abilities in fine motor control, visual-motor integration, and visual-spatial processing [27].

It is important for games to provide techniques allowing people with different abilities to play together. Social equity requires that persons with disabilities should be able to play with their friends [54]. More broadly, video game play has become an important form of social interaction; according to the Entertainment Software Association, 83% of game players play with others at least weekly, and 61% report that video games have helped them stay connected to friends and family [15]. Friends or family members should not be excluded from such social interaction due to differences in game-playing ability.

One approach to accommodating such differences is to reduce the difficulty of a game for players with lower ability. This can even be done during gameplay, using a technique called *dynamic difficulty adjustment* (DDA). In order to perform this adjustment for aiming, though, a DDA algorithm must understand what to adjust, and by how much. To do this, the algorithm must observe a player playing the game to be personalized, and then form a model of that player's level of aiming ability, as illustrated in Figure 1.

While there exist many DDA algorithms in existing literature, they emphasize the difficulty adjustment itself, corresponding to the "adjust aiming tasks" step. There is a lack of a systematized way to calculate *how much* to adjust difficulty, based on player ability.

The core contribution of this paper addresses this problem by providing a new technique for dynamically modelling individual players' spatial and temporal aiming abilities. Specifically, we developed a dynamic spatiotemporal model of aiming ability, building upon the work of Huang *et al.* [29, 30] and Lee *et al.* [38]. Our model is not in itself a DDA algorithm, but a support tool for such algorithms, occupying the "model aiming ability" step in Figure 1. The model uses observations of a player performing aiming tasks to determine how to adjust the aiming tasks for that player. Our model possesses the following key properties:

- Converges rapidly with small quantities of data allowing the dynamic balancing to begin quickly.
- Accurately models the aiming ability of an individual, accounting for both the spatial and temporal aspects of aiming.

CHI '23, April 23–28, 2023, Hamburg, Germany

ACM ISBN 978-1-4503-9421-5/23/04…$15.00
https://doi.org/10.1145/3544548.3581293

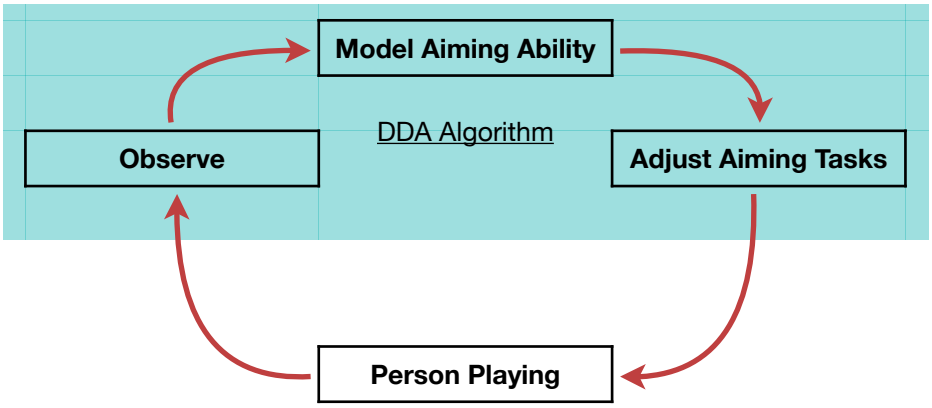


Figure 1: Diagram of the main loop of a dynamic difficulty adjustment (DDA) algorithm. The algorithm observes player performance, models the player's aiming ability, and adjusts the difficulty of aiming tasks accordingly. This paper addresses the topmost box, "Model Aiming Ability".

- Requires only data that can be collected during play, allowing it to be trained directly within a game by presenting targets of varying difficulty.

To our knowledge, our spatiotemporal model is the first to possess all these qualities. The paper is structured as follows. We first review related work in aim assistance, dynamic difficulty adjustment, and modelling the difficulty of aiming tasks. We then present our dynamic spatiotemporal model of aiming ability, including the method for training the model for an individual. We detail the design of a 72-participant online study, as well as 108-participant and 36-participant pilot studies, and report the results of all three. We conclude with discussion of how these results inform the development of dynamic balancing algorithms for aim assistance.

2 RELATED WORK

To allow groups of people to play together, games require mechanisms that account for differences in players' abilities. In this section, we review commonly used mechanisms, with particular focus on aim assistance algorithms and dynamic difficulty adjustment (DDA). Underlying these algorithms is the need to determine the abilities of individual players to allow personalization of the algorithms. We therefore complete this section with a review of existing models of the difficulty of aiming tasks [28–30, 38].

Player balancing can be defined as "adjusting game parameters to boost performance of weaker players" [10, 23]. Player balancing is well-represented in existing research, but some aspects have received substantially more attention than others. In particular, the majority of the literature on aim assistance involves tasks with spatial constraints, such as clicking in a particular part of the screen (Section 2.1). In contrast, very little research has been published regarding aiming tasks that also have temporal constraints, in which the click or other selection action must also be performed at a particular time. Temporal aiming, or *temporal pointing*, can be defined as "a discrete selection of a target about to appear for a bounded time window." [39]. For example, in the game Whac-a-Mole, the player must hit a mole between the time it pops up from its hole and the time it sinks back down [51]. This game involves a spatial aiming task (hit the right hole) and a temporal aiming task (hit when the mole is visible).

Many algorithms have variable settings that must be tailored to the abilities of the players. If the balancing algorithm is applied too weakly, it will be insufficient to provide balance; too strongly, and the formerly weaker player may now have an unfair advantage. Additionally, the algorithm must be able to perform this adjustment rapidly, to both identify and adapt to a player's initial level of skill, and to closely track changes in their performance over time [2]. The field of dynamic difficulty adjustment has produced techniques for accurately modelling player ability; however, these techniques often require significant training time [11, 40], address non-aiming forms of play [61], or are not easily mapped to the required strength of assistance [9]. To our knowledge, there have also been no previous attempts to model the temporal aiming ability of players for the sake of personalizing aim assistance.

This section starts with an overview of aim assistance, particularly the distinction between approaches based on spatial constraints and the few approaches that apply to tasks with temporal constraints. We then give a brief review of DDA, with some examples of DDA algorithms. Afterward, we give an overview of models that estimate aiming difficulty using mathematical parameters, theoretically enabling their use to estimate player ability for the purpose of personalizing balancing algorithms.

2.1 Aim Assistance

Aim assistance, generally, refers to skill assistance intended to help with aiming tasks. Aiming can be defined as "the ability to hit a target" [45]. For example, hitting a moving or distant opponent in a shooting game requires aiming. Vicencio-Moreira et al. define aim assistance as "algorithmic changes that alter the accuracy of targeting movements" [58].

The importance of games as a social activity motivates people to want to play together, even if they have different abilities [62]. Persons with motor disabilities can be particularly excluded from gaming with others [27, 65]. Aim assistance can allow players to

play together on a more even level in multiplayer games, or even to tailor the challenge of single player games to an appropriate level to players with more or less ability.

Some degree of aim assistance is also included in nearly all shooter games intended to be played with a game controller instead of a mouse and keyboard, due to the lower aiming precision the gamepad affords [21, 47, 59]. Shooters employing aim assistance include the Halo series [24, 59], the Gears of War series [34, 59], the Modern Warfare series [25, 59], the Call of Duty series [25, 59], and Fortnite [8].

The above games also use aim assistance for the purposes of balancing. These games support cross-play between console players and PC players. The presence of analog stick aim assistance in these games therefore serves not only to increase the usability of game controllers in the abstract, but also to balance between players using different input hardware.

In many games, aiming has both spatial and temporal aspects. For example, consider a game that simulates a pistol duel. A player might miss their opponent by clicking in the wrong part of the screen: a spatial miss. However, they may also take aim too slowly, leading to the opponent winning the draw and shooting first: a temporal miss.

In this section, we describe common types of spatial aim assistance, as well as some algorithms that can apply to temporal aiming tasks..

2.1.1 Spatial Aim Assistance. The following lists major types of aim assistance, all of which function on spatial principles. For simplicity, we refer to the action of a player performing the selection part of aiming as "clicking," even though these techniques can apply, for example, to touch-based interfaces.

Area Cursor The player's cursor is expanded in size to make it easier to successfully aim at a target. If a player's cursor is not exactly on the target when the player clicks, area cursors allow it to nonetheless be close enough to score a hit. Area cursors can be applied in a variety of games, for both 2D and 3D aiming tasks [4, 5, 35, 57–59].

Bullet Magnetism Projectiles veer toward nearby targets, as though drawn by a magnet. If the target is close enough, a projectile that would have missed on a straight path will instead score a hit [5, 35, 52, 57–59].

Sticky Targets Once a player has successfully aimed their cursor at or near a target, Sticky Targets will slow the cursor if it begins to move away, as though the cursor were stuck to the target. This helps players avoid accidentally moving the cursor too far, and aiming past the target rather than at it [1, 4, 5, 35, 44, 52, 59].

Target Gravity When the player moves their cursor, Target Gravity steers the cursor toward nearby targets, as though attracted by a gravitational force [4, 5, 35, 52, 59]. It can also be called Reticule/Reticle Magnetism, in analogy to Bullet Magnetism [21, 25].

Target Lock The player's cursor snaps directly to a target, and remains pointing at that target unless forced to switch to another [5, 23, 35, 52, 59]. Because of its power to make aiming easier, Target Lock is rare in shooter games, in which aiming is a primary component of gameplay.

2.1.2 Temporal Aim Assistance. While there are ample examples of aim assistance algorithms that function on spatial principles, there are few designed to work for tasks with temporal constraints. However, there are existing algorithms which have effects in the temporal dimension, despite not being created for temporal balancing. The Comet variant on the Area Cursor technique, for example, increases the size of the target in the direction opposite its movement [26]. Increasing the size of the target is a spatial modification, but the Comet helps players who click in the right place but too late, and therefore miss temporally.

Overall, while spatial aim assistance algorithms are well-studied, algorithms that address temporal aiming are rare, and usually do so indirectly.

2.2 Dynamic Difficulty Adjustment

Dynamic difficulty adjustment (DDA) can be defined as "modulating in-game systems to respond to a particular player's abilities over the course of a game session." [33] Using DDA for balancing can raise the ceiling on how well the game is able to adapt to a player's needs, through being able to keep pace with changes to a player's performance during play.

In "Extending Reinforcement Learning to Provide Dynamic Game Balancing," authors Andrade *et al.* set out three requirements for a dynamic balancing algorithm to accomplish [2]:

(1) adapt to a player's initial level of ability as rapidly as possible
(2) follow changes in a player's ability as closely as possible over time
(3) avoid having the balancing assistance be too obvious or jarring

The third item has been found to be important to the reception of DDA-based assistance algorithms. Even if players are aware of the theoretical existence of DDA in a game, algorithms that too obviously show when they are active can be disruptive [3, 37].

DDA is a growing field, as noted by Zohaib in "Dynamic Difficulty Adjustment (DDA) in Computer Games: A Review" [68]. Many algorithms exists with a variety in focus and technique, some very different from what we are investigating. In the following section, we note a few DDA algorithms that, at least in part, are designed to answer questions similar to those that we address in this paper.

Dynamic Level Generation Wheat *et al.* use DDA to procedurally build 2D platformer levels, tailored to the difficulty needs of individual players [61]. This algorithm uses in-game measures to judge the ability level of players, such as how long a player takes to complete a level, and how many points they collect.
The procedural level generator was able to consistently increase rated enjoyment from players over the course of play, but it is not clear if it would be able to function outside its native environment of 2D platformers.

Emotional Affect DDA Physiological data from players can be used to detect the emotional state of players, and to inform DDA. If the system detects that a player is bored, the difficulty of the game can be increased; likewise, when a player is stressed, the difficulty of the game can be reduced [11, 40].

These algorithms require extensive hardware for measuring affect, including sensors for heart rate, galvanic skin response, body temperature, and brain activity, and also require substantial training time to correctly interpret individuals' physiological data.

Modelling Skill from Player Inputs Buckley *et al.* demonstrate that a modelling algorithm using random forests (multiple decision trees working collaboratively) can use player inputs, like character movement and aiming, to train a model of player skill [9]. This algorithm uses only data that can be collected easily during play by the game itself, and achieved a 76% prediction accuracy with only one minute of training. However, the authors do not demonstrate a method of using these skill evaluations to judge how to adjust difficulty in a DDA algorithm. Since the model is based on player inputs rather than anything that can be controlled by the game, the model itself is not designed to facilitate or direct such a conversion.

Aim Assistance DDA Vicencio-Moreira *et al.* tested a method of using DDA for player balancing by directly applying aim assistance algorithms – area cursor and bullet magnetism – in a first-person shooter game [57]. In testing, the matches did not become measurably more balanced between players. The authors note as a major reason that the DDA algorithm required a player to fall behind before it took effect, often leaving players too far behind to catch up before being granted significant assistance.

The findings of this study suggest that using a proxy—here, score—for aiming may not be fast or accurate enough to characterize ability for aiming-based DDA. Ideally, DDA for aim assistance should be able to measure aiming ability directly, for the sake of rendering the required level of assistance as soon as possible.

Despite the considerable diversity of DDA algorithms, we are aware of none that meet our own requirements, including rapidly and accurately training a model of an individual's aiming ability using difficulty parameters that can be directly modified to adjust difficulty (as explained in Section 3). Insufficiencies in existing algorithms include lacking a treatment of temporal aiming, requiring extensive training (ex: [11, 40]), or not using directly modifiable in-game parameters like target size and speed (ex: [9]).

2.3 Modelling Difficulty of Aiming Tasks

Most aim assistance algorithms have parameters that can be adjusted: how large is an area cursor? What is the magnitude of bullet magnetism? How sticky is a sticky target? These parameters can be adjusted to personalize the algorithm to an individual. For those with weaker aiming ability, area cursors should be bigger, bullets more magnetic, or targets more sticky.

Such personalization requires two things: a measure of the player's ability, and a conversion from that measure to the numerical strength settings of the balancing algorithms. In many cases, this process is performed informally, for example, by a physiotherapist estimating a player's ability through observation and iteratively finding the appropriate personalization settings. To provide more accurate personalization, a mathematical model of an individual's ability can be created, to weigh inputs and use them to determine an appropriate strength of assistance for an individual.

Currently, we are aware of no model that serves this purpose for aim assistance.

One obvious candidate for such a model might be Fitts' Law [18, 43]. Fitts' Law states the difficulty of an aiming tasks as follows:

$$Difficulty = \log_2 \frac{2A}{W}$$

in which A is movement amplitude – the initial distance between the target and the cursor – and W is the width of the target. However, prior research has found Fitts' Law to be less applicable to aiming tasks with temporal constraints [53, 64, 66], while research by Huang *et al.* for their Ternary-Gaussian aiming model (see below) find that movement amplitude is not a predictor for the distribution of a player's clicks around a moving target [28–30].

There are two models in the existing literature, however, that provide a starting point. These model difficulty of aiming *tasks*, not individuals' aiming ability. These existing models are Huang *et al.'s* Ternary-Gaussian aiming model [29], in particular its 2D variant [30], and Lee *et al.*'s Cue Integration aiming model [38]. The former estimates the difficulty of clicking on moving targets – a spatial constraint – while the latter estimates the difficulty of selecting a moving target at a particular time – a temporal constraint. While these were not developed for modelling a person's aiming ability, as we shall see in Section 3, the input parameters used by these models are a useful starting point for the development of a model suited to our purposes.

Given that the Ternary-Gaussian model is focused only on spatial constraints and the Cue Integration model only on temporal constraints, it should be possible to use them together to form a composite spatiotemporal model of aiming. Indeed, the authors of the models showed in a collaborative publication that the models are compatible [28]. This collaborative publication presented 12 participants with a task requiring them to successfully click a moving target at a correct time. Each participant was presented with a targets of varying difficulty over the course of the trial, with the results able to train the model to estimate the overall difficulty of any given target. The authors successfully demonstrated that by multiplying the spatial and temporal predictions, the two models in conjunction were more accurate in predicting spatiotemporal accuracy – whether a participant clicks in the right place *and* at the right time – than either model alone [28].

2.3.1 Ternary-Gaussian Aiming Model. The Ternary-Gaussian aiming model uses *endpoint distribution* as its measure of performance. A *spatial endpoint* is the location of a player's click, relative to the target. If the distribution of the endpoints tightly clusters on top of the target, then that player is aiming accurately.

It is well-established in target selection literature that endpoint distributions around a stationary target follow a Gaussian curve, with endpoints clustering around the target, both for 1-dimensional [12, 13, 18–20, 41–43, 60, 63] and 2-dimensional [6, 7, 46, 67] aiming tasks. In developing the Ternary-Gaussian model, authors Huang *et al.* demonstrated that endpoint distribution is also Gaussian for moving targets [29]. In addition to endpoint distribution, the model can also predict error rate, by mapping the Gaussian curve onto the

size of the target, and computing the probability that an endpoint from that curve will fall within the target [29, 30].

The endpoint predictions are influenced by a set of difficulty parameters. In the original form of the model, these are the width and speed of the target. The 2D variant adds the height of the target as an additional parameter.

2.3.2 Cue Integration Model. The Cue Integration model measures and predicts player performance on aiming tasks with temporal, rather than spatial, constraints [38]. Like the Ternary-Gaussian model, the Cue Integration model's measure of performance is a Gaussian distribution of endpoints. A *temporal endpoint* is the specific time that the player clicked, relative to the temporal target. The authors demonstrate that, as with spatial endpoints, the distribution of temporal endpoints follows a Gaussian curve.

Similarly to the Ternary-Gaussian model, the Cue Integration model's endpoint predictions can be used to predict error rate, through mapping the distribution onto the size of the temporal target.

The difficulty parameters for the Cue Integration model are the length of the time in which the target is clickable (temporal width), and the player's ability to perceive when to click (effective temporal distance). The latter is constructed through integrating two temporal cues, giving the model its name: cue viewing time (time between the target appearing and becoming clickable) and period of input repetition (rate at which repeating targets appear).

Our model uses a subset of these six difficulty parameters (width, height, speed, temporal width, cue viewing time, period of input repetition) to define the difficulty of an aiming task, as described in Section 3.1.

In this section, we have provided an overview of aiming and aim assistance, with an emphasis on the division between spatial aim and temporal aim. We also described dynamic difficulty adjustment. Finally, we described two models for characterizing the difficulty of aiming tasks, one for spatial aiming and one for temporal aiming. Despite the fact that they address difficulty of aiming tasks rather than modeling individuals' aiming ability, these two models influenced the design of our own spatiotemporal model as described in the following section (Section 3).

As the literature stands, we are aware of no previous model fully able to close the loop seen in Figure 1, rapidly modelling the aiming ability of an individual by observing aiming tasks based on parameters that can be directly adjust in-game. *Existing approaches require extensive training time, require data that can not be gathered in-game, or use inputs that can not be manipulated to adjust difficulty.* As we shall see, our dynamic spatiotemporal model fills these gaps, while accounting for both spatial and temporal aiming ability.

In the remainder of the paper, we fully describe this new model, as well as the study we conducted to assess it.

3 THE DYNAMIC SPATIOTEMPORAL MODEL FOR ASSESSING AIMING ABILITY OF INDIVIDUAL GAME PLAYERS

The core contribution of this paper is a new technique for dynamically modelling individual players' spatial and temporal aiming abilities. We experimentally demonstrate that the model's predictions are accurate and that it can be trained quickly, making it suitable for use in dynamic player balancing algorithms. Such a model of individuals' aiming abilities is critical to the personalization of aim assistance algorithms, such as presented in Section 2.1, so that people who are weaker at aiming receive enough assistance to make them competitive, and not so much assistance as to imbalance the game.

The model's suitability for dynamic adaptation derives from two main properties, which will be demonstrated by the study reported in Section 4.

First, the **model requires only data that can be collected during play**, by measuring the player's performance aiming at targets of varying difficulty. This permits the model to be trained dynamically during play by presenting targets of differing difficulty until enough aiming actions have been observed. The model is also designed to be dynamically adaptable to changes in player ability, by incorporating newer data and discarding older data.

Second, the **model converges rapidly with small quantities of data**. As we shall see, each component of the model – spatial and temporal – can be trained to near its peak accuracy using only 45 datapoints. The 90 datapoints required to train both the spatial and temporal components can be collected rapidly (e.g., only three minutes in our test game *ChronoSwarm*). This permits a dynamic balancing algorithm to become usable very quickly, and begin using live game data to refine its predictions after only a brief initial training session.

In addition to these properties, our model is the first to include individuals' temporal aiming ability, making it suitable for spatial and temporal difficulty adjustment.

We developed this model through iterations of testing and redesign, including two large-scale crowdsourced pilot studies described in Appendix A. The final model is presented below, and is evaluated through a final study presented in Section 4.

3.1 Difficulty Parameters

We model difficulty of aiming through a set of difficulty parameters, a subset of those proposed by Huang *et al.* and Lee *et al.* (Section 2.3). These difficulty parameters are summarized in Table 1. Varying these parameters allows us to adjust the difficult of an aiming task; for example, making a target smaller, faster, or available for a briefer time makes the target harder to hit. As we shall see in Section 3.3, the model is trained by presenting players with aiming tasks that vary in difficulty across these parameters.

The parameters for the spatial aiming component of our model are the width of the target (called tangent width in the model), and the speed of the target (Table 1). Width is defined relative to the direction in which the target is moving. The width, or *tangent* axis, is parallel to the direction of movement, while the height, or *normal* axis, is perpendicular to the target's movement. This coordinate system makes the target effectively stationary in the normal axis, permitting the effects of the two axes to be cleanly separated.

In many games only one of width and height need be considered, since targets typically maintain the same height relative to width, merely growing or shrinking with distance. In this case, width is

Spatial Difficulty Parameters		*Temporal Difficulty Parameters*	
Parameter	**Description**	**Parameter**	**Description**
Tangent Width	Width of the target relative to its direction of movement	Temporal Width	Length of time the target is clickable
Speed	Rate of the target's movement	Cue Viewing Time	Time between the target appearing and becoming clickable

Table 1: Spatial difficulty parameters drawn from the Ternary-Gaussian Aiming model [29, 30] and temporal difficulty parameters drawn from the Cue Integration model [38].

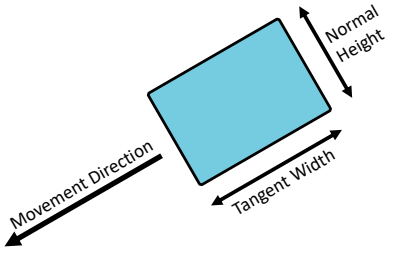


Figure 2: The horizontal "tangent" axis and vertical "normal" axis are defined relative to the direction of the target's movement.

the preferred parameter, given that it has a much greater effect on difficulty than height (see Section A.2).

The parameters for the temporal aiming component of our model are: the length of the time during which the target is clickable (*temporal width*) and the time between the target appearing and becoming clickable (*cue viewing time*). Pilot testing showed that Lee *et al.*'s period of input repetition parameter had no significant predictive effect, so was omitted from our model (Section A.1).

Pilot testing also showed that spatial parameters had little or no predictive effect on temporal aiming, and temporal parameters had little or no predictive effect on spatial aiming (Section A.2). This confirmed that spatial and temporal aiming are measurably distinct abilities, and therefore justified separate training of spatial and temporal aiming in the model. As we shall see in Section 5.3, this insight is critical to the fast training of the model.

3.2 The Dynamic Spatiotemporal Model

The intended use for our model is to characterize the aiming ability of an individual. The model is trained by presenting that individual with aiming tasks of varying difficulty. This is accomplished by varying the difficulty parameters used by the model as predictors, allowing us to train the model to predict how closely that individual can aim at targets both spatially and temporally.

This section provides the math behind the model, explaining how these observations of an individual's play are turned into a function that predicts their success given a particular difficulty of target. Success is expressed as a probability distribution representing the likelihood of hitting at a particular distance from the target.

One of the major requirements of the mathematical form of our model was the need to model player performance in aiming tasks separately for spatial constraints (click in the right place) and temporal constraints (click at the right time). This we accomplished by using separate algorithms for the two components.

In our new dynamic spatiotemporal model, the spatial component of the model is the 2D Ternary-Gaussian model [30], while the temporal component is novel, combining elements of both the Ternary-Gaussian [29] and Cue Integration [38] models.

First, the model of a player's aim in the tangent axis (parallel to direction of movement) is in the form of a Gaussian distribution, with mean given as:

$$\mu = a_s + b_s V + c_s W$$

in which V is speed, W is tangent width, and a_s, b_s, and c_s are model parameters whose values are determined when training the model. Standard deviation for the tangent axis is given as:

$$\sigma = \sqrt{d_s + e_s V^2 + f_s W^2 + g_s \frac{V}{W}}$$

in which V is speed, W is tangent width, and d_s, e_s, f_s, and g_s are the model parameters.

In the normal axis (perpendicular to movement), the model assumes that mean in the normal axis is zero, while standard deviation is given as:

$$\sigma = \sqrt{h_s + i_s V^2 + j_s H^2}$$

in which V is speed, H is normal height, and h_s, i_s, and j_s are the model parameters.

While the difficulty parameters for temporal aiming are sourced from the Cue Integration model [38], removing period of input repetition as a parameter requires a different structure. Our pilot testing showed that using the structure used in the spatial tangent axis gave comparable performance when used with the remaining temporal parameters. The mean and standard deviation for the temporal component of the model are given as:

$$\mu = a_t + b_t C + c_t W_t$$

$$\sigma = \sqrt{d_t + e_t C^2 + f_t W_t^2 + g_t \frac{C}{W_t}}$$

in which C is cue viewing time, W_t is temporal width, and a_t, b_t, c_t, d_t, e_t, f_t, and g_t are the model parameters.

In the following section, we describe the process for training the model for an individual, and how it can be used to personalize aiming difficulty for that individual.

3.3 Training the Model

Our dynamic spatiotemporal model is designed for use within a dynamic difficulty adjustment (DDA) algorithm. To do this, the model follows three steps, as visualized in Figure 1. The first two

steps are demonstrated experimentally (Section 4). The third step is specific to the DDA algorithm being used, and therefore is not included in the study design.

First, the model requires us to **observe** the player performing aiming tasks of different difficulty levels. These difficulty levels are defined using the difficulty parameters laid out in Section 3.1. In defining the difficulty levels, it is important to vary the difficulty parameters independently, so the model can be sensitive to any interactions between them. Since the spatial and temporal components of the model are independent, crossing the spatial parameters together and the temporal parameters together provides sufficient coverage of the condition-space.

The second step is to **model aiming ability** using these observations, using a process drawn from Huang *et al.* [30]. The player's endpoints for each combination of difficulty parameters first must be estimated as a Gaussian distribution; our tests use maximum likelihood estimation for this purpose (MATLAB's "mle" function). These estimates are then used to fit the model parameters $a_s...j_s$ and $a_t...g_t$; our tests use nonlinear regression for this fitting (MATLAB's "nlinfit" function). By inputting arbitrary values of the difficulty parameters into the model, it can now predict the player's aim under those difficulty conditions, in the form of Gaussian distributions of where (spatial) and when (temporal) the player is most likely to click.

The third step is to **adjust aiming tasks** using the model's understanding of the player's aiming ability. If a player is struggling with aiming in a game, then the model can predict how much the player's aim will improve if the value of a given difficulty parameter is altered to be easier. By comparing several such easier values, the model is able to predict *how much* the difficulty parameter must change for that player's aiming performance to rise to the desired level. This can be accomplished separately for spatial and temporal aiming due to the independence of those components; if a player has good spatial aim but poor temporal aim, or vice versa, then the model can adjust the relevant difficulty parameters only.

3.4 Differences from Ternary-Gaussian and Cue Integration Models of Aiming

Our dynamic spatiotemporal model draws significantly from the Ternary-Gaussian model developed by Huang *et al.* [29, 30] and the Cue Integration model developed by Lee *et al.* [38], as well as Huang and Lee's collaboration paper that used both models [28]. The model also differs in significant ways, as detailed below.

The first difference is in how the model is used. The Ternary-Gaussian and Cue Integration models are both designed and tested for use in predicting the difficulty of aiming tasks, by aggregating data across multiple participants. In contrast, our emphasis is on predicting the performance of individuals: we wish to characterize the difficulty of aiming tasks *for an individual* rather than in general.

The second difference is that our method is designed to increase the speed of training. Speed of training was not a priority for the Ternary-Gaussian or Cue Integration models. For example, while Huang and Lee do not specify how long participants required to complete the spatiotemporal collaboration study, each participant was required to perform 1,080 aiming tasks [28] which would be impractical for an algorithm that is to be trained in real-time.

To emphasize speed in training, we tightly control the size of the condition-space by reducing the number of difficulty parameters and maintaining independence between spatial and temporal parameters. As we will see in Section 5.3, our model can be trained with 45 spatial and 45 temporal aiming tasks.

The third difference applies specifically to the temporal component of our model. The spatial component of the dynamic spatiotemporal model uses Huang *et al.*'s 2D Ternary-Gaussian model [30]. However, while the temporal component of the model uses difficulty parameters drawn from Lee *et al.*'s Cue Integration model [38], the formulation of the temporal component is different.

3.5 A Note on the Separation of Spatial and Temporal Components

As noted in Section 3.2, one of our goals in designing our model was to be able to separately model spatial and temporal aiming. This separation is critical to being able to characterize a player's ability in both spatial and temporal aiming.

For example, consider the pistol duel game example from 2.1: a player must click in the right place—the location of the opponent—and at the right time—after the start of the duel but before the opponent can shoot back. Increasing the size of the opponent—spatial assistance—will have little to no benefit if the player has no trouble clicking on the opponent, but is unable to react in time to the start of the duel; such a player needs *temporal* assistance. By modelling spatial and temporal aiming separately, we are able to more precisely understand how players aim.

The separation of the components is also critical to rapid training, in order to reduce the condition space of the training data. Using square targets, there are two spatial parameters—width and speed—and two temporal parameters—cue viewing time and temporal width. To train the model, each parameter must vary across multiple values. For example, in our study we use three values, representing easy, medium, and hard difficulty. With three values of each of the four conditions, the total condition-space would be 81 conditions if fully crossed: $3^4 = 81$. By separating the components, only 18 are needed to train both: $3^2 + 3^2 = 18$.

In the case that a combined spatiotemporal prediction is required, then the individual predictions can be multiplied together, as demonstrated by Huang and Lee [28]. For example, a player with 90% spatial accuracy and 70% temporal accuracy has $90\% \times 70\% = 63\%$ spatiotemporal accuracy. As detailed in Section 3.1 and Appendix A, this separation is justified since spatial parameters have little predictive power over temporal aiming ability and vice versa.

The next section describes the experiment we performed to assess this model's performance.

4 EXPERIMENTAL DESIGN

We performed a crowdsourced experiment with 72 participants to address our core questions of:

RQ1 *How much do individuals vary in aiming ability?* Given that our research is motivated by the presence of individual differences in aiming ability, it behooves us to characterize how large that difference is in the population studied. Aim assistance is only necessary if there are, in fact, large differences in people's aiming ability that require balancing.

RQ2 *How accurately does the model characterize a player's aiming ability?* To be useful, our predictive model needs to produce accurate predictions. In analyzing it, we thus need to examine how much error occurs in the model between predictions of a person's aiming success and test data capturing actual aiming success.

RQ3 *How much data is required to train the model?* For the model to practically serve for player balancing, it needs to be possible to train the model quickly, both initially and to keep up with changes in player ability. It is therefore important to determine how many observations of aiming tasks are required to train the model.

Participants played a bespoke game (called *ChronoSwarm*), which is based entirely on aiming at moving targets. The game's difficulty is designed to be adjustable by manipulating the difficulty parameters used by the model described in Section 3.1. The game logs where and when players click, providing both test and training data for determining the accuracy of the model, and how quickly it model can be trained.

The study materials, participants, procedures, and analysis techniques are described in detail below. These apply both to this study, as well as the pilots reported in Appendix A. This study was registered with the Open Science Framework (OSF) prior to data collection[1].

4.1 Test Game: ChronoSwarm

ChronoSwarm is a simple target-shooting game in which players are tasked with clicking on a series of rectangular blocks that move across the screen at a constant rate (Figure 3). Players must also time their clicks to occur when a sweeping clock hand is between two target lines. Thus, hitting in ChronoSwarm has a spatial component (clicking on top of the block) and a temporal component (clicking at the right time). Figure 4 shows examples of spatial and temporal hits and misses. The game was designed to emphasize the challenge of spatial and temporal aiming with minimal additional gameplay elements, isolating as much as possible the effect of difficulty on aiming performance.

ChronoSwarm runs in a web browser to allow participants to play it without an installation procedure. Once loaded the game runs entirely locally, to ensure network performance is not a factor in player performance. Players use their mouse or other pointing device to attempt to hit each target as it appears. Once a player attempts to hit, feedback is shown (hit or miss); the target is removed from the display, and a new target is presented. ChronoSwarm's gameplay is shown in the attached video figure.

The game's difficulty can be varied by adjusting the height and width of the target, the speed of its movement, and the positioning of the clock hands (Figure 5).

4.2 Recruiting

72 participants were recruited from the online crowdsourcing platform Amazon Mechanical Turk [49]. The study was open to participants at least 18 years old who had completed 1,000 or more Mechanical Turk tasks with an approval rate of at least 95%.

[1]Link to registration: https://osf.io/dfsh2/?view_only=0a34b579e89e48129584736e305357d9

Participants were also required to be familiar with fast-paced games involving timed actions, to be using a Windows computer with a pointing device and a screen of at least 12" or 30cm in diagonal size. These additional requirements were to help ensure that participants could complete the digital game pointing task that forms the core of the study. More detailed factors, such as whether a participant was using a mouse or a trackpad, are reflected in the free parameters of the model (see Section 3.2 and [29]).

4.3 Participant Flow

The study was split into three sections, each completed within a web browser, with hyperlinks moving participants between the sections. The flow between these sections, and their sub-components, is illustrated in Figure 6.

After initial recruitment through Mechanical Turk, participants linked to an demographic survey hosted on Qualtrics [50]. The survey tool randomly assigned participants into spatial and temporal groups (as described in Section 4.4). After completing the demographic survey, each participant was given a custom link to the ChronoSwarm game, with conditions order-balanced between participants.

Each participant was required to play a brief ChronoSwarm tutorial. Any participant that failed to achieve a ten percent success rate in the tutorial was excluded from analysis.

Upon completion of the tutorial, the game began. Participants were presented with 540 ChronoSwarm targets (similar to those shown in Figures 3, 4, and 5), in sets of 30. In total, completing the ChronoSwarm task took approximately 20 minutes per participant.

After completing the ChronoSwarm segment and saving their data, participants were prompted to save their results to the server, after which they were linked to a brief exit questionnaire asking the difficulty of ChronoSwarm.

On submitting the exit questionnaire, players were given a completion code to copy back to Mechanical Turk.

4.4 Conditions

To train the model, as described in Section 3.3, we required observations of the participants carrying out aiming tasks at different levels of difficulty. Participants in the spatial and temporal conditions performed aiming tasks varying over two difficulty parameters, each with values representing easy, medium, and hard level of challenge. These parameters and values are summarized in Table 2.

The specific values for the spatial parameters were selected through iterative refinement in three ten-participant mini-pilots conducted before the pilots reported in Appendix A. Given the greater importance of width as a difficulty parameter over height (see Section A.1), we were able to use square spatial targets, removing the need to consider spatial width and height as separate parameters and therefore reducing the number of study conditions needed.

We chose the parameters for cue viewing time and temporal width using Human Benchmark's *Aim Trainer* [32], in which players click a series of 30 stationary targets as rapidly as possible. The program's statistics take the form of a curve, showing most participants average at least 250ms, and at most 700-800ms, to click on each target. We set the "hard" values of cue viewing time and

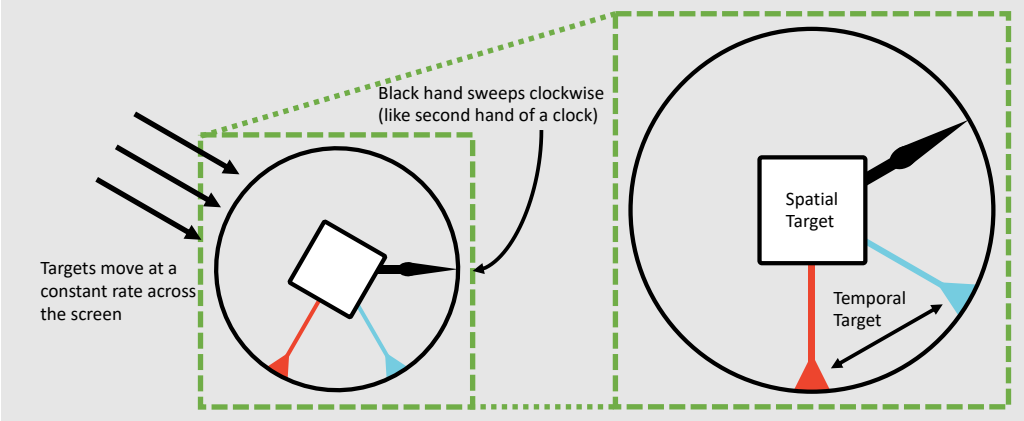
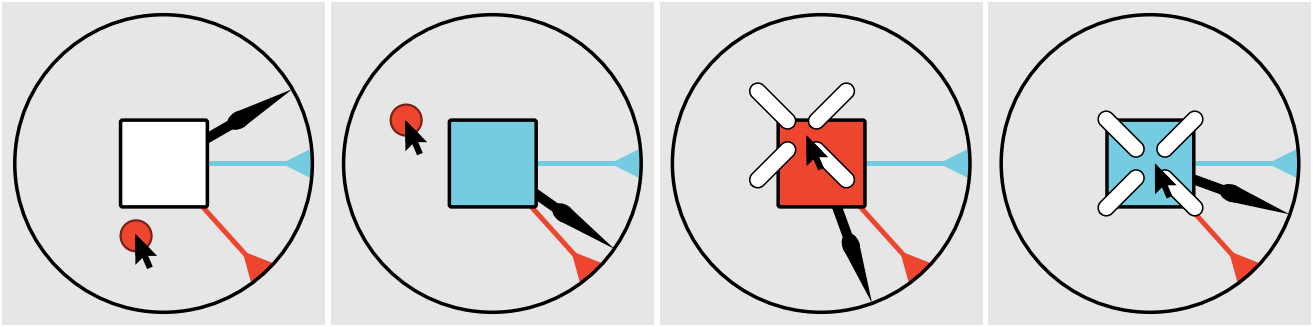


Figure 3: Gameplay in ChronoSwarm. A target moves across the screen. The player must click the target box when the sweeping black hand is between the blue and red lines. This combines both spatial aiming (hitting the box) and temporal aiming (hitting at the right time).



(a) spatial miss, temporal miss **(b) spatial miss, temporal hit** **(c) spatial hit, temporal miss** **(d) spatial hit, temporal hit**

Figure 4: Examples of hitting and missing in ChronoSwarm.

Spatial Difficulty Parameters				*Temporal Difficulty Parameters*			
	Easy	**Medium**	**Hard**		**Easy**	**Medium**	**Hard**
Spatial Width	120px	84px	48px	Temporal Width	250ms	200ms	150ms
Speed	96px/s	288px/s	480px/s	Cue Viewing Time	500ms	375ms	250ms

Table 2: Study Conditions. Each difficulty parameter was tested at easy, medium, and hard settings. Difficulty parameters were defined in Table 1.

temporal width to match the start and peak of the curve, with the other values spaced out along the curve.

To keep the number of conditions tractable, participants were split between a spatial group and a temporal group. Each group had 9 conditions, formed by fully crossing the easy/medium/hard values of both difficulty parameters: spatial width and speed for the spatial group, cue viewing time and temporal width for the temporal group. In addition, each of these condition sets were crossed with the highest and lowest values of the opposite-axis parameter that our pilot testing (Sections 3.1, A.2) showed had some significance as a predictor: cue viewing time for the spatial group and spatial width for the temporal group. In total, this resulted in 18 conditions for each group. These 18 conditions were order-balanced between participants using a balanced Latin square. This required that participants be in multiples of 18, with the figure we selected being 36 participants in each group, for a total of 72 participants.

4.5 Data Collection

We collected three sources of data: demographic survey, ChronoSwarm gameplay data, and exit questionnaire.

The primary data reported and analyzed in this paper are derived from the ChronoSwarm gameplay logs, with a demographic overview also reported in Section 5. For each target, these logs record the difficulty parameter values (see Table 2), whether it

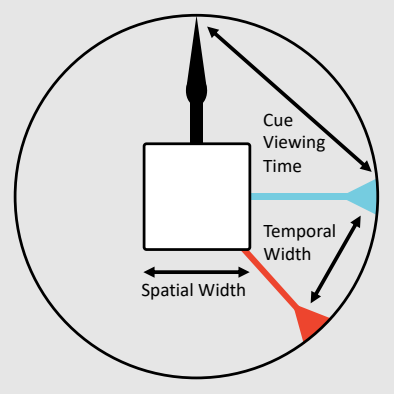


Figure 5: Three of the four parameters that are used to adjust the difficulty of spatial and temporal aiming in ChronoSwarm. Not pictured is the speed at which the target moves.

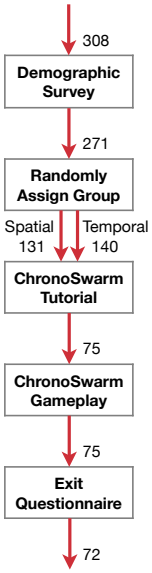


Figure 6: Participant flow through the study. Participants who entered nonsense data in the demographic questionnaire, participants who failed to complete the tutorial, and participants who did not complete all three sections were removed from the study.

was successfully clicked (both spatially and temporally), and the distance from the target's centre (i.e., the tangent, normal, and temporal endpoint values – Section 4.6). After completing all conditions for their group (spatial or temporal), participants were prompted to press a button to upload the log to the ChronoSwarm server, where it was stored as a text file.

The demographic survey collected information about age, gender, and gameplay experience. Participants were asked their level of gameplay experience in the two categories of "fast-paced shooter games" and "non-shooter action games that need both movement and timing." For each, players were asked to identify as a casual, frequent, or competitive player, and to name the game in that category with which they had the most experience. Finally, players were asked what clicking and display devices they were using to play ChronoSwarm and the diagonal size of their screen. Participants who provided nonsense information (e.g., random text for screen size) were removed from the study.

The exit questionnaire asked players to rank the difficulty of the game on a 7-point Likert scale. This was split into seven categories: overall difficulty, difficulty of the smallest and largest targets, difficulty of the fastest and slowest targets, and difficulty of clicking during the shortest and longest times.

4.6 Measures

Our outcome measures fall into three categories: the player's gameplay success, the player's aiming accuracy, and the model's accuracy. These measures are detailed below.

4.6.1 Player's Gameplay Success. Success in the game is determined by the percentage of targets that the player hit. As shown in Figure 4, a player is considered to have hit a target if they click on top of the target's rectangle (spatial hit) during the time that the sweeping hand is between the blue and red hands (temporal hit).

Differences in success between players are measured using the coefficient of variation (CoV), which is the ratio of standard deviation to mean. CoV indicates the extent of variability in data relative to the mean, with a higher CoV indicating more dispersal in the dataset [36].

We use one metric to measure gameplay success, and one to measure differences in success between players:

Success Rate: A player's success rate is calculated as the percentage of all targets hit across all trials within a given condition. Success rates are calculated separately for spatial hits – **spatial** success rate – and temporal hits – **temporal** success rate.

Coefficient of Variation: The coefficient of variation (CoV) is the ratio of standard deviation to mean. We calculated the CoV for success rate across all 36 participants in a group, using participants' average success rate. A high CoV indicates variation between the success rates of different players.

4.6.2 Player's Aiming Accuracy. Accuracy is measured in both the spatial and temporal dimensions, using the distance of a player's click from the target. A larger distance means the player is less accurate in aiming for the target than if the click were very close to the target. For a given hit attempt:

Spatial Endpoint is the location of the player's click relative to the centre of the targeted block. Spatial endpoints are measured in pixels.

Temporal Endpoint is the time at which the player clicks relative to the temporal width. Temporal endpoints are measured in milliseconds.

4.6.3 Model's Accuracy. The accuracy of the model captures how well the model predicts a player's actual performance in playing the game. This is measured both in terms of prediction of endpoint accuracy and prediction of success rate. Success rate more directly addresses the question of how well a player aims, but requires

aggregation of multiple binary hit/miss datapoints. Endpoint data provide more granularity, with even a single endpoint giving a measure of distance. We calculate both accuracy measures for the sake of comparison.

The quality of prediction is measured through the *mean absolute error* between the model's predicted player performance and actual performance on test data. Whenever "error" appears in this paper, it refers to this use of mean absolute error.

Given a collection of observations of hit attempts at a specific difficulty level and a model prediction of endpoint distribution for this difficulty level, four measures are used to gauge the accuracy of the model's predictions:

Spatial Endpoint Error is calculated as the mean of the differences between predicted and actual endpoint means, calculated separately for both the tangent (width) and normal (height) axes.

Temporal Endpoint Error is calculated as the mean of the differences between predicted and actual endpoint means.

Spatial Success Rate Error is calculated as the mean of the absolute values of differences between predicted and actual spatial success rates.

Temporal Success Rate Error is calculated as the mean of the absolute values of differences between predicted and actual temporal success rates.

4.7 Comparing Models

Endpoint and success rate error are our measures of the model's accuracy (RQ2). However, to analyze how increasing the number of training samples affects the accuracy of the model, we need to compare multiple models for each participant, each trained on a different number of samples (RQ3).

To build and train these different versions of the model, participants were given 30 targets at each difficulty condition. As shown in Figure 7, the first 20 targets for each condition represented the available training data, with the last 10 targets for each condition forming the test dataset.

For each model Model_N, the model was trained on the first N samples from each of the 18 conditions. For example, the training dataset for Model_5 was the first 5 samples from each condition – 90 samples – while Model_{20} was trained on the first 20 samples from each condition – 360 samples. We constructed Model_5 through Model_{20}, producing 16 models to compare.

5 RESULTS AND ANALYSIS

As shown in Figure 6, 308 participants were recruited and 72 successfully completed the experiment. 148 participants were rejected for having failed to complete the survey or the ChronoSwarm tutorial. This step served to divert bots or unmotivated participants from the study, as recommended by Sharpe Wessling *et al.* [55]. Another 64 were eliminated for not completing the study, and 24 for providing nonsense data in the demographic survey. Nonsense survey data included, for example, entering random text when asked to provide the size of their display. 36 participants completed each of the spatial and temporal arms of the study.

Participants ranged in age from 22 to 69, with a mean age of 37. 47 participants identified as male and 25 as female. Experience with shooter games ranged from casual (24 participants), through frequent (34 participants), and competitive (14 participants).

We structure our results around our research questions of (RQ1) how much aiming ability varies between players, (RQ2) the accuracy of the dynamic spatiotemporal model, and (RQ3) how rapidly the model can be trained. Results are presented by research question and by spatial/temporal group.

5.1 RQ1: How Much do Individuals Vary in Aiming Ability?

This research question asks whether there are indeed differences in aiming ability within the set of participants that were tested. Our measure for this question is to compare the differences in success rate between players. This comparison uses the coefficient of variation (CoV), the ratio of standard deviation to mean (Section 4.6).

CoV analysis captures how much variation there was between participants' performances in both the spatial and temporal groups. A player with higher mean success rates is exhibiting higher aiming ability than a player with low mean success rates. When comparing success rates between players, a high CoV value indicates that there was indeed variation in participants' aiming ability.

For the CoV analysis, we compute each player's success rate for each condition. For each participant, we compute the average success rate across all conditions. The CoV between participants is then calculated as the standard deviation of those averages divided by the mean.

In the spatial group, mean success rate across all conditions was 54.7%, with a standard deviation of 0.225. This corresponds to a spatial CoV of 0.411, or 41.1% of the mean.

In the temporal group, mean success rate across all conditions was 43.7%, with a standard deviation of 0.227. This corresponds to a temporal CoV of 0.520, or 52.0% of the mean.

For both the spatial group and the temporal group, CoV is over 40%. Viewed as confidence intervals, these values show that approximately one third of participants have a success rate below or above the range 32%-77% for the spatial group, and 21%-66% for the temporal group. These results confirm our intuition that there is indeed variation in the aiming abilities of both spatial and temporal participants.

5.2 RQ2: How Accurately does the Model Characterize a Player's Aiming Ability?

The goal for this question is to measure how accurately the model is able to predict player performance.

The metric for this research question is mean absolute error between each model's predictions and the actual outcomes from the test data. Error analyses were conducted separately for the spatial and temporal components of the model, and for both endpoints and success rates. Lower error indicates a more accurate prediction, with a baseline of 0 denoting perfect accuracy. In the case of success rate error, the maximum possible error is 100%.

In each case—spatial/temporal and endpoint/success rate—there are a total of 16 values, representing the different training dataset sizes (Model_5 through Model_{20}, as described in Section 4.7).

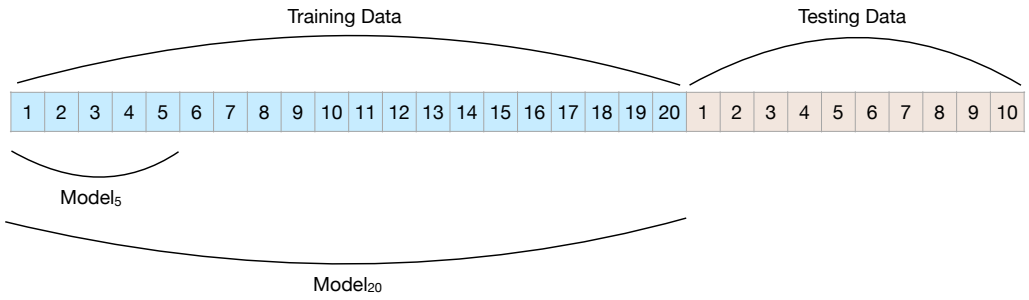


Figure 7: 20 samples were collected for each condition to train the model, while 10 samples were collected to test the model. 16 versions of the model were created using between 5 and 20 samples. For example, $Model_5$ is trained with the first 5 samples of the training data, while $Model_{20}$ is trained with all 20 training data samples.

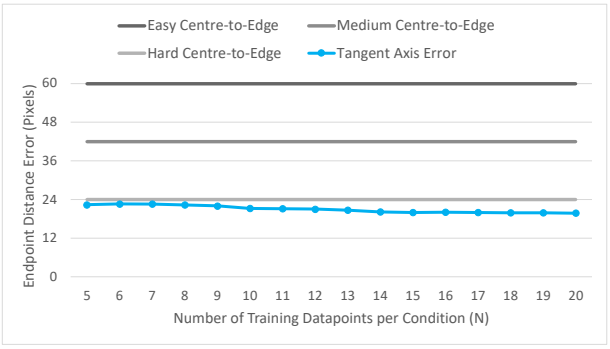


Figure 8: Mean absolute error graph for spatial endpoint predictions. Horizontal axis shows number of samples used to construct each model. Vertical axis is in pixels. From top to bottom, the horizontal lines represent the distance from the centre to the edge of the target at the "easy," "medium," and "hard" values of spatial width; these provide scale by showing the magnitude of the error compared to the sizes of the targets.

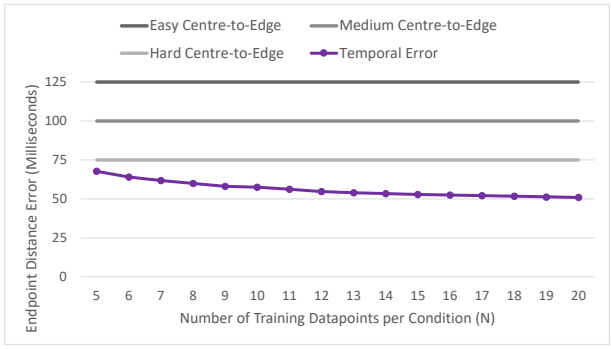


Figure 9: Mean absolute error graph for temporal endpoint predictions. Horizontal axis shows number of samples used to construct each model. Vertical axis is in milliseconds. From top to bottom, the horizontal lines represent the distance from the centre to the edge of the temporal width at the "easy," "medium," and "hard" values of temporal width; these provide scale by showing the magnitude of the error compared to the sizes of the targets.

5.2.1 Endpoint Error. As shown in Figure 8, in the spatial group, error for the tangent axis endpoint predictions ranged from a low of 20px (at N=20) to a high of 23px (at N=6). Error for the normal axis is constant at 12px. The figure provides scale by also showing the centre-to-edge size of the targets used in the study. The error values are smaller than the size of the smallest target (24px centre-to-edge) and considerably smaller than the size of the larger targets.

Figure 9 shows that in the temporal group, error for the endpoint predictions ranged from a low of 51ms (at N=20) to a high of 68ms (at N=5). These values are smaller than the centre-to-edge temporal width of the smallest target (75ms).

For both the spatial and temporal groups, endpoint error was less than the difference between the centre and edge of the smallest targets, suggesting error was small relative to the constraints of the aiming task. This is a highly encouraging result, suggesting that the endpoint predictions are accurate enough to model the aiming ability of game players.

5.2.2 Success Rate Error. In the spatial group, error for spatial success predictions ranged from 17.4% (N=7) to 18.0% (N=19), as shown in Figure 10. In the temporal group, error for temporal success predictions ranged from 14.2% (N=19) to 17.2% (N=5), as shown in Figure 10. These results suggest that the dynamic spatiotemporal model is able to predict player success, though less accurately than it predicts endpoint distribution. As we shall discuss in Section 6, this suggests that DDA algorithms should be based on endpoint predictions rather than success rate predictions.

5.3 RQ3: How Much Data is Required to Train the Model?

The results of this question address the model's utility for a dynamic assistance algorithm. In particular, it addresses whether the initial training for the model can be conducted quickly, and by extension whether the model is responsive to changes in a player's aiming ability over time. The metric for this research question is success rate error between the model's predictions and the test data, as seen in Section 5.2. Again, a lower error indicates a more accurate prediction.

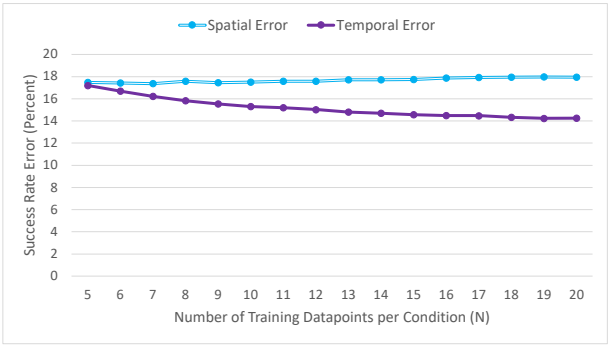


Figure 10: Mean absolute error graph for spatial and temporal success rate predictions. Horizontal axis shows number of samples used to construct each model. Vertical axis is success rate.

In the spatial group, there is no apparent improvement over time for either endpoint error (Figure 8) or success rate error (Figure 10). In the temporal group, there is a decrease over time in endpoint error (Figure 9); the error at N=5 (68ms) is approximately 33% larger than at N=20 (51ms). There is also a decrease over time in success rate error (Figure 10); the error at N=5 (17.2%) is approximately 21% larger than at N=20 (14.3%).

The analysis of error across increasing training dataset size N shows that, for both the spatial and temporal components of the model, training can be reasonably performed with as few as five datapoints per condition. In particular, endpoint error values at N=5 are already lower than half the smallest width for both spatial and temporal aiming. Temporal error drops as the number of observations increase, but at N=5 is already as low as the largely invariant spatial error.

In summary, we find that there is considerable variation in both spatial and temporal aiming ability in our participant population, that the model is successfully able to predict the aiming performance of individuals, and that the model can be trained with relatively few observations. In the next section, we discuss the deeper implications of these results.

6 DISCUSSION

In this section, we summarize the results of the study assessing our dynamic spatiotemporal model, and discuss the implications of these results—combined with the outcomes of the pilot studies reported in Appendix A—on using the model for player balancing based on aim assistance and dynamic difficulty adjustment (DDA).

RQ1 asks how much individuals differ in their aiming ability, for both spatial and temporal aiming tasks. Analyzing player success rates, we found considerable variation in aiming ability between the participants in our study. The presence of variation between participants justifies the need for aim assistance, in order to balance between players with differences in aiming ability. The demographic questionnaire reveals a large enough spread of ages and gaming backgrounds that we can be confident our online participants are reasonably reflective of the general population of computer-users.

RQ2 asks how accurately our dynamic spatiotemporal model is able to predict player aiming performance. Error analysis between model predictions and actual test observations showed that the model accurately predicted the distribution of where and when players tended to click. It also showed the model predicted how successful players will be at clicking targets, though less accurately than the direct endpoint distribution predictions.

The results indicate that the model is able to perform its core function: individualized prediction of accuracy on aiming tasks defined by discrete difficulty parameters, such as target speed, target size, and time the target is available.

RQ3 asks how rapidly the dynamic spatiotemporal model can be trained, in terms of number of observations. Error analysis between models with different sizes of training data showed that 45 datapoints – 5 per difficulty condition – were sufficient for the spatial component to converge on its accuracy peak. The same analysis for the temporal component showed that 45 datapoints were enough for the model to near its accuracy peak, but that increasing the number of samples could increase accuracy of predictions.

The accuracy suggests that 90 observations is sufficient for training the model. While the temporal component shows modest improvement in accuracy as more observations are collected, 45 are already enough to reach the same accuracy as the spatial component. 90 observations are few enough to be included, for example, in a tutorial, allowing initial training to be conducted even before the start of normal gameplay.

We begin our discussion with the matter of extensibility: how well do the findings from our crowdsourced ChronoSwarm study transfer to other games? We also describe the process of using the model in such a game as part of a dynamic balancing algorithm.

6.1 Extensibility to Other Games

In our study, participants played the bespoke *ChronoSwarm* game. ChronoSwarm is designed to focus on aiming tasks, while removing confounding gameplay elements such as strategy, luck, or quality of AI opponent.

The difficulty parameters used in ChronoSwarm are near universal in games involving aiming, including multiple popular game genres. A target could be another player in a multiplayer shooter game, a moving platform in a platformer game, or a narrow corridor in a driving game. These targets can all be characterized using our difficulty parameters, using a size and a speed (which might be zero in some cases).

In some games, the player's perspective may influence the measure of a difficulty parameter. In a game with depth, some computation may be needed to get width and speed from the player's perspective.

In a shooter game where the objective is to hit other players, there may be no explicit temporal width defining when an opponent can or cannot be hit. If an opponent is able to hide behind cover, it may nonetheless have an *effective* temporal width. If the opponent walks out from behind a wall, remains in the player's range for half a second, and then moves back behind the wall, then as a target that opponent had a temporal width of 500ms.

Cue viewing time is relevant in games where the player uses a device like Splatoon 3's sensor [16] that shows opponents behind walls. If the player can see an opponent before they move into range, then that corresponds to an *effective* cue viewing time. Some games may have no such mechanic to provide advance warning; in this case, the cue viewing time can be considered zero.

In cases where a difficulty parameter is not used in a game, the parameter can be omitted from the analysis, increasing the speed of training. This can be seen in our use of square targets in ChronoSwarm, for example, which removed the need to consider height. This optimization applies in many games, as targets typically maintain the same ratio of height and width; for example, a person gets bigger as they get closer, but still has the same ratio of width to height. If a game includes targets where width and height vary truly independently, a height parameter could be easily added back into the model, at the cost of increased training time.

6.2 Performing DDA with the Dynamic Spatiotemporal Model

While the model was successful in predicting player performance, the mean absolute error analysis suggested that the model is more accurate in predicting endpoint distributions than in predicting success rates. To use endpoint predictions in a dynamic balancing algorithm, a game designer should determine a goal accuracy for players, measured by their predicted endpoint distribution. For example, the goal may be for at least 50% of the player's clicks to fall within 30 pixels of the target.

In order to perform the difficulty adjustment itself, the dynamic balancing algorithm should follow the steps laid out in Section 3.3: (1) **observe** the player aiming at targets of varying difficulty, (2) use those observations to **model aiming ability** of that player, and (3) **adjust aiming tasks** until the model predicts the player's accuracy will meet the goal (see Figure 1).

Some games may have varied targets that appear often enough to quickly gather the needed observations during normal play. In general, however, we recommend that the initial training be conducted during an opening tutorial. While the tutorial is explaining how to play the game, it can select targets with varying difficulty for the player to practice against. Once this initial training is complete, the model can continue observing player performance for ongoing training, including testing independent variation in spatial width and height if appropriate for the game.

The exact nature of the difficulty adjustment will depend on what game is being balanced, and whether a player is having difficulty with spatial aiming, temporal aiming, or both. If a player's spatial aiming performance is below the goal, then the spatial constraints can be eased by making the targets wider or slower. Likewise, if a player's temporal aiming performance is too low, then the game can adjust by lengthening the warning before a target appears (cue viewing time) or increasing the time that it remains targetable (temporal width).

In some cases, it may be difficult to directly modify the difficulty parameters of targets. For example, in a shooter game where the goal is to aim at other players, slowing down opponents to make them easier to hit will disrupt those opponents' ability to play the game. In the same game, the temporal width may depend on when opponents leave and re-enter cover, which is likewise difficult to adjust without restricting those opponents.

In cases like these, aim assistance can modify the difficulty parameters indirectly. An area cursor [4, 5, 35, 57–59] can make a target effectively wider for the sake of clicking on it without actually changing the target's size, or sticky targets [1, 4, 5, 35, 44, 52, 59] can help a player's cursor track a fast target without actually slowing it down. In the case of temporal width dictated by opponents moving behind cover, players can be given the temporary ability to see and shoot targets through cover: if a player can still hit the target 100ms after it enters cover, then the player has effectively been granted a 100ms increase to temporal window.

6.3 Application Beyond Gaming

Aiming is critical in many games, including shooter games like Fortnite, platformer games like the Super Mario series, and racing games like Mario Kart; therefore, this paper has the potential for high impact in game development. Our model is primarily intended to make the play of digital games more fun and equitable by allowing persons with different abilities to play together. Aim assistance occurs in other domains as well, such as in power wheelchairs [17] which are typically controlled by a joystick, and control of teleoperated robots [14] and unmanned aerial vehicles [22]. Such applications often provide assistance in targeting, and modeling the user of the system may help in providing the right level of assistance.

6.4 Limitations

One possible limitation of our study is our use of the Amazon Mechanical Turk [49] crowdsourcing platform. The online nature of the study limited our ability to observe participants and ensure they were giving a sincere effort. As shown in Figure 6, 236 of 308 participants were excluded due to failing to complete the study or providing nonsense answers to written questions. This indicates that screening effectively removed insincere participants, but that surprisingly, sincere participants were the minority.

The study has no control group for comparison of model accuracy. This is not possible, as there are no existing models that could act as comparators. Instead, we provide context by comparing the model's error to the size of the targets presented in the game.

The study was also based on a single game, designed to minimize the effect of any gameplay element other than spatial and temporal aiming, for the sake of isolating the effect of aiming difficulty. We have argued above that this game's core elements are also core elements of a wide range of game genres. However, further testing with other types of games would be suitable future work, including games demanding a wider range of abilities from players.

In these discussions, we have laid the theoretical foundation for how to use our dynamic spatiotemporal model. We described both how, in a deployment in an actual game, an implementation of our model would be able to parameterize the difficulty of game targets, as well as how to use targets of varying difficulty to execute a DDA algorithm. Future work in this space would involve putting this foundation into practice in a game, and measuring how effectively the model functions in a real deployment compared to the testbed of ChronoSwarm.

7 CONCLUSION

To use aim assistance for player balancing, a measure of each player's ability is required to personalize the amount of assistance provided. Dynamic measurement based on the players' ability to hit targets in a game allows the model to be trained during play. In this paper, we present and assess a novel model for characterizing spatial and temporal aiming ability of individual players, drawing from the work of Huang *et al.* [29, 30] and Lee *et al.* [38]. Our dynamic spatiotemporal model is trained using data that can be collected during play, and can be trained rapidly enough to be used in dynamic balancing aim assistance algorithms.

The study we performed to assess the model confirmed that both spatial and temporal aiming ability vary between players. The model converges with a small number of samples per setting of target difficulty, and the predictions of where and when players would click are accurate to distances smaller than the size of the smallest targets.

REFERENCES

[1] David Ahlström, Martin Hitz, and Gerhard Leitner. 2006. 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*. 58–67.

[2] Gustavo Andrade, Geber Ramalho, Hugo Santana, and Vincent Corruble. 2005. Extending Reinforcement Learning to Provide Dynamic Game Balancing.

[3] Alexander Baldwin, Daniel Johnson, and Peta Wyeth. 2016. Crowd-pleaser: Player perspectives of multiplayer dynamic difficulty adjustment in video games. In *Proceedings of the 2016 Annual Symposium on Computer-Human Interaction in Play*. 326–337.

[4] Scott Bateman, Regan L Mandryk, Carl Gutwin, and Robert Xiao. 2013. Analysis and comparison of target assistance techniques for relative ray-cast pointing. *International Journal of Human-Computer Studies* 71, 5 (2013), 511–532.

[5] Scott Bateman, Regan L Mandryk, Tadeusz Stach, and Carl Gutwin. 2011. Target assistance for subtly balancing competitive play. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2355–2364.

[6] Xiaojun Bi, Yang Li, and Shumin Zhai. 2013. FFitts law: modeling finger touch with fitts' law. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 1363–1372.

[7] Xiaojun Bi and Shumin Zhai. 2013. Bayesian touch: a statistical criterion of target selection with finger touch. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*. 51–60.

[8] Bijan Stephen. Accessed Sep. 14, 2022 [Online]. How do you make Fortnite fair? https://www.theverge.com/2020/6/4/21280358/fortnite-aim-assist-sypherpk-tfue-shooters-epic-games-fairness

[9] David Buckley, Ke Chen, and Joshua Knowles. 2013. Predicting skill from gameplay input to a first-person shooter. In *2013 IEEE Conference on Computational Inteligence in Games (CIG)*. IEEE, 1–8.

[10] Jared E Cechanowicz, Carl Gutwin, Scott Bateman, Regan Mandryk, and Ian Stavness. 2014. Improving player balancing in racing games. In *Proceedings of the first ACM SIGCHI annual symposium on Computer-human interaction in play*. 47–56.

[11] Guillaume Chanel, Cyril Rebetez, Mireille Bétrancourt, and Thierry Pun. 2011. Emotion Assessment From Physiological Signals for Adaptation of Game Difficulty. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 41, 6 (2011), 1052–1063. https://doi.org/10.1109/TSMCA.2011.2116000

[12] ERFW Crossman. 1957. The speed and accuracy of simple hand movements. *The nature and acquisition of industrial skills* (1957).

[13] Edward Robert FW Crossman and PJ Goodeve. 1983. Feedback control of hand-movement and Fitts' law. *The Quarterly Journal of Experimental Psychology Section A* 35, 2 (1983), 251–278.

[14] Anca D Dragan and Siddhartha S Srinivasa. 2012. Assistive teleoperation for manipulation tasks. In *Proceedings of the seventh annual ACM/IEEE international conference on human-robot interaction*. 123–124.

[15] Entertainment Software Association. Accessed Dec. 9, 2022 [Online]. 2022 Essential Facts. https://www.theesa.com/resource/2022-essential-facts-about-the-video-game-industry/

[16] Nintendo EPD. 2022. Splatoon 3. [Nintendo Switch].

[17] Chinemelu Ezeh, Pete Trautman, Louise Devigne, Valentin Bureau, Marie Babel, and Tom Carlson. 2017. Probabilistic vs linear blending approaches to shared control for wheelchair driving. In *2017 International Conference on Rehabilitation Robotics (ICORR)*. IEEE, 835–840.

[18] Paul M Fitts. 1954. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of experimental psychology* 47, 6 (1954), 381.

[19] PAUL M Fitts, WILLIAM F Bennett, and HARRY P Bahrick. 1956. Application of autocorrelation and cross-correlation analysis to the study of tracking behavior. In *Symposium on Air Force human engineering, personnel, and training research. Baltimore: Air Research and Development Command*. 56–8.

[20] Paul M Fitts and Barbara K Radford. 1966. Information capacity of discrete motor responses under different cognitive sets. *Journal of Experimental psychology* 71, 4 (1966), 475.

[21] Giant Bomb. Accessed Sep. 14, 2022 [Online]. The Recurring Types of Auto-Aiming. https://www.giantbomb.com/auto-aim/3015-145/

[22] Marco Antonio Gutierrez, Luis Fernando D'Haro, and Rafael Banchs. 2016. A multimodal control architecture for autonomous unmanned aerial vehicles. In *Proceedings of the Fourth International Conference on Human Agent Interaction*. 107–110.

[23] Carl Gutwin, Rodrigo Vicencio-Moreira, and Regan L Mandryk. 2016. Does helping hurt? Aiming assistance and skill development in a first-person shooter game. In *Proceedings of the 2016 Annual Symposium on Computer-Human Interaction in Play*. 338–349.

[24] Halopedia. Accessed Sep. 14, 2022 [Online]. Aim assist. https://www.halopedia.org/Aim_assist

[25] Hannes Burger. Accessed Sep. 14, 2022 [Online]. Battlefield 4 and the Aim Assist. http://zombiegamer.co.za/bf4-aim-assist

[26] Khalad Hasan, Tovi Grossman, and Pourang Irani. 2011. Comet and target ghost: techniques for selecting moving targets. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 839–848.

[27] Hamilton A Hernandez, Zi Ye, TC Nicholas Graham, Darcy Fehlings, and Lauren Switzer. 2013. Designing action-based exergames for children with cerebral palsy. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 1261–1270.

[28] Jin Huang and Byungjoo Lee. 2019. Modeling Error Rates in Spatiotemporal Moving Target Selection. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–6.

[29] Jin Huang, Feng Tian, Xiangmin Fan, Xiaolong Zhang, and Shumin Zhai. 2018. Understanding the uncertainty in 1D unidirectional moving target selection. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–12.

[30] Jin Huang, Feng Tian, Nianlong Li, and Xiangmin Fan. 2019. Modeling the uncertainty in 2D moving target selection. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. 1031–1043.

[31] Jeff Huang, Thomas Zimmermann, Nachiappan Nagapan, Charles Harrison, and Bruce C Phillips. 2013. Mastering the art of war: how patterns of gameplay influence skill in Halo. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 695–704.

[32] Human Benchmark. Accessed Sep. 14, 2022 [Online]. Aim Trainer. https://humanbenchmark.com/tests/aim

[33] Robin Hunicke. 2005. The case for dynamic difficulty adjustment in games. In *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology*. 429–433.

[34] James O'Connor. Accessed Sep. 14, 2022 [Online]. Gears 5 Operation 3 Balance Changes Are Coming For The Lancer And Gnasher. https://www.gamespot.com/articles/gears-5-operation-3-balance-changes-are-coming-for/1100-6476166/

[35] Jawad Jandali Refai. 2018. *Core task assistance in video games*. Ph. D. Dissertation. University of New Brunswick.

[36] Joint Research Centre of the European Commission. Accessed Sep. 14, 2022 [Online]. Coefficient of Variation. https://datacollection.jrc.ec.europa.eu/wordef/coefficient-of-variation

[37] Josh Tolentino. Accessed Sep. 14, 2022 [Online]. Good Idea, Bad Idea: Dynamic Difficulty Adjustment. https://www.destructoid.com/good-idea-bad-idea-dynamic-difficulty-adjustment/

[38] Byungjoo Lee, Sunjun Kim, Antti Oulasvirta, Jong-In Lee, and Eunji Park. 2018. Moving target selection: A cue integration model. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–12.

[39] Byungjoo Lee and Antti Oulasvirta. 2016. Modelling error rates in temporal pointing. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. 1857–1868.

[40] Changchun Liu, Pramila Agrawal, Nilanjan Sarkar, and Shuo Chen. 2009. Dynamic Difficulty Adjustment in Computer Games Through Real-Time Anxiety-Based Affective Feedback. *International Journal of Human–Computer Interaction* 25, 6 (2009), 506–529. https://doi.org/10.1080/10447310902963944

[41] I Scott MacKenzie. 1992. Fitts' law as a research and design tool in human-computer interaction. *Human-computer interaction* 7, 1 (1992), 91–139.

[42] I Scott MacKenzie. 1993. Fitts' law as a performance model in human-computer interaction. (1993).

[43] I Scott MacKenzie. 2018. Fitts' law. *Handbook of human-computer interaction* 1 (2018), 349–370.

[44] Regan L Mandryk and Carl Gutwin. 2008. Perceptibility and utility of sticky targets. In *Proceedings of graphics interface 2008*. Citeseer, 65–72.
[45] Merriam-Webster. Accessed Sep. 14, 2022 [Online]. aim. https://www.merriam-webster.com/dictionary/aim
[46] Atsuo Murata. 1999. Extending effective target width in Fitts' law to a two-dimensional pointing task. *International journal of human-computer interaction* 11, 2 (1999), 137–152.
[47] Daniel Natapov, Steven J Castellucci, and I Scott MacKenzie. 2009. ISO 9241-9 evaluation of video game controllers. In *Proceedings of Graphics Interface 2009*. Citeseer, 223–230.
[48] Lian Ting Ni, Darcy Fehlings, and Elaine Biddiss. 2014. Design and evaluation of virtual reality–based therapy games with dual focus on therapeutic relevance and user experience for children with cerebral palsy. *GAMES FOR HEALTH: Research, Development, and Clinical Applications* 3, 3 (2014), 162–171.
[49] Gabriele Paolacci, Jesse Chandler, and Panagiotis G Ipeirotis. 2010. Running experiments on amazon mechanical turk. *Judgment and Decision making* 5, 5 (2010), 411–419.
[50] Qualtrics. 2022. Qualtrics. https://www.qualtrics.com/
[51] Bob's Space Racers. 1977. Whac-a-Mole. [arcade game].
[52] Jawad Jandali Refai, Scott Bateman, and Michael W Fleming. 2020. External Assistance Techniques That Target Core Game Tasks for Balancing Game Difficulty. *Frontiers in Computer Science* 2 (2020), 17.
[53] Richard A Schmidt, Howard Zelaznik, Brian Hawkins, James S Frank, and John T Quinn Jr. 1979. Motor-output variability: a theory for the accuracy of rapid motor acts. *Psychological review* 86, 5 (1979), 415.
[54] Adrian L Jessup Schneider, Kathy Keiver, Alison Pritchard Orr, James N Reynolds, Neven Golubovich, and TC Nicholas Graham. 2020. Toward the Design of Enjoyable Games for Children with Fetal Alcohol Spectrum Disorder. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–13.
[55] Kathryn Sharpe Wessling, Joel Huber, and Oded Netzer. 2017. MTurk character misrepresentation: Assessment and solutions. *Journal of Consumer Research* 44, 1 (2017), 211–230.
[56] Nicholas Shim, Ronald Baecker, Jeremy Birnholtz, and Karyn Moffatt. 2010. Tabletalk poker: an online social gaming environment for seniors. In *Proceedings of the International Academic Conference on the Future of Game Design and Technology*. 98–104.
[57] Rodrigo Vicencio-Moreira, Regan L Mandryk, and Carl Gutwin. 2014. Balancing multiplayer first-person shooter games using aiming assistance. In *2014 IEEE Games Media Entertainment*. IEEE, 1–8.
[58] Rodrigo Vicencio-Moreira, Regan L Mandryk, and Carl Gutwin. 2015. Now you can compete with anyone: Balancing players of different skill levels in a first-person shooter game. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. 2255–2264.
[59] Rodrigo Vicencio-Moreira, Regan L Mandryk, Carl Gutwin, and Scott Bateman. 2014. The effectiveness (or lack thereof) of aim-assist techniques in first-person shooter games. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 937–946.
[60] Alan Traviss Welford. 1968. Fundamentals of skill. (1968).
[61] Daniel Wheat, Martin Masek, Chiou Peng Lam, and Philip Hingston. 2015. Dynamic Difficulty Adjustment in 2D Platformers through Agent-Based Procedural Level Generation. In *2015 IEEE International Conference on Systems, Man, and Cybernetics*. 2778–2785. https://doi.org/10.1109/SMC.2015.485
[62] Jacob O Wobbrock, Edward Cutrell, Susumu Harada, and I Scott MacKenzie. 2008. An error model for pointing based on Fitts' law. In *Proceedings of the SIGCHI conference on human factors in computing systems*. 1613–1622.
[63] Robert Sessions Woodworth. 1899. Accuracy of voluntary movement. *The Psychological Review: Monograph Supplements* 3, 3 (1899), i.
[64] Charles E Wright and David E Meyer. 1983. Conditions for a linear speed-accuracy trade-off in aimed movements. *The Quarterly Journal of Experimental Psychology* 35, 2 (1983), 279–296.
[65] Bei Yuan, Eelke Folmer, and Frederick C. Harris. 2010. Game accessibility: a survey. *Universal Access in the Information Society* 10, 1 (June 2010), 81–100. https://doi.org/10.1007/s10209-010-0189-5
[66] Howard N Zelaznik, Susan Mone, George P McCabe, and Christopher Thaman. 1988. Role of temporal and spatial precision in determining the nature of the speed-accuracy trade-off in aimed-hand movements. *Journal of Experimental Psychology: Human Perception and Performance* 14, 2 (1988), 221.
[67] Shumin Zhai, Jing Kong, and Xiangshi Ren. 2004. Speed–accuracy tradeoff in Fitts' law tasks—on the equivalency of actual and nominal pointing precision. *International journal of human-computer studies* 61, 6 (2004), 823–856.
[68] Mohammad Zohaib. 2018. Dynamic difficulty adjustment (DDA) in computer games: A review. *Advances in Human-Computer Interaction* 2018 (2018).

A APPENDIX

This appendix describes the pilots performed before the study reported in Section 4. The primary purpose of these pilots was to evaluate the six candidate difficulty parameters for our spatiotemporal model: tangent width, normal height, speed, cue viewing time, temporal width, and period of input repetition (see Section 2.3). Our goal was to determine which of these parameters had the greatest effect on the difficulty of aiming at a target. By selecting the most important input parameters for the model, the model can be trained more rapidly with fewer data points than would be the case using all parameters.

The effect of each parameter was measured using linear multiple regression, using the parameters as predictors of success rate. If a parameter has a high correlation, then changing the value of that parameter has a large effect on how successfully players are able to aim; therefore, that parameter has a large impact on the difficulty of aiming at a target. Likewise, a parameter that is a low or non-significant predictor of success is a parameter that has little to no effect on aiming difficulty.

Pilot 1 assessed how well the spatial parameters—tangent width, normal height, speed—predict spatial aiming success, and how well the temporal parameters—temporal width, cue viewing time, period of input repetition—predict temporal aiming success. Pilot 2 assessed which *spatial* parameters are significant predictors of *temporal* success, and vice versa.

Except where explicitly noted, the ChronoSwarm pilots followed the same experimental design as used in the main study. For full details, see Section 4.

A.1 Pilot 1

In Pilot 1, each of the six parameters had three different values representing easy, medium, and hard difficulty with values similar to those reported in Table 2. Normal height used the same values as tangent width, and period of input repetition used the values 1500/2000/2500ms.

Participants were split between spatial and temporal groups. The conditions for each group used the easy/medium/hard values fully crossed for each of its respective parameters: tangent width, normal height, and speed for the spatial group; cue viewing time, temporal width, and period of input repetition for the temporal group. This gave a total of 27 spatial conditions and 27 temporal conditions. These conditions were order-balanced between participants using a balanced Latin square, requiring 54 spatial and 54 temporal participants for a total of 108 participants. Each participants was presented with a total of 675 ChronoSwarm targets in sets of 25, requiring approximately 25 minutes including the ChronoSwarm tutorial.

The results of the multiple regression analysis for Pilot 1 are found in the upper part of Table 3. In summary, tangent width was the strongest predictor for spatial success rate with a standardized coefficient of $\beta = 0.737$, followed by speed at $\beta = -0.487$, with the weakest predictor being normal height at $\beta = 0.373$. Cue viewing time was the strongest predictor of temporal success rate with a standardized coefficient of $\beta = 0.783$, followed by temporal width at $\beta = 0.544$, with period of input repetition having no significant predictive effect.

Pilot 1 Correlations	**Spatial Success** $R^2 = 0.919, p < .001$	**Temporal Success** $R^2 = 0.922, p < .001$
Parameter	*Standardized Coefficient*	*Standardized Coefficient*
Tangent Width	$\beta = 0.737, p < .001$	—
Speed	$\beta = -0.487, p < .001$	—
Normal Height	$\beta = 0.373, p < .001$	—
Cue Viewing Time	—	$\beta = 0.783, p < .001$
Temporal Width	—	$\beta = 0.544, p < .001$
Period of Input Repetition	—	$\beta = 0.114, p = .064$
Pilot 2 Correlations	**Spatial Success** $R^2 = 0.954, p < .001$	**Temporal Success** $R^2 = 0.980, p < .001$
Parameter	*Standardized Coefficient*	*Standardized Coefficient*
Tangent Width	$\beta = 0.933, p < .001$	$\beta = 0.194, p < .001$
Speed	$\beta = -0.509, p < .001$	$\beta = 0.020, p = .624$
Cue Viewing Time	$\beta = 0.331, p < .001$	$\beta = 0.780, p < .001$
Temporal Width	$\beta = 0.046, p = .520$	$\beta = 0.474, p < .001$

Table 3: Pilots 1 and 2 multiple correlation results for spatial and temporal success rate.

A.2 Pilot 2

In Pilot 2, the two least-predictive difficulty parameters—normal height and period of input repetition—were not used. Fully crossing the easy/medium/hard values of tangent width and speed produced nine spatial conditions; likewise, crossing easy/medium/hard values of cue viewing time and temporal width produced nine temporal conditions.

Unlike Pilot 1, participants in Pilot 2 were not split into spatial and temporal groups. Instead. each participant performed all nine spatial and all nine temporal conditions, for a total of 18 conditions order-balanced using a balanced Latin square. 36 participants were presented with a total of 270 ChronoSwarm targets in sets of 15, requiring approximately 11 minutes including the ChronoSwarm tutorial.

The results of the multiple regression analysis for Pilot 2 are found in the lower part of Table 3. In summary, cue viewing time was a significant predictor of spatial success rate with a standardized coefficient of $\beta = 0.331$, while temporal width did not predict spatial success. Tangent width was a significant predictor for temporal success rate with a standardize coefficient of $\beta = 0.194$, while speed did not predict temporal success.

The results of these pilots had two critical implications for designing our spatiotemporal aiming model and the study reported in Section 4. In Pilot 1, we determined which parameters had the least effect on the difficulty of a target: non-significant period of input repetition is omitted from the model and therefore the study; normal height is included in our model but has a much smaller effect than its counterpart tangent width (both measures of target size), allowing initial model training to be sped up by using targets with non-independent width and height variation (e.g. squares or geometrically similar rectangles). In Pilot 2, we confirmed that spatial and temporal difficulty can be separately considered, given that spatial difficulty parameters had low or no predictive effect for temporal difficulty, and vice versa.