Comparing Game-Level and Encounter-Level Rating Systems for Ranking Laser Tag Players

Collin Carroll

December 2025

1 Abstract

Accurate player ranking is important for fair matchmaking in team-based games, yet traditional rating systems rely heavily on game-level outcomes and often ignore individual performance just because it's difficult to measure reliably. This study compares a standard game-level TrueSkill system with an encounter-level approach that uses in-game interactions such as hits and missiles.

Using data from 410 ranked Space Marines 5 laser tag matches, both systems are evaluated using win–loss predictive accuracy and mean squared margin error (MSME). The encounter-level system achieves slightly higher predictive accuracy and substantially lower MSME, showing improved calibration and faster convergence of player ratings. Although encounter-level ratings require additional computational power and tuning, they provide a better measure of player skill for matchmaking.

2 Introduction

When playing games, it's only fun when it feels fair, but it's common for random or estimated teams to lead to matches that feel one-sided. Unfair games ruin the fun for everybody. To solve this problem, players can be ranked against one another, allowing teams to be evenly matched based on skill level. Skill level, or "rating", reflects both a player's contribution to team success and their individual mechanical performance. Using the TrueSkill system, it's possible to rate players depending on the outcome of a game, comparing two teams and updating ratings based on the predicted outcome and the actual result. This game-level system raises additional questions, for example, what happens when a player consistently performs well despite their team losing? The system may be drastically underrating this player because their rating is based only upon the outcome of their games, not their individual performance. This is where encounter-level ratings come into play; using a combination of game-level ratings and encounter-level ratings gives a clearer picture of the player's overall performance, which helps matchmaking systems do a better job at creating fair

games. This study will explore the extent to which adding various levels and forms of an encounter-level rating system improves match outcome predictions, potential drawbacks of the system, and implementation using real game data.

3 Background

Laser tag games in this study were played on the Laserforce system on the Space Marines 5 (SM5) game mode. SM5 is a 6v6 competitive mode where players use teamwork and individual skill to attempt to win by eliminating the other team or by scoring more points by the end of the 15-minute game. Each player on a team plays a specific role, each of which specializes in different fields such as attacking, defending, or resupplying the team with lives and ammo. Since different roles contribute in different ways to the overall success of the team, it's difficult to compare performance between players. One reason to rank players (other than for bragging rights) is to evenly divide teams through a matchmaking system. With more accurate ratings, matchmaking systems can do a better job at making fair and fun games. Chess is a good example of a game that uses a rating system. In chess, the Elo ratings system is used because it is a simple formula for ratings and predictions for 1v1 games. Since SM5 is a 6v6 game, it's not possible or effective to use the Elo rating system. TrueSkill solves this problem by supporting any number of players and teams. In this study, we'll be using the OpenSkill library (v5.1.0) [Jos25a], a fast, open-source rewrite of TrueSkill in Python 3 with the PlackettLuce model. To attempt to fix TrueSkill's problem of only using game-level data, we'll combine it with encounter-level data and compare how effective these methods are at predicting future games.

4 Data

This study uses data collected from 410 Laserforce matches played between June 2023 and November 2025. All games were played using the Space Marines 5 (SM5) mode on the Laserforce system. The dataset was collected from my local laser tag arena, with our local players. The data only includes "ranked" matches. A match is considered "ranked" if it has between 5-7 players on each team, with an equal number of players (5v5, 6v6, 7v7). Games with an unequal number of players have a significant effect on the accuracy of the rating system, so we decided to exclude them for now, since adjusting for uneven player counts is difficult and varies depending on the game.

Each match log records every in-game event generated during play. These events include every shot or missile fired, eliminations, resupplies, and other game-specific interactions like getting bases to score an extra 1001 points. All events are timestamped and contain unique player identifiers, allowing the game to be reconstructed from its events. Raw game data is uploaded in the tdf format, a proprietary format used by the Laserforce system to record game data.

On the server, these files were parsed and interpreted into Python. This data was then fed into Tortoise-ORM [tor25], storing it in a more efficient and readable format in the database, which can be used to run analysis when calculating ratings.

5 Methods

5.1 Game-Level Ratings

Game-level ratings are TrueSkill was intended to be used, updating ratings based on what teams players were on and the outcome of the game. It's important to note that in this system, rating changes do not depend on how close the score of the game was, it simply increases the ratings for the players on the winning team and decreases the ratings for the players on the losing team. After a game is rated, the amount that this rating is changed for each player depends on their rating, the rating difference between teams, and the outcome. The parameters for game-level ratings are outlined below.

μ	$= 25$	The starting value of a player's skill level. Represents the mean of a normal distribution.
σ	$= \frac{25}{3}$	The starting value of the variation of a player's skill. Represents the confidence in a player's skill, where a lower sigma value is more confidence.
β	$= \frac{25}{6}$	Hyperparameter that determines the level of uncertainty or variability present in the prior distribution of ratings. [Jos25b]
κ	$= 0.0001$	Arbitrary small positive real number that is used to prevent the variance of the posterior distribution from becoming too small or negative. It can also be thought of as a regularization parameter. [Jos25b]
τ	$= \frac{25}{275}$	The value that is added to sigma after each game to prevent it from getting too low. The larger this value is, the more volatile ratings are. This value was increased from OpenSkill's default to account for the large amount of game data, allowing player skill levels to change even after many games.

Encounter Type	$\Delta\mu$ Weight	$\Delta\sigma$ Weight
Hit (enemy player)	0	0
Hit (enemy medic)	0	0
Missile hit (enemy player)	0	0
Missile hit (enemy medic)	0	0

Table 1: Skill (μ) and uncertainty (σ) update weights for different encounter types in SM5 in the game-level system.

- Benefits

  - Low computational cost
  - Less data required
  - Simple to implement

- Drawbacks
  - Doesn't account for player performance, which could lead to inaccuracy
  - Ratings converge to true skill slowly

5.2 Encounter-Level Ratings

Encounter-level rating systems use several game-level ratings during games by treating each encounter as a separate game. For example, if the commander on the green team zaps the medic on the red team, the system will treat this as a game between the two players where the green commander won.

This allows more in-depth information about a player's skill, and since there's hundreds of encounters per game, this allows ratings to converge much faster than with game-level encounters. In this study, we'll also be testing whether these ratings also help predict future games to measure accuracy. Encounters currently include zaps and missiles against an opposing player (friendly fire is not considered). It's also possible to create more complicated encounters with this system, such as rewarding resupplies for hitting "doubles", meaning that both the medic and the ammo zapped a friendly player at the same time, giving them both lives and shots. The ability to hit "doubles" consistently is a skill for resupplies and benefits the team, but it's difficult to quantify encounters like these and ensure ratings given will help the system be more accurate. With encounter-level ratings, we need to be careful that we don't give too much weight to each encounter. Since we're pretending that each interaction between players is its own game, OpenSkill adjusts the skill level for both players considerably. To solve this issue we multiply the change in μ and σ by a value given by the parameter for the specific encounter. These values allow encounters to be weighted differently compared to each other, as well as compared to when the game is rated overall using the game-level system (which has a weight of `1`). The parameters for encounter-level ratings were chosen with the goal of increasing prediction accuracy and based on how important each type of encounter is. The parameters are outlined below.

μ	$= 25$	The starting value of a player's skill level. Represents the mean of a normal distribution.
σ	$= \frac{25}{3}$	The starting value of the variation of a player's skill. Represents the confidence in a player's skill, where a lower sigma value is more confidence.
β	$= \frac{25}{6}$	Hyperparameter that determines the level of uncertainty or variability present in the prior distribution of ratings. [Jos25b]
κ	$= 0.0001$	Arbitrary small positive real number that is used to prevent the variance of the posterior distribution from becoming too small or negative. It can also be thought of as a regularization parameter. [Jos25b]
τ	$= \frac{25}{275}$	The value that is added to sigma after each game to prevent it from getting too low. The larger this value is, the more volatile ratings are. This value was increased from OpenSkill's default to account for the large amount of game data, allowing player skill levels to change even after many games.

Encounter Type	$\Delta\mu$ Weight	$\Delta\sigma$ Weight
Hit (enemy player)	0.01	0.01
Hit (enemy medic)	0.02	0.01
Missile hit (enemy player)	0.025	0.01
Missile hit (enemy medic)	0.05	0.01

Table 2: Skill (μ) and uncertainty (σ) update weights for different encounter types in SM5 in the encounter-level system.

- Benefits
  - Ratings converge to true skill fast
  - Better uses event data
- Drawbacks
  - Higher computational cost
  - Parameters need to be finely tuned to ensure accuracy

5.3 Statistical Methods

5.3.1 Purpose

The purpose of this statistical analysis is to evaluate the predictive performance of an encounter-level rating system compared to a traditional game-level rating system. Performance is assessed in terms of overall predictive accuracy and mean squared margin error (MSME).

5.3.2 Design

For each of the 410 games in the dataset, ratings were computed using both the game-level and encounter-level systems. Predictive accuracy was determined by using rating knowledge from all preceding games. A prediction was counted as correct if it identified the actual winner. For games where the predicted win chance was between 40–60%, and the final scores were very close (differing by no more than 10%), the prediction was also counted as correct. Predictive accuracy was calculated as the number of correctly predicted games divided by the total number of games.

All rating computations and analyses were performed using Python, with code based on the Laserforce ranking project on GitHub. This project is a website that contains player rankings for my local laser tag arena and the data used in this study.

5.3.3 Score Margin Prediction (Mean Squared Margin Error)

To evaluate how well predicted win probabilities align with the actual game outcomes, we also used mean squared margin error (MSME). For each game, we compute:

$$\text{Actual margin} = \frac{\text{Red team score} - \text{Green team score}}{\text{Red team score} + \text{Green team score}},$$

$$\text{Predicted margin} = 2 \times \text{Red team win probability} - 1,$$

$$\text{Squared error} = (\text{Predicted margin} - \text{Actual margin})^2.$$

MSME is then calculated as the mean of the squared errors across all games:

$$\text{MSME} = \frac{1}{n} \sum_{i=1}^{n} (\text{Predicted margin}_i - \text{Actual margin}_i)^2,$$

where n is the total number of games.

MSME is lower when predicted confidence and actual game dominance are similar. An MSME of 0 indicates perfect predictions, and higher values indicate less accuracy. MSME is based on both prediction correctness and how confident the prediction was, so it captures information that standard win–loss accuracy does not.

6 Results

The performance of the encounter-level and game-level rating systems was evaluated across 410 ranked SM5 games using predictive accuracy, and mean squared margin error (MSME).

6.1 Predictive Accuracy

The encounter-level rating system correctly predicted 272 out of 410 games, resulting in an overall predictive accuracy of 66.34%. The game-level rating system correctly predicted 255 out of 410 games, giving an accuracy of 62.20%. While the encounter-level system demonstrates higher predictive accuracy, the improvement is small and should be interpreted in the context of this specific dataset rather than as a general guarantee of superior performance for all use cases.

6.2 Score Margin Prediction

Mean squared margin error (MSME) was computed to evaluate how well predicted win probabilities corresponded to actual game dominance. The encounter-level system achieved an MSME of 0.14, substantially lower than the game-level system's MSME of 0.25. This 44% reduction shows that encounter-level weighting improves the model's ability to predict how close a game will be, not just the winning team.

MSME is useful in this context because it penalizes overconfident incorrect predictions and rewards well-calibrated confidence estimates, which are important for matchmaking systems.

Rating System	Accuracy	MSME
Game-level	62.20%	0.25
Encounter-level	66.34%	0.14

Table 3: Comparison of predictive performance across 410 games for game-level and encounter-level rating systems.

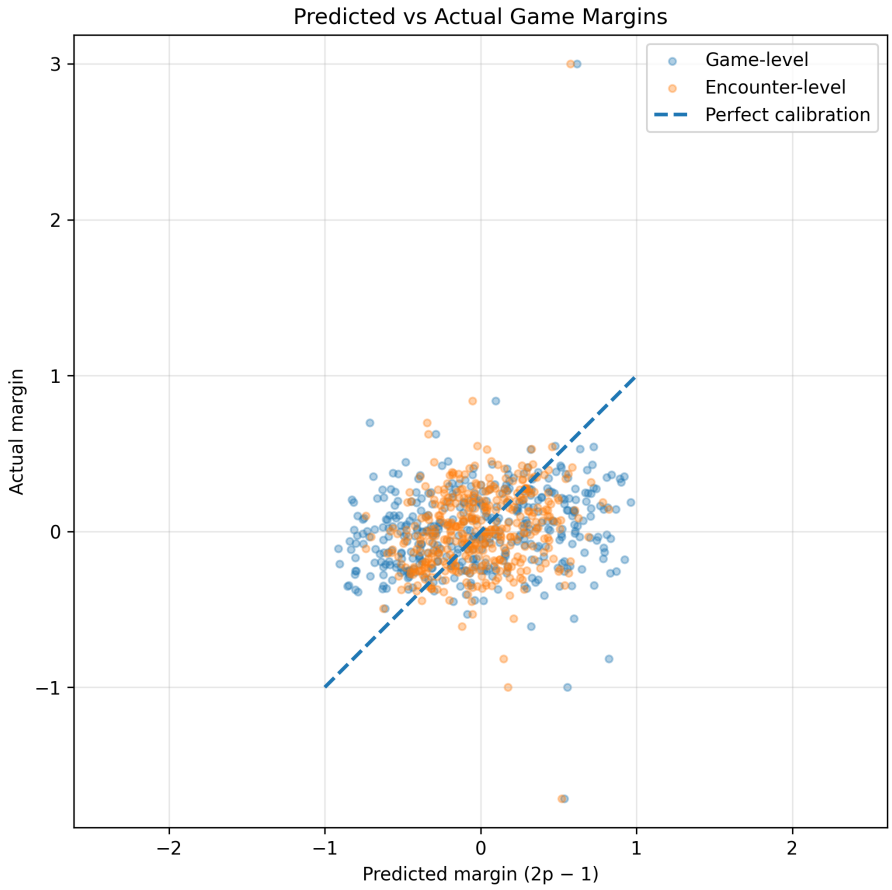

Figure 1: Calibration of predicted win margins against actual game margins for game-level and encounter-level rating systems across 410 ranked SM5 games. Points closer to the diagonal line indicate better alignment between predicted confidence and observed game dominance.

7 Discussion

7.1 Interpretation of Results

The encounter-level rating system showed consistent improvements over the game-level system in terms of probabilistic calibration and margin prediction, as shown by substantially lower mean squared margin error (MSME). Even though gains in win–loss predictive accuracy were modest, this outcome is expected in team games where match outcomes are relatively noisy and influenced by many factors like coordination and role-based skill differences. One potential

bias is how teams are selected. It's possible that these results could be different if the teams were selected by the program itself instead of by the players. An advantage to the encounter-level system is its ability to more accurately represent the degree of confidence associated with each prediction rather than merely predicting the winning team.

These results suggest that encounter-level information captures more detailed aspects of player performance that are not fully reflected in net game outcomes. Using individual encounters such as hits and missiles, the model can better distinguish between close games and clean sweeps.

7.2 Convergence Speed and Rating Stability

Unlike game-level rating systems, the encounter-level model updates player ratings multiple times within a single match. Each encounter provides additional information about player performance, which increases the number of observations available to the system. Because of this, player uncertainty values (σ) decrease rapidly, leading to faster convergence of ratings.

This faster convergence allows the encounter-level system to form confident estimates of player skill with significantly fewer completed games. Consistently comparing a player's rating to every other player's rating in the game allows the system to quickly understand a player's skill level.

7.3 Cost–Benefit Analysis

The encounter-level rating system is more complicated and requires more computing power than the traditional game-level system. Tracking and storing every event in a game adds extra overhead, and updating ratings after each encounter takes more processing. Additionally, designing and tuning the encounter weights adds another layer of complexity.

Even with these extra costs, the benefits are clear when you care about fair matchmaking and accurate confidence estimates. The system gives more accurate ratings, which helps make balanced teams, identify close games, and avoid overconfident predictions. In competitive settings with lots of players and detailed game data, the extra computations and effort are worth it because they give much more reliable rankings quickly.

7.4 Limitations

Although this study had positive results for encounter-based systems, there are some limitations. Encounter weights were selected based on observations of how important each event is, not actual game data, which introduces potential bias.

Additionally, the data the dataset was limited to a single game mode (SM5) on the Laserforce system, which restricts our generalizability to other games or game modes. Lastly, predictive accuracy metrics are sensitive to the chosen thresholds for defining close games and normalized score margins. Alternative definitions may give different results.

7.5 Potential Improvements

Future work could address these limitations by learning encounter weights directly from data using optimization or machine learning techniques. For example, optimization could be used to tune encounter contributions to minimize MSME or improve calibration. Using different ratings for each role in addition to the general rating could be used to account for differences in skill across roles.

Another improvement could be the implementation of rating games with weights based on score, which is the metric used to determine if a team won a game. Using score differences instead of treating games like binary outcomes gives additional data to the system.

References

[Jos25a] Vivek Joshy. *GitHub - vivekjoshy/openskill.py: Multiplayer Rating System. No Friction.* June 2025. URL: https://github.com/vivekjoshy/openskill.py.

[Jos25b] Vivek Joshy. *OpenSkill Plackett-Luce Model.* 2025. URL: https://openskill.me/en/v5.1.0/api/openskill/models/weng_lin/plackett_luce/index.html#openskill.models.weng_lin.plackett_luce.PlackettLuce.

[tor25] tortoise. *GitHub - tortoise/tortoise-orm: Familiar asyncio ORM for python, built with relations in mind.* Dec. 2025. URL: https://github.com/tortoise/tortoise-orm.