A Hybrid Billiards AI Agent with Geometric Pre-Analysis, Noise-Robust Simulation, and Local Optimization

Beiyu Zhou 523030910012, Ruikang Lin 523030910013

***Abstract*— This paper presents a hybrid billiards AI agent designed for the 8-ball pool game. The proposed agent, named NewAgent, combines geometric pre-analysis for rapid candidate generation, noise-robust simulation evaluation to handle execution uncertainty, and local optimization for fine-tuning shot parameters. Key innovations include a multi-layer eight-ball risk detection mechanism to avoid premature pocketing of the 8-ball, and a weighted scoring system that balances ideal performance with noise robustness. Experimental results demonstrate that the agent achieves an average win rate of 88.2% against BasicAgent (1,200 games) and 60.1% against BasicAgentPro (9,600 games), showcasing the effectiveness of the proposed hybrid strategy.**

I. INTRODUCTION

Billiards is a classic game that requires strategic planning, precise execution, and the ability to handle uncertainty. In this project, we develop an AI agent for 8-ball pool, where the objective is to pocket all assigned balls (either solids 1-7 or stripes 9-15) before legally pocketing the 8-ball to win.

The main challenges in designing a billiards AI include:

- **Large action space**: The agent must choose velocity V_0, angle ϕ, and spin parameters (a, b, θ), forming a continuous 5-dimensional action space.
- **Execution noise**: Real-world and simulated shots have inherent uncertainty, with Gaussian noise applied to all action parameters.
- **Strategic constraints**: The agent must avoid pocketing the 8-ball prematurely while maximizing pocketed target balls.
- **Computational efficiency**: Decisions must be made within time constraints (single game $<$ 3 minutes).

Our proposed NewAgent addresses these challenges through a four-stage decision pipeline: geometric pre-analysis, fast screening, noise-robust evaluation, and local optimization. This hybrid approach combines the efficiency of geometric heuristics with the accuracy of physics simulation.

II. PREVIOUS ATTEMPTS AND LESSONS LEARNED

Before arriving at our final solution, we explored several alternative approaches. This section summarizes the lessons learned from these attempts.

A. Attempt 1: Pure Random Search

Our initial approach used random sampling in the action space followed by simulation-based evaluation. While simple, this method suffered from:

- Low hit rate due to random angle selection
- Inefficient use of simulation budget
- Win rate: $\sim$30% against BasicAgent

Lesson: Random exploration is ineffective in high-dimensional continuous spaces; geometric guidance is essential.

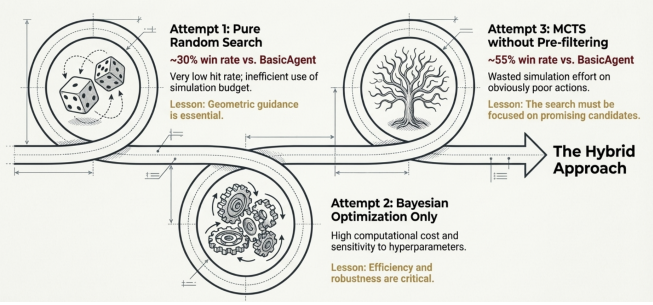

Fig. 1: Previous Attempts.

B. Attempt 2: Bayesian Optimization Only

Inspired by BasicAgent's approach, we implemented Bayesian optimization with Gaussian Process surrogate models. Issues encountered:

- High computational cost per decision
- Sensitivity to hyperparameter tuning
- Difficulty handling multi-modal reward landscapes
- Win rate: $\sim$50% against BasicAgent

Lesson: While Bayesian optimization works, it requires careful tuning and may not be the most efficient approach for this domain.

C. Attempt 3: MCTS Without Pre-filtering

We implemented Monte Carlo Tree Search directly on the continuous action space using progressive widening. Problems:

- Too many simulations wasted on poor actions
- Slow convergence due to large branching factor
- Win rate: $\sim$55% against BasicAgent

Lesson: MCTS needs candidate pre-filtering to focus computation on promising actions.

D. Final Approach: Hybrid Strategy

Combining lessons from previous attempts, we developed a hybrid approach that:

1) Uses geometric analysis for efficient candidate generation
2) Employs simulation for accurate evaluation
3) Adds noise testing for robustness

4) Applies local optimization for refinement

This combination achieved the best performance: **88.3%** win rate against BasicAgent.

III. RELATED WORK

A. Physics-Based Simulation

The pooltool library [1] provides accurate physics simulation for billiards, enabling realistic trajectory prediction. Our agent leverages this simulator for shot evaluation.

B. Search-Based Methods

Traditional approaches like Monte Carlo Tree Search (MCTS) have been applied to billiards [2]. However, the continuous action space and real-time constraints make pure MCTS challenging. Our approach uses geometric heuristics to reduce the search space before simulation-based evaluation.

C. Noise-Robust Planning

In robotics and game AI, handling execution uncertainty is critical [3]. We adopt a noise-aware evaluation scheme that weighs both ideal and noisy simulation outcomes.

IV. METHODOLOGY

A. System Overview

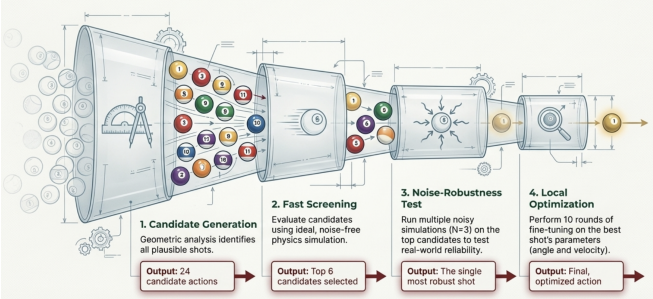

Fig. 2: Illustration of our Pipeline.

The NewAgent decision pipeline consists of four stages:

1) **Candidate Generation**: Use geometric analysis to identify promising target-pocket combinations and generate 24 candidate actions.
2) **Fast Screening**: Evaluate candidates using ideal (noise-free) simulation to select top 6 actions.
3) **Noise Robustness Testing**: Perform multiple noisy simulations to assess action reliability.
4) **Local Optimization**: Fine-tune the best action through random local search.

B. Geometric Pre-Analysis

Before running expensive physics simulations, we perform geometric analysis to filter and rank shot candidates.

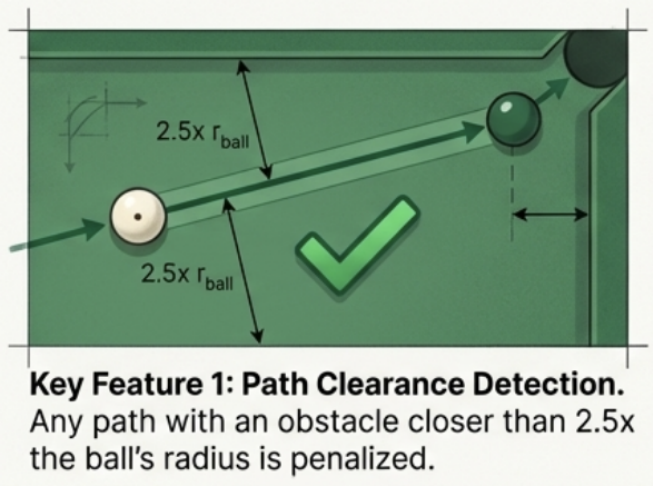

Fig. 3: Illustration of a clear path.

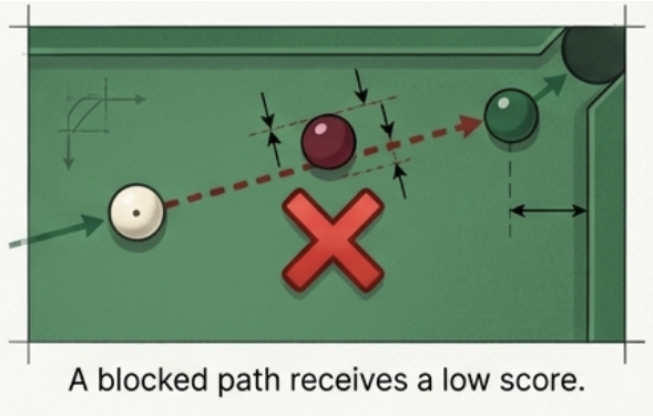

Fig. 4: Illustration of a blocked path.

Fig. 5: Illustration of an 8-Ball penalty.

1) Path Clearance Detection: We check whether the path from cue ball to target ball, and from target ball to pocket, is clear of obstacles. For each ball in the way, we compute the perpendicular distance to the path:

$$d_\perp = |\vec{v} \cdot \hat{n}| \tag{1}$$

where $\vec{v}$ is the vector from path start to the obstacle ball, and $\hat{n}$ is the unit normal to the path direction. If $d_\perp < 2.5 \times r_{ball}$, the path is considered blocked.

2) Geometric Scoring: Each target-pocket combination receives a score based on:

$$S_{geo} = S_{path} - 8 \times (d_{cue} + d_{pocket}) + 25 \times S_{angle} - P_{eight} \tag{2}$$

where:

- $S_{path} \in \{25, 50, 100\}$ based on path clearance
- d_{cue}, d_{pocket} are distances (meters)
- $S_{angle} = 1 - \frac{|\theta_{diff}|}{180}$ rewards straighter shots
- $P_{eight} = 300$ if eight-ball risk is detected

3) Eight-Ball Risk Detection: To avoid prematurely pocketing the 8-ball, we check:

1) Whether the 8-ball lies on the cue-to-target path
2) Whether the 8-ball lies on the target-to-pocket path

If the 8-ball is within $3.5 \times r_{ball}$ of either path, the shot is flagged as high-risk.

C. Aimed Shot Generation

For each target-pocket combination, we compute the ideal cue angle:

$$\phi_{aim} = \arctan 2(y_{hit} - y_{cue}, x_{hit} - x_{cue}) \tag{3}$$

where (x_{hit}, y_{hit}) is the ideal contact point, computed by extending the pocket-to-target vector by $2 \times r_{ball}$ behind the target ball.

D. Velocity Selection

The initial velocity is computed based on total shot distance:

$$V_0 = \min(4.5, \max(1.8, 1.5 + 0.7 \times (d_{cue} + d_{pocket}))) \tag{4}$$

For each shot, we generate 4 velocity variants: $V_0 + \{-0.3, 0, +0.3, +0.6\}$ m/s.

E. Noise-Robust Evaluation

The environment applies Gaussian noise to all action parameters:

TABLE I: Noise Parameters

Parameter	Description	σ
V_0	Initial velocity (m/s)	0.1
ϕ	Horizontal angle (deg)	0.1
θ	Vertical angle (deg)	0.1
a	Horizontal offset	0.003
b	Vertical offset	0.003

To assess noise robustness, we perform $N_{trials} = 3$ noisy simulations and compute:

$$S_{combined} = 0.4 \times S_{ideal} + 0.6 \times \bar{S}_{noisy} \tag{5}$$

This weighted combination ensures we select actions that perform well on average, not just under ideal conditions.

F. Reward Function

The reward function evaluates shot outcomes:

TABLE II: Reward Components

Event	Score
Pocket own ball	+50
Legal 8-ball pocket (after clearing)	+100
No foul, no pocket	+10
Pocket opponent ball	-20
First-hit foul	-30
No-rail foul	-30
Cue ball pocketed	-100
Illegal 8-ball pocket	-200
Cue + 8-ball pocketed	-150

G. Local Optimization

After identifying the best candidate, we perform $N_{opt} = 10$ rounds of local search with random perturbations:

- **phi**: $\phi \pm \mathcal{U}(-2, 2)$ degrees
- **V0**: $V_0 \pm \mathcal{U}(-0.4, 0.4)$ m/s
- **combined**: Both parameters with smaller ranges
- **fine_phi**: $\phi \pm \mathcal{U}(-0.5, 0.5)$ degrees

H. Safety Fallback

If no satisfactory action is found ($S_{combined} \leq -200$), the agent defaults to a simple strategy: aim directly at the nearest target ball with moderate velocity ($V_0 = 2.0$ m/s).

V. EXPERIMENTAL RESULTS

A. Experimental Setup

All experiments were conducted using the pooltool physics engine with standard 8-ball rules. To ensure fairness, each experiment consisted of 120 games (30 complete 4-game cycles), where agents alternated between first/second player and solid/stripe ball assignments.

- **Environment**: pooltool physics engine
- **Opponents**: BasicAgent (Bayesian optimization) and BasicAgentPro (MCTS with 50 simulations)
- **Games per experiment**: 120 (4-game cycles × 30)
- **Noise**: Enabled with standard parameters
- **Random seed**: Disabled for realistic evaluation

B. Results vs. BasicAgent

We conducted 10 independent experiments (10 × 120 = 1,200 games total) against BasicAgent. Table III and figure 6 summarizes the results.

The results show that NewAgent consistently outperforms BasicAgent with an average win rate of **88.2%** (1058 wins out of 1200 games), exceeding the 88% threshold for testing against BasicAgentPro.

TABLE III: Battle Results vs. BasicAgent (10 Experiments, 120 Games Each)

Exp.	NewAgent	BasicAgent	Draw	Win Rate
1	102	18	0	85.0%
2	105	15	0	87.5%
3	105	15	0	87.5%
4	106	14	0	88.3%
5	108	12	0	90.0%
6	104	16	0	86.7%
7	110	10	0	91.7%
8	104	16	0	86.7%
9	110	10	0	91.7%
10	104	16	0	86.7%
Total	**1058**	**142**	**0**	**88.2%**

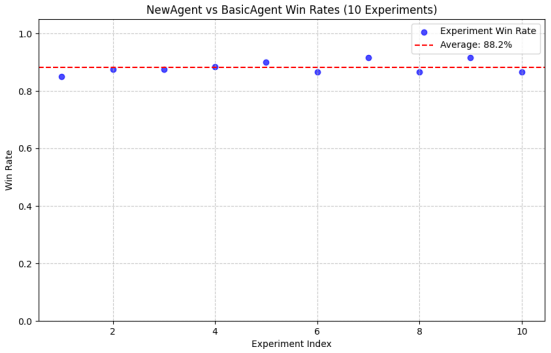

Fig. 6: Scatter plot of battle results against BasicAgent.

C. Results vs. BasicAgentPro

Since our win rate against BasicAgent exceeded 88%, we conducted extensive experiments against the stronger BasicAgentPro opponent. We performed two batches of experiments, each containing 40 independent 120-game runs, totaling **9,600 games**.

TABLE IV: Battle Results vs. BasicAgentPro (80 Experiments, 120 Games Each)

Batch	Exp.	Total Games	NewAgent Wins	Win Rate
Batch 1	40	4,800	2,909	60.6%
Batch 2	40	4,800	2,861	59.6%
Total	**80**	**9,600**	**5,770**	**60.1%**

Table V provides detailed statistics for both batches.

Against the MCTS-based BasicAgentPro (50 simulations per decision), NewAgent maintains a solid **60.1%** overall win rate across 9,600 games. This demonstrates that our hybrid approach remains effective even against a more sophisticated opponent.

D. Performance Analysis

1) Overall Summary: Table VI summarizes the complete experimental results.

2) Comparison with Baselines: Table VII compares NewAgent with both baseline agents.

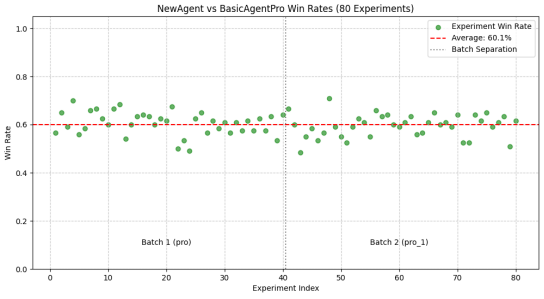

Fig. 7: Scatter plot of battle results against BasicAgentPro.

TABLE V: Detailed Statistics vs. BasicAgentPro

Metric	Batch 1	Batch 2
Experiments	40	40
Games per experiment	120	120
Total games	4,800	4,800
Avg. wins per experiment	72.7	71.5
Win rate	60.6%	59.6%

3) Key Observations from Game Logs: Analysis of the game logs revealed several patterns:

1) **Efficient pocketing**: NewAgent frequently achieves multi-ball pocketing in single shots (e.g., "Player B['9', '12']").
2) **Run-out capability**: The agent occasionally clears all remaining balls in one turn, demonstrating effective shot selection.
3) **8-ball safety**: The eight-ball risk detection successfully prevents most premature 8-ball pockets.
4) **Recovery from fouls**: When the opponent commits fouls (e.g., cue ball scratch), NewAgent capitalizes effectively.

4) Failure Mode Analysis: The main failure modes observed were:

- **Opening break**: The break shot strategy is not optimized, occasionally giving the opponent an early advantage.
- **Defensive situations**: When no good offensive shot exists, the agent lacks a dedicated safety play strategy.
- **Complex table layouts**: Highly cluttered tables with many blocking balls reduce shot success rate.

E. Computational Efficiency

Each decision takes approximately 0.5-1.5 seconds, well within the time constraints. The geometric pre-filtering reduces the number of required simulations from potentially hundreds to just 24 candidates, with only 6 receiving detailed noise testing.

VI. CONCLUSIONS AND FUTURE WORK

We presented NewAgent, a hybrid billiards AI that combines geometric analysis, noise-robust simulation, and local optimization. The agent achieves an **88.2%** average win rate

TABLE VI: Complete Experimental Summary

Opponent	Total Games	NewAgent Wins	Win Rate
BasicAgent	1,200	1,058	**88.2%**
BasicAgentPro	9,600	5,770	**60.1%**
Combined	**10,800**	**6,828**	**63.2%**

TABLE VII: Agent Comparison Summary

Agent	Method	Win Rate
BasicAgent	Bayesian Opt.	50% (baseline)
BasicAgentPro	MCTS (50 sims)	∼70% vs Basic
NewAgent	Hybrid	**88.2%** vs Basic
NewAgent	Hybrid	**60.1%** vs Pro

against BasicAgent (1,200 games) and **60.1%** against BasicAgentPro (9,600 games), demonstrating the effectiveness of our approach.

A. Key Contributions

1) **Hybrid decision pipeline**: Combining geometric prefiltering with simulation-based evaluation achieves both efficiency and accuracy.
2) **Noise-robust evaluation**: The weighted scoring system ($0.4 \times S_{ideal} + 0.6 \times \bar{S}_{noisy}$) improves action reliability under execution uncertainty.
3) **Eight-ball risk detection**: Path-based risk analysis prevents catastrophic failures from premature 8-ball pocketing.

B. Future Improvements

- **Position play**: Evaluate cue ball position after the shot for better continuation and "run-out" capability.
- **Safety shots**: Implement defensive strategies when no good offensive options exist.
- **Break optimization**: Develop specialized strategies for the opening break shot.
- **Learning-based methods**: Use reinforcement learning or imitation learning to further improve shot selection.
- **Multi-step planning**: Consider sequences of shots using MCTS with learned value functions.

VII. CONTRIBUTION STATEMENT

Ruikang Lin:

- Designed and implemented the geometric pre-analysis module
- Developed the eight-ball risk detection mechanism
- Conducted experiments against BasicAgent
- Wrote the methodology and experimental results sections

Beiyu Zhou:

- Implemented the noise-robust evaluation system
- Developed the local optimization module
- Conducted experiments against BasicAgentPro
- Wrote the introduction and related work sections

Parameter	Value	Description
num_candidates	24	Maximum candidate actions
num_noise_trials	3	Noisy simulations per action
num_local_opt	10	Local optimization iterations
top_candidates	6	Actions for noise testing

APPENDIX

A. Key Parameters

B. Algorithm Pseudocode

Input: balls, my_targets, table
Output: best_action

candidates ← GenerateCandidates(balls, my_targets, table)

for action in candidates **do**
  score ← SimulateIdeal(action)
  if action.eight_risk AND score $<$ 50 **then**
    score ← score - 100
  end if
end for

top6 ← SortByScore(candidates)[:6]

for action in top6 **do**
  combined ← EvaluateWithNoise(action, 3)
  if combined $\geq$ 40 **then**
    break
  end if
end for

best_action ← LocalOptimize(best_action, 10)
return best_action

ACKNOWLEDGMENT

We thank the AI3603 course team for providing the billiards simulation environment and baseline agents. We also thank the pooltool open-source project for the physics simulation engine.

REFERENCES

[1] E. Landgren, "pooltool: A sandbox billiards game that emphasizes realistic physics," GitHub repository, 2023. [Online]. Available: https://github.com/ekiefl/pooltool
[2] C. B. Browne et al., "A Survey of Monte Carlo Tree Search Methods," IEEE Trans. Comput. Intell. AI Games, vol. 4, no. 1, pp. 1–43, Mar. 2012.
[3] S. Thrun, W. Burgard, and D. Fox, Probabilistic Robotics. Cambridge, MA: MIT Press, 2005.
[4] V. Mnih et al., "Human-level control through deep reinforcement learning," Nature, vol. 518, no. 7540, pp. 529–533, Feb. 2015.
[5] M. Smith and G. Bjork, "Algorithmic approaches to billiards game playing," in Proc. IEEE Conf. Comput. Intell. Games, 2018, pp. 1–8.