

# Simulation of ray behavior in biconvex converging lenses using machine learning algorithms

Juan Deyby Carlos-Chullo, Marielena Vilca-Quispe, Whinders Joel Fernandez-Granda,  
Eveling Castro-Gutierrez

Universidad Nacional de San Agustín de Arequipa, Arequipa, Peru

---

## Article Info

### Article history:

Received May 20, 2024

Revised Oct 21, 2024

Accepted Oct 30, 2024

### Keywords:

Converging biconvex lenses

Machine learning

Proximal policy optimization

Reinforcement learning

Soft actor-critic

## ABSTRACT

This study used machine learning (ML) algorithms to investigate the simulation of light ray behavior in biconvex converging lenses. While earlier studies have focused on lens image formation and ray tracing, they have not applied reinforcement learning (RL) algorithms like proximal policy optimization (PPO) and soft actor-critic (SAC), to model light refraction through 3D lens models. This study addresses that gap by assessing and contrasting the performance of these two algorithms in an optical simulation context. The findings of this study suggest that the PPO algorithm achieves superior ray convergence, surpassing SAC in terms of stability and accuracy in optical simulation. Consequently, PPO offers a promising avenue for optimizing optical ray simulators. It allows for a representation that closely aligns with the behavior in biconvex converging lenses, which holds significant potential for application in more complex optical scenarios.

*This is an open access article under the [CC BY-SA](#) license.*



---

## Corresponding Author:

Juan Deyby Carlos Chullo

Universidad Nacional de San Agustín de Arequipa

Arequipa, Peru

Email: jcarlosc@unsa.edu.pe

---

## 1. INTRODUCTION

Converging lenses, such as biconvex lenses, are designed to form both real and virtual images [1]. These lenses are essential for improving the precision with which we observe and study objects [2]. Through the refraction of light, converging lenses enable illuminated objects to project onto a screen, creating images that can be examined for various scientific purposes [3]. While several applications simulate image formation through these lenses, many do not fully capture the complex behavior of light rays.

One such application, AR-GiOs, as analyzed in [4], has shown promising results in the academic field, particularly for learning about the formation of real and virtual images. However, despite its success in educational settings, AR-GiOs still struggles to accurately simulate the behavior of rays passing through optical systems. This gap highlights the limitations of current simulation tools in capturing the subtle details of ray behavior, which are fundamental to the study of physical optics.

Several applications attempt to simulate image formation through lenses, but they often fail to accurately model the light rays involved in the process [4], [5]. These rays, referred to as principal, central, and focal rays, are essential for understanding key optical behaviors when light passes through lenses or mirrors. Accurate simulation of these rays is crucial because they dictate how images are formed and how optical systems function, yet many existing tools lack the necessary fidelity to simulate them effectively.

No studies have been identified that apply reinforcement learning (RL) algorithms to model light refraction through lenses. RL methods like proximal policy optimization (PPO) [6], [7] and soft actor-critic (SAC) [8], [9] are extensively used in artificial intelligence (AI) and machine learning (ML) for decision-making tasks [10], [11]. These algorithms operate based on learning from interactions with their environment, where an agent makes decisions and is given feedback through rewards or penalties [12]. Due to the dynamic nature of light refraction, RL algorithms have the potential to improve the precision of ray simulations.

This article proposes the use of PPO and SAC algorithms to control the trajectory of rays as they pass through a lens, guiding them to converge at points where virtual or real images are formed. By applying the thin lens equation and magnification formulas, the deviation and trajectory of the rays are calculated as they interact with the lens [13]. A simulator created in Unity utilizes these RL algorithms to simulate the passage of the three critical rays (principal, central, and focal) through a converging lens, aiming to achieve accurate ray convergence and image formation.

While it is possible to simulate converging rays through a lens, achieving an accurate simulation of rays passing through a 3D lens model requires highly complex and computationally demanding models. Given the detailed geometry of converging lenses, it is not feasible to approximate their shape using multiple primitive models in Unity. Therefore, moderately complex 3D models and RL algorithms are employed to enhance the accuracy of the simulation.

The remainder of this article is structured as follows: section 2 covers related works, section 3 details the proposed simulation of ray behavior in biconvex converging lenses, and section 4 provides the results and discussion. Lastly, section 5 outlines our conclusions and suggests directions for future work.

## **2. RELATED WORKS**

### **2.1. Converging biconvex lenses**

Biconvex lenses, with their two curved surfaces facing outward, serve as an example of converging lenses. It is crucial to note that, despite their appearance, these lenses are positive (with thickness decreasing from the center towards the edges) and have the ability to focus light rays [14]. Commonly used in optics courses in schools or universities, these lenses are employed to illustrate the principles of refraction and the formation of both real and virtual images [1].

### **2.2. Machine learning**

ML is a crucial branch of AI, enabling computers to process information and learn from it [12]. Through the use of algorithms, ML addresses complex data problems and automates processes, with applications in various fields such as data mining, image analysis, and predictive modeling [15]. Its broad applicability extends into numerous scientific areas, particularly within the physical sciences, where it applies algorithms and modeling techniques for data analysis in disciplines like statistical mechanics, high-energy physics, cosmology, quantum many-body systems, quantum computing, chemistry, and materials research [16].

### **2.3. Reinforcement learning**

RL is a ML method where an agent engages with its environment and discovers an optimal strategy through trial and error [10], [17]. It is recognized as one of the three primary types of ML, alongside supervised and unsupervised learning. Unlike other approaches, its objective is to acquire different actions based on the conditions in the environment, with the agent serving as the principal decision-maker [17]. RL has demonstrated significant potential for advancing AI [18]. In this framework, the agent receives feedback from the environment but lacks access to labeled data or explicit guidance. It is employed in sequential decision-making tasks across various domains, including natural and social sciences, engineering, and AI [19].

#### **2.3.1. Proximal policy optimization**

PPO is a RL technique that has demonstrated cutting-edge performance across a range of challenging tasks [20]. PPO has been utilized in multiple areas, including robotics, gaming, and autonomous systems, to improve agent performance in complex environments. For example, in [21], PPO was employed to automate simulated autonomous driving, leading to enhanced outcomes. Similarly, in [22], PPO was effectively used to predict stock market trends, highlighting its versatility and efficiency in financial applications.

### 2.3.2. Soft actor-critic

SAC operates within the maximum entropy RL framework, aiming to maximize both expected performance and entropy simultaneously, thereby enabling actors to act with maximum randomness while achieving task success [8]. Its efficacy has been extensively evaluated in various experiments, including tests on Atari games and a large-scale MOBA game, as demonstrated in [23]. In comparative studies, PPO has consistently emerged as a top performer, as evidenced by comparisons with SAC across different test conditions [24]. Specifically, PPO showcased superior performance, especially in scenarios involving a high number of units and layers.

In research comparing RL algorithms, PPO consistently demonstrates remarkable performance, surpassing SAC in various conditions, particularly when dealing with complex architectures [25]. Furthermore, in comparative studies of deep RL algorithms, PPO consistently outperforms alternatives like DDPG, SAC, and TD3, as demonstrated in [26]. To implement RL in simulation environments, practitioners often leverage tools such as Unity and ML-Agents, as highlighted in previous research [27], [28].

## 3. METHOD

### 3.1. Proposed simulation of ray behavior in converging biconvex lenses

This work aims to develop a simulation of ray behavior in converging biconvex lenses using 3D models, utilizing RL techniques like PPO and SAC to modify the refraction angles of light rays passing through a lens. The goal is to compare the feasibility and stability of these algorithms in achieving more accurate simulation results. The proposal includes the following steps as outlined in Table 1.

Table 1. Steps for simulating light through a biconvex lens

Steps	Description
1	Define the objective
2	Model the converging biconvex lens using Blender
3	Develop a simulation environment using Unity
4	Identify constraints and critical properties
5	Explanation of PPO and SAC algorithms
6	RL environment configuration

### 3.2. Simulation of converging biconvex lens behavior

To validate the proposal, a simulator based on RL algorithms, specifically PPO and SAC, has been developed. These algorithms were employed to accurately model and simulate the behavior of light rays in converging biconvex lenses under simulated conditions.

#### 3.2.1. Define the objective

The objective is to compare the feasibility and stability of the PPO and SAC algorithms within the context of simulating converging biconvex lenses. The purpose is to determine which of these algorithms is more effective in achieving accurate and reliable simulation results that precisely describe the behavior of light rays interacting with converging biconvex lenses.

PPO and SAC were selected because of their extensive use in the literature, being recognized for effectively managing both single-agent settings as well as multi-agent cooperative and competitive scenarios. Additionally, other RL algorithms, such as MA-POCA [29], also support these types of environments. However, this study focuses on PPO and SAC due to their popularity and demonstrated success in complex physical simulations.

#### 3.2.2. Modeling the converging biconvex lens using Blender

In the simulation, two distinct models of converging biconvex lenses were implemented to ensure high precision in ray tracing. Each model, depicted in Figure 1, was constructed using Blender 3D version 3.6.1 and has a size of 6.5 MB. These models consist of 141,604 vertices and 269,316 triangles, with one model having its surface normals oriented inward and the other outward. This difference in normal orientation was essential to enable precise collision detection by the rays emitted using Unity's Raycast function, both when entering and exiting the lens. The models were generated by intersecting two spheres, which were created in Blender with the highest possible level of detail, constrained by the computational capabilities and the limits of the

Blender software. Each sphere has a radius of 10 meters, and their centers are separated by a distance of 19.9 units, resulting in an intersection of 0.1 meters. This intersection was chosen to produce a lens thin enough to avoid the optical aberration that occurs in thicker lenses.

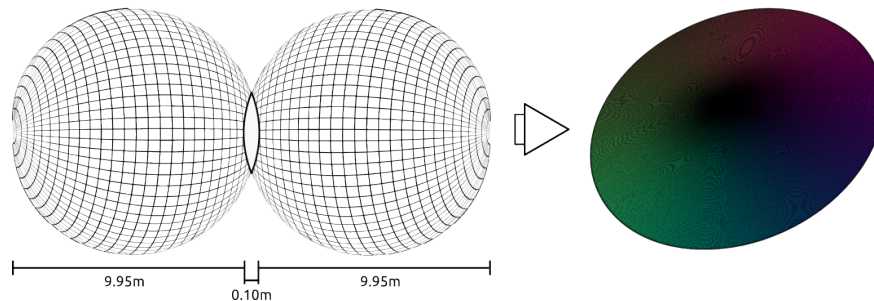


Figure 1. Characteristics of the spheres with 256 rings and 2048 segments each, with dimensions of 10x10x10 meters and an intersection of 0.10 meters, resulting in a converging biconvex lens of approximately 67 rings, 2048 segments, and dimensions of 1.977 meters x 1.977 meters x 0.1 meters

### 3.2.3. Develop a simulation environment using Unity

In the context of optical ray simulation with a converging lens, a simulation environment was developed using Unity version 2021.3.11f1 and the RL agents library, a popular tool in RL environments [7]. This environment includes elements such as a converging biconvex lens, focal points, and a ray launch point to determine the initial direction and trajectory of the rays passing through the lens. The development aimed to apply PPO and SAC algorithms to simulate and optimize the behavior of light rays.

Figure 2 presents a screenshot of the simulation environment within Unity. It illustrates the converging lens, focal points, and the trajectory of three types of rays projected from a designated origin point. These rays include the principal, central, and focal rays, moving from left to right from the viewpoint, showcasing the simulated optical phenomena.

The simulator designed for this study integrates physics optics principles into Unity's framework, providing a platform for comparing the performance of PPO and SAC algorithms in simulating light ray trajectories. Central to the simulator is the algorithm responsible for tracing the path of light rays as they interact with the lens surfaces. By recording collision points, the behavior of rays can be analyzed, informing adjustments required for accurate simulation. The utilization of the thin lens formula and lens magnification aids in calculating the optimal points for ray passage or approach, enhancing the realism of the simulation.

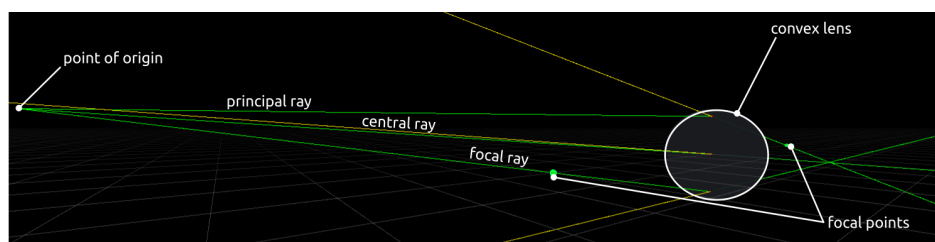


Figure 2. Simulation environment developed in Unity. The environment contains a converging biconvex lens. Focal points are located on both sides of the lens. Three rays – principal ray, central ray, and focal ray – are projected from the origin point. All rays are projected from left to right from the viewpoint

### 3.2.4. Identifying constraints and critical properties

A key constraint is to limit the rays to three specific types: principal, central, and focal, due to their relevance in the field of optics. Additionally, the launch point of the rays was restricted to a distance of 10 meters from the lens, and along the Y and Z axes within a range that ensures collision with the lens, preventing the rays from escaping into empty space or striking the lens edges. Given that the lens has a radius of 1 meter,

a maximum radius of 0.975 meters was chosen to avoid reaching the edge. Furthermore, the refractive index of the lens was set to 2, despite glass typically having a value of 1.45, in order to achieve tighter convergence of the rays. Finally, the number of interactions during the training of the PPO and SAC algorithms was limited to 500k due to the time required for training, with 48 simulated ray instances, taking approximately 20 to 30 minutes to complete the training for each ray type.

**3.2.5. Explanation of PPO and SAC algorithms**

The PPO method is a widely used on-policy algorithm in RL, based on combining value and policy gradients to optimize agent performance [21], [30]. Its key objective is to make sure that, after updating the policy, it remains relatively close to the previous one. To avoid drastic shifts, PPO incorporates a clipping mechanism. This algorithm samples data from its environment and uses stochastic gradient descent to optimize a clipped loss function [31].

In contrast, SAC is an off-policy algorithm in RL that follows an actor-critic approach and does not rely on predefined models or rules [30]. SAC employs a revised RL objective function and emphasizes maximizing rewards over the agent’s lifespan along with policy entropy [31]. Therefore, in this study, visual analyses were conducted to assess whether the AI agent responsible for adjusting the angle in optical ray simulators was trained using ML, ensuring its stability and applicability. To achieve this, the Unity ML agent was utilized, and a comparison between the PPO and SAC algorithms was performed.

**3.2.6. RL environment configuration**

In this study, the ray’s origin point is randomly selected within a distance of 2F from the X-axis, allowing its location at any point within the area of a circle defined by the Y and Z axes, as shown in Figure 3. In this context, the variable observed by the agent is the radius, representing the distance from the center of the circle to the origin point of the ray. The agent makes decisions based on this variable while interacting with the environment and the converging biconvex lens.

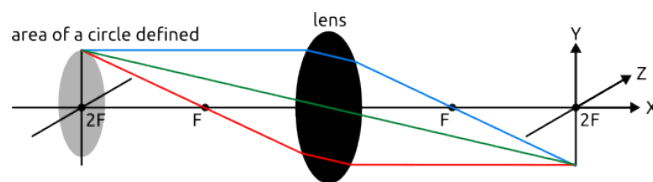


Figure 3. The starting points of the rays originate at a distance of 2F from the right side. Knowing that F is at a distance of 2.5 meters and 2F at 5 meters

To ensure proper behavior and prevent unnecessary collision repetitions, a small displacement of  $10^{-8}$  meters in the direction of the ray was applied each time a collision occurred with the lens model. This displacement was necessary because the ray could collide with both internal and external colliders generated from the normals of the lens model, ensuring that the ray continued its trajectory without additional interference in subsequent collisions Figure 4. Throughout the training process, the agent received observations from the environment, aiding in the decision-making process. Rewards were utilized to motivate the agent to adjust the angle of the rays after colliding with the lens.



Figure 4. The guidelines, for instance, in the case of the principal ray at the top of the lens, are as follows: the green lines represent the ray’s trajectory, the red lines correspond to the normals at the points where the ray collided, and the yellow line is used to project the resulting ray

RL agent, the RL agent's task is to adjust the ray's direction each time it passes through the lens, aiming to minimize the distance between the resulting ray's path and the target point. This target point is calculated using the thin lens formulas in (1) and the magnification in (2). The formula applied for converging thin lenses is known as the thin lens equation, which relates the image distance ( $d_i$ ), object distance ( $d_o$ ), and focal length ( $f$ ) of the lens, as shown in (2). Additionally, lateral magnification is used, linking the image height ( $h_i$ ), object height ( $h_o$ ), image distance ( $d_i$ ), and object distance ( $d_o$ ) as presented in (1).

The availability of these actions for the agent depends on the state of the environment at that moment. In this particular case, the decision has been made to apply the RL algorithms PPO and SAC, taking only one action in each training cycle. Unlike many approaches described in the literature, where multiple actions are taken in each training cycle, in this environment, the agent takes only one action out of two possible actions in each training cycle:

- The refraction angle will only be modified when the emitted light reaches the four key points (the point of origin, the external collision point where the light enters the lens, the internal collision point where it exits the lens, and the endpoint).
- Rays that have three points or fewer will be discarded and will not be considered in the simulation because some rays do not collide with the lens, which is due to defects in the 3D model.

$$\frac{1}{f} = \frac{1}{d_o} + \frac{1}{d_i} \quad (1)$$

$$m = \frac{h_i}{h_o} = -\frac{d_i}{d_o} \quad (2)$$

Hyperparameters, similar to other RL algorithms, both PPO and SAC have multiple hyperparameters that influence the agent's performance in a converging lens environment. In this case, the objective is to adjust the angle of refraction to produce an outgoing ray trajectory that falls within a precision threshold of 0.01 meters from the point previously calculated using lens and magnification formulas. The reward is determined by the distance between the resulting ray and the target point. Table 2 lists the hyperparameters of PPO and SAC used in this study, with both configurations based on examples from Unity's ML-Agents toolkit.

Table 2. Hyperparameters for PPO and SAC algorithms in the experiment. Parameters include policy deviation penalty, learning rate, batch size, iterations, samples, and entropy settings. These values affect the agents' performance in simulating light ray trajectories through a converging lens

Hyperparameter	PPO value	SAC value
Policy deviation penalty coefficient	0.2	-
Learning rate	0.0001	0.0003
Batch size	64	0.-
Number of iterations	10	-
Number of collected samples	1000	-
Replay buffer size	-	1000
Target entropy	-	0.2
Entropy regularization factor	-	0.01
Minimum entropy	-	0.5

Reward function, Figure 5 provides a visual representation of how this reward function operates within the agent-environment interaction. The reward mechanism plays a crucial role in the agent's learning process [32]. To achieve the intended behavior, a specific goal needs to be defined for optimization. The agent's task is to adjust the angle of refraction so that the resulting ray is within at least 0.01 meters of the target point. As the ray passes through the 3D model, the reward is determined by:

- If the distance to the target is less than or equal 0.01 m.
- Reward = 1.0

- If the distance to the target is greater than to 0.01 m.
  - The change in distance to the target is calculated, indicating whether the ray is approaching or moving away from the target. This is achieved by subtracting the current distance to the target from the distance it had in the previous step.
  - If the change in distance is positive, the agent is rewarded for approaching the target.
  - If the change in distance is negative, the agent is penalized for moving away from the target.

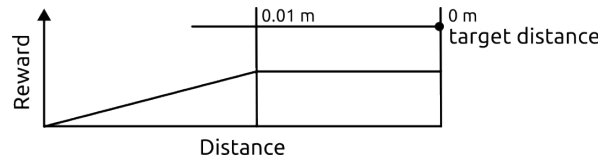


Figure 5. Visual depiction of the reward function governing agent-environment interaction. Rewards are determined by ray proximity to the target, encouraging convergence and discouraging divergence. The figure clarifies reward dynamics in collision scenarios and distance relationships

Experiment environment, the setup includes an AMD Ryzen 7 3800XT processor, an NVIDIA GeForce RTX 3070 GPU, and 32 GB of RAM. Additionally, the following software versions were utilized: Unity Engine 2021.3.11f1, TensorFlow 2.13.0, and Unity ML-Agents Toolkit Release 20. These specifications were chosen to ensure a robust and efficient system capable of handling complex simulations and the computationally demanding tasks required for training RL algorithms in optical ray simulations.

#### 4. RESULTS AND DISCUSSION

We found that the PPO algorithm achieved superior ray convergence with higher stability and accuracy than SAC. PPO reached a reward above 0.99 in fewer steps for principal, central, and focal rays, while SAC showed improvement only after 500k steps, making PPO better suited for optimizing optical ray simulators.

##### 4.1. Evaluate the behavior of PPO and SAC algorithms

The learning results were visualized using TensorBoard, a tool from TensorFlow, with data generated for the principal, central, and focal rays. As shown in Figures 6 and 7, the PPO algorithm achieved rewards of 0.9932, 0.9943, and 0.9938 for the principal, central, and focal rays, respectively, within 200k steps, successfully meeting the target reward of 0.99 with a precision threshold of 0.01m. In contrast, the SAC algorithm obtained rewards of 0.8951, 0.8829, and 0.8715 for the same rays over the same steps, falling short of the target. While SAC showed improvement between 200k and 500k steps, it still lagged behind PPO in accuracy and stability. PPO’s results closely aligned with the predictions from the thin lens formula, demonstrating its superior performance, consistent with previous studies highlighting PPO’s effectiveness in achieving reliable outcomes in similar simulation tasks.

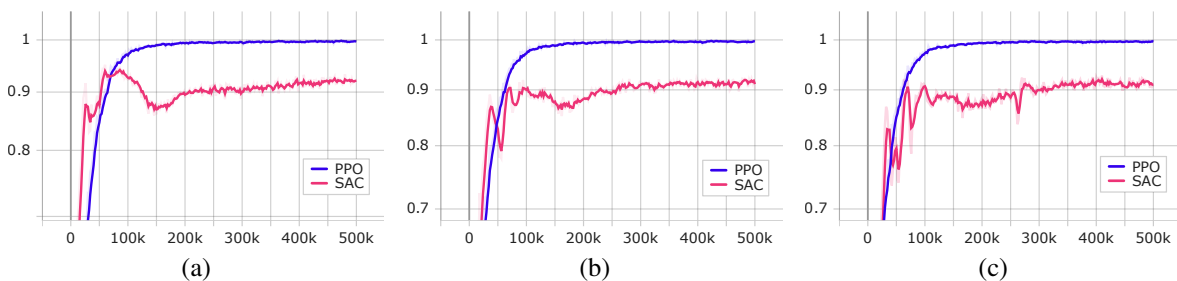


Figure 6. Comparison of PPO and SAC algorithms in different rays (a) principal ray, (b) center ray, and (c) focal ray. The graphs represent environment/cumulative reward. PPO is represented in blue, and SAC in pink. The PPO algorithm shows stable performance throughout while SAC experiences early signs of overfitting but eventually stabilizes



Run	Smoothed Value	Step	Time	Relative
256_4096_9.975_x0_ppo_v2_0.01_N_Pow2+1.0.02_center\Ray	0.9943	0.994	200,000 6/8/23, 8:37 PM	9.326 min
256_4096_9.975_x0_ppo_v2_0.01_N_Pow2+1.0.02_focal\Ray	0.9938	0.9945	200,000 6/9/23, 12:06 AM	9.874 min
256_4096_9.975_x0_ppo_v2_0.01_N_Pow2+1.0.02_principal\Ray	0.9932	0.992	200,000 6/11/23, 2:12 PM	9.27 min
256_4096_9.975_x0_sac_v2_0.01_N_Pow2+1.0.02_center\Ray	0.8829	0.8855	200,000 6/8/23, 9:09 PM	13.76 min
256_4096_9.975_x0_sac_v2_0.01_N_Pow2+1.0.02_focal\Ray	0.8715	0.865	200,000 6/9/23, 12:38 AM	15.14 min
256_4096_9.975_x0_sac_v2_0.01_N_Pow2+1.0.02_principal\Ray	0.8951	0.89	200,000 6/11/23, 2:43 PM	13.71 min

Figure 7. Results of all rays (principal, central, and focal) for the PPO and SAC algorithms at the 200k step

#### 4.2. Principal findings

Optical physics simulators are a critical area of study, as they require agents capable of executing behaviors in real-time under various circumstances. In this project, a simulator for converging rays and lenses was designed to evaluate the performance of algorithms like PPO and SAC. The results contribute to improving realistic representations of optical phenomena, showing that PPO effectively emulates the behavior of objective optical systems and accurately reproduces the predicted outcomes.

#### 4.3. Comparison to prior work

In comparison with previous applications of PPO and SAC in other domains, such as game simulations and general physics environments [12], [30], this project stands out by developing a simulator focused on the behavior of rays in converging biconvex lenses. Unlike other applications that use rays as guides but do not simulate them correctly [4], this work allows for the visualization of the optical phenomenon using only rays trained with RL. The results demonstrate how a ray is refracted when entering and exiting the lens. The choice of PPO, based on its stability and adaptability, has proven effective for this complex task, distinguishing this research from previous works focused on broader or less specific areas.

#### 4.4. Strengths and limitations

This study explored the application of RL algorithms to simulate ray behavior in a single type of converging biconvex lens with three specific ray types (principal, central, and focal). However, further and more comprehensive studies are needed to explore the behavior of additional ray types and more complex optical systems, such as multi-lens setups. Using a dense 3D polygonal mesh also introduced computational challenges, such as memory limitations and occasional missed collisions, which may have impacted the accuracy of the simulations.

### 5. CONCLUSION

This research successfully applied RL algorithms to simulate the behavior of light rays passing through biconvex converging lenses, demonstrating the viability of RL in modeling optical phenomena. The results, particularly with PPO achieving a reward exceeding 0.99 with high accuracy, indicate its superior stability and efficiency in this context. In contrast, SAC, while known for its general applicability in various domains, underperformed in this specific scenario. This finding aligns with the need to tailor RL algorithms to problem-specific dynamics, as SAC's versatility in other studies was not considered in this study. Recent observations suggest that RL algorithms can significantly improve the accuracy of optical ray simulations. Our findings provide conclusive evidence that PPO, in particular, enhances the precision of ray convergence through biconvex lenses, making it a promising tool for future optical system modeling. Our study demonstrates that PPO is more reliable for simulating ray behavior in optical systems. Future studies may explore the application of these algorithms to multi-lens systems, where ray tracing becomes more intricate.

### ACKNOWLEDGMENTS

Thanks to the "Research Center, Transfer of Technologies and Software Development R + D + i" - CiTeSoft EC-0003-2017-UNSA, for their collaboration in the use their equipment and facilities, for the development of this research work.






## REFERENCES

- [1] H. Isik, "Comparing the images formed by uses of lens surfaces," *Physics Education*, vol. 58, no. 3, p. 035002, May 2023, doi: 10.1088/1361-6552/acb87d.
- [2] K. Fliegau, J. Sebald, J. M. Veith, H. Spiecker, and P. Bitzenbauer, "Improving early optics instruction using a phenomenological approach: a field study," *Optics*, vol. 3, no. 4, pp. 409–429, Nov. 2022, doi: 10.3390/opt3040035.
- [3] S. Wörner, S. Becker, S. Küchemann, K. Scheiter, and J. Kuhn, "Development and validation of the ray optics in converging lenses concept inventory," *Physical Review Physics Education Research*, vol. 18, no. 2, p. 020131, Nov. 2022, doi: 10.1103/PhysRevPhysEducRes.18.020131.
- [4] H. P. Kencana, B. H. Iswanto, and F. C. Wibowo, "Augmented reality geometrical optics (AR-GiOs) for physics learning in high schools," *Journal of Physics: Conference Series*, vol. 2019, no. 1, p. 012004, Oct. 2021, doi: 10.1088/1742-6596/2019/1/012004.
- [5] Y.-J. Liao, W. Tarn, and T.-L. Wang, "The effects of an augmented reality lens imaging learning system on students' science achievement, learning motivation, and inquiry skills in physics inquiry activities," *Education and Information Technologies*, Sep. 2024, doi: 10.1007/s10639-024-12973-9.
- [6] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *Arxiv*, 2017, [Online]. Available: <http://arxiv.org/abs/1707.06347>.
- [7] A. Raza, M. A. Shah, H. A. Khattak, C. Maple, F. Al-Turjman, and H. T. Rauf, "Collaborative multi-agents in dynamic industrial internet of things using deep reinforcement learning," *Environment, Development and Sustainability*, vol. 24, no. 7, pp. 9481–9499, Jul. 2022, doi: 10.1007/s10668-021-01836-9.
- [8] T. Haarnoja *et al.*, "Soft actor-critic algorithms and applications," *Arxiv*, 2018, [Online]. Available: <http://arxiv.org/abs/1812.05905>.
- [9] B. Peng, Y. Xie, G. Seco-Granados, H. Wymeersch, and E. A. Jorswieck, "Communication scheduling by deep reinforcement learning for remote traffic state estimation with bayesian inference," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 4, pp. 4287–4300, Apr. 2022, doi: 10.1109/TVT.2022.3145105.
- [10] M. Kim, J.-S. Kim, M.-S. Choi, and J.-H. Park, "Adaptive discount factor for deep reinforcement learning in continuing tasks with uncertainty," *Sensors*, vol. 22, no. 19, p. 7266, Sep. 2022, doi: 10.3390/s22197266.
- [11] V. K. R. Radha, A. N. Lakshmipathi, R. K. Tirandasu, and P. R. Prakash, "The general design of the automation for multiple fields using reinforcement learning algorithm," *Indonesian Journal of Electrical Engineering and Computer Science (IJECS)*, vol. 25, no. 1, p. 481, Jan. 2022, doi: 10.11591/ijeecs.v25.i1.pp481-487.
- [12] H. An and J. Kim, "Design of a hyper-casual futsal mobile game using a machine-learned AI agent-player," *Applied Sciences*, vol. 13, no. 4, p. 2071, Feb. 2023, doi: 10.3390/app13042071.
- [13] T. Goncharenko, N. Yermakova-Cherchenko, and Y. Anedchenko, "Experience in the use of mobile technologies as a physics learning method," *CEUR Workshop Proceedings*, vol. 2732, pp. 1298–1313, 2020.
- [14] Y. B. Bhakti, I. A. D. Astuti, and R. Prasetya, "Four-tier optics diagnostic test (4T-ODT) to identify student misconceptions," in *Advances in Social Science, Education and Humanities Research*, 2023, pp. 308–314.
- [15] B. Mahesh, "Machine learning algorithms - a review," *International Journal of Science and Research (IJSR)*, vol. 9, no. 1, pp. 381–386, Jan. 2020, doi: 10.21275/ART20203995.
- [16] G. Carleo *et al.*, "Machine learning and the physical sciences," *Reviews of Modern Physics*, vol. 91, no. 4, p. 045002, Dec. 2019, doi: 10.1103/RevModPhys.91.045002.
- [17] A. T. Huynh, B. T. Nguyen, H. T. Nguyen, S. Vu, and H. D. Nguyen, "A method of deep reinforcement learning for simulation of autonomous vehicle control," in *International Conference on Evaluation of Novel Approaches to Software Engineering, ENASE - Proceedings*, 2021, vol. 2021-April, pp. 372–379, doi: 10.5220/0010478903720379.
- [18] C. Strannegård *et al.*, "The ecosystem path to AGI," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 13154 LNAI, 2022, pp. 269–278.
- [19] S. S. Mousavi, M. Schukat, and E. Howley, "Deep reinforcement learning: an overview," *Lecture Notes in Networks and Systems*, vol. 16, pp. 426–440, 2018, doi: 10.1007/978-3-319-56991-8\_32.
- [20] Y. Wang, H. He, and X. Tan, "Truly proximal policy optimization," *Proceedings of Machine Learning Research*, vol. 115, pp. 113–122, 2019.
- [21] Y. Savid, R. Mahmoudi, R. Maskeliūnas, and R. Damaševičius, "Simulated autonomous driving using reinforcement learning: a comparative study on Unity's ML-agents framework," *Information*, vol. 14, no. 5, p. 290, May 2023, doi: 10.3390/info14050290.
- [22] H. K. Sagiraju and S. Mogalla, "Application of multilayer perceptron to deep reinforcement learning for stock market trading and analysis," *Indonesian Journal of Electrical Engineering and Computer Science (IJECS)*, vol. 24, no. 3, pp. 1759–1771, 2021, doi: 10.11591/ijeecs.v24.i3.pp1759-1771.
- [23] H. Zhou, Z. Lin, J. Li, Q. Fu, W. Yang, and D. Ye, "Revisiting discrete soft actor-critic," *Arxiv*, 2022, [Online]. Available: <http://arxiv.org/abs/2209.10081>.
- [24] I. Vohra, S. Utrani, A. K. Rao, and V. Dutt, "Evaluating the efficacy of different neural network deep reinforcement algorithms in complex search-and-retrieve virtual simulations," *Communications in Computer and Information Science*, vol. 1528 CCIS, pp. 348–361, 2022, doi: 10.1007/978-3-030-95502-1\_27.
- [25] D. S. Alarcon and J. H. Bidinotto, "Deep reinforcement learning for evtol hovering control," *33rd Congress of the International Council of the Aeronautical Sciences, ICAS 2022*, vol. 7, pp. 5130–5142, 2022.
- [26] H. Shengren, E. M. Salazar, P. P. Vergara, and P. Palensky, "Performance comparison of deep RL algorithms for energy systems optimal scheduling," in *2022 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*, Oct. 2022, vol. 2022-October, pp. 1–6, doi: 10.1109/ISGT-Europe54678.2022.9960642.
- [27] J. Possik *et al.*, "A distributed simulation approach to integrate anylogic and Unity for virtual reality applications: case of COVID-19 modelling and training in a dialysis unit," in *2021 IEEE/ACM 25th International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*, Sep. 2021, pp. 1–7, doi: 10.1109/DS-RT52167.2021.9576149.
- [28] A. Juliani *et al.*, "Unity: a general platform for intelligent agents," *Arxiv*, 2018, [Online]. Available: <http://arxiv.org/abs/1809.02627>.
- [29] A. Cohen *et al.*, "On the use and misuse of absorbing states in multi-agent reinforcement learning," *Arxiv*, 2021, [Online]. Available: <http://arxiv.org/abs/2111.05992>.




- [30] C. Yu *et al.*, “The surprising effectiveness of PPO in cooperative multi-agent games,” *Advances in Neural Information Processing Systems*, vol. 35, 2022.
- [31] A. P. Kalidas, C. J. Joshua, A. Q. Md, S. Basheer, S. Mohan, and S. Sakri, “Deep reinforcement learning for vision-based navigation of UAVs in avoiding stationary and mobile obstacles,” *Drones*, vol. 7, no. 4, p. 245, Apr. 2023, doi: 10.3390/drones7040245.
- [32] M. Hildebrand, R. S. Andersen, and S. Bøgh, “Deep reinforcement learning for robot batching optimization and flow control,” *Procedia Manufacturing*, vol. 51, pp. 1462–1468, 2020, doi: 10.1016/j.promfg.2020.10.203.

## BIOGRAPHIES OF AUTHORS






**Juan Deyby Carlos-Chullo**    was born in Cusco, Peru. He obtained his Bachelor’s degree in Systems Engineering from the National University of San Agustin de Arequipa in 2021. His research interests encompass video games, simulators, artificial intelligence, and usability. Additionally, he has contributed to a project involving augmented reality called ZOODEX, which was affiliated with CiTeSoft (Research Center for Technology Transfer and Software Development R+D+i). He can be contacted at email: jcarlosc@unsa.edu.pe.






**Marielena Vilca-Quispe**    was born in Arequipa, Peru. She is a graduate of Systems Engineering from the National University of San Agustin de Arequipa in 2020. Her research interests encompass video games, augmented reality, artificial intelligence and usability. Additionally, she has contributed to a project involving augmented reality called ZOODEX, which was affiliated with CiTeSoft (Research Center for Technology Transfer and Software Development R+D+i). She can be contacted at email: mvilcaquispe@unsa.edu.pe.



**Whinders Joel Fernandez-Granda**    has a degree in Physics from the National University of San Agustin de Arequipa, Peru. He has a master’s degree in Higher Education and is a teacher in the Academic Department of Physics at UNSA. His research area is the teaching of physics, having published various articles on the subject. He is also the author of books on data processing in experimental physics and the application of physics to various areas of knowledge. He can be contacted at email: wfernandezgr@unsa.edu.pe.



**Eveling Castro-Gutierrez**    Eveling Castro-Gutierrez holds a Ph.D. in Computer Science and is the Coordinator of CiTeSoft at UNSA. She is a faculty member at the National University of San Agustin de Arequipa and a member of IEEE. Additionally, she serves as the Coordinator of Women in Engineering (WIE). She holds a Master’s degree in Software Engineering and has been the principal investigator of projects at CONCYTEC and UnsaInvestiga since 2010. Moreover, she has published research articles in Scopus and Web of Science (WoS) in Computer Vision and computational Thinking. She has been granted copyright, industrial design rights, utility model patents, and invention patents, including the first international patent (PCT), on behalf of UNSA in 2022. She can be contacted at email: ecastro@unsa.edu.pe.