2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Enhancing Reinforcement Learning in Sensor Fusion: A Comparative Analysis of Cubature and Sampling-based Integration Methods for Rover Search Planning

Jan-Hendrik Ewers[1], Sarah Swinton[1], David Anderson[2], Euan McGookin[2], and Douglas Thomson[2]

***Abstract*— This study investigates the computational speed and accuracy of two numerical integration methods, cubature and sampling-based, for integrating an integrand over a 2D polygon. Using a group of rovers searching the Martian surface with a limited sensor footprint as a test bed, the relative error and computational time are compared as the area was sub-divided to improve accuracy in the sampling-based approach. The results show that the sampling-based approach exhibits a 14.75% deviation in relative error compared to cubature when it matches the computational performance at 100%. Furthermore, achieving a relative error below 1% necessitates a 10000% increase in relative time to calculate due to the $\mathcal{O}(N^2)$ complexity of the sampling-based method. It is concluded that for enhancing reinforcement learning capabilities and other high iteration algorithms, the cubature method is preferred over the sampling-based method.**

I. Introduction

Machine Learning (ML) has gained massive popularity in the last decade [1] driven by both the ongoing research into learning algorithms and cheap computational performance [2]. Phenomenon such as the success of generative artificial intelligence [3] have put ML more into the eye of the public than ever before. However, behind the scenes, researchers spend a vast amount of time preparing data for learning. All three major forms of ML require high-quality data for the best performance: supervised learning needs well-labelled data points, unsupervised learning must have noise-free data for labeling, and reinforcement learning has to have access to a good representation of the environment. Studies on the impact of non-systematic noise on training have shown massively degraded performance [4], [5].

While significant progress has been made in developing robust ML algorithms [6], [7], the impact of noise in reinforcement learning remains a critical challenge [8]. In the realm of reinforcement learning noise can be beneficial. Action noise, which is the addition of randomness to the action selected by the algorithm, can be desirable to encourage exploration of the action space. Using Gaussian or Ornstein-Uhlenbeck noise has been shown to significantly improve training across popular reinforcement algorithms such as Soft Actor-Critic [9] or Deep Deterministic Policy Gradient [10] when coupled with scheduled noise reduction [11]. However, this scheduling shows that noise must be minimized to fine-tune the model. In comparison, observation noise in reinforcement learning, akin to poor-quality labelled data for supervised learning, is known to make training unstable [8]. Noise in the reward channel for reinforcement learning is particularly undesirable as it can lead to actions being misinterpreted. [12] highlights multiple scenarios in which the agent may take advantage of a reward function leading to *reward corruption* and a stagnation in learning on a local maxima. Whilst a solution to this was proposed in [12], a much easier approach is to minimize the chances of this happening in the first place by implementing a robust method wth as little noise as possible. Overall noise in ML, when not controllable, is undesirable as it increases the difficulty in successfully learning the policy.

A famously noisy task is numerical integration. From Euler integration to complex modern algorithms, errors are inevitable compared to continuous methods as numerical methods are just approximations. However, reducing these errors is paramount to decreasing noise. It is very common to have simulation step updates within a reinforcement learning environment which require integration in multiple forms. This might be double integrating acceleration to get the position, or integrating a function within a 2D shape. The latter is what this research focuses on. This problem takes the form

$$I(f, H) = \int_H f(\vec{x}) dH \tag{1}$$

where $\vec{x} \in \mathbb{R}^2$, f is the integrand, and H is the collection of s-simplices. In search path planning, the integrand is commonly a Probability Distribution Map (PDM), and H is the path buffered by the sensor footprint r. A widely used method of integration is a sampling-based approach which calculates the integration result through grid sampling of the PDM and then calculating the summation of pixels within H. A different approach uses cubature [13] to achieve a similar result. This relies on the transformation of H into a sum of computable simplices such as triangles or rectangles. *cubpack++* by [14], implement methods for 15 primitive regions that any simplex can be dissected into, like the aforementioned triangle and rectangle. The integration result is then the sum of cubature results for each simplex. Another approach by [15], *r2d2lri*, uses a sequence of embedded lattice rules and claims to result in little computational overhead. Whilst [16] found *r2d2lri* to be faster during

This work was supported by the Engineering and Physical Sciences Research Council, Grant/Award Number: EP/T517896/1-312561-05

[1] Aerospace Sciences Research Division, University of Glasgow, Scotland {j.ewers.1, s.swinton.1}@research.gla.ac.uk

[2] Aerospace Sciences Research Division, University of Glasgow, Scotland {dave.anderson, euan.mcgookin, douglas.thomson}@glasgow.ac.uk

integration over a infinite domain, *cubpack++* proved to be more adaptable to different scenarios without a massive speed penalty. For the rest of this work, *cubpack++* will be used for for its adaptability and adaptive properties.

The task of numerically integrating an integrand over a 2D shape within search planning can be thought of as *"How much probability has been seen by executing this mission?"* In this application, the integrand is a PDM that defines the probability of detection at a given point and the 2D shape is the search path buffered by some quantity to get a polygon. To apply reinforcement learning to this problem, the newly observed probability could be used to calculate the reward function. As previously outlined, this requires a fast and noiseless numerical integration method for good training characteristics.

Search planning typically is applied in situations where exhaustive search (also known as coverage planning) is infeasible due to mission constraints such as limited resources. One such situation where a reduced vehicle lifespan is the limiting factor is searching the Martian surface for items of scientific interest based on a PDM. In 2020, NASA's Mars Mission utilized two robots: the Perseverance rover and the Ingenuity helicopter - opening the door to the use of collaborative robotic teams for planetary exploration [17]. The use of multi-rover teams for planetary exploration missions has been considered in terms of both surface [18] and cave [19] exploration. By using a team of planetary exploration rovers, the data collection capabilities of a mission can be increased due to the extended sensor footprint. Whilst this study was focused on exploratory Mars rovers, search planning has applications within other domains such as in wilderness [20] or marine [21] search and rescue.

In this study, the two aforementioned integration methods are benchmarked against each other to determine the performance differences between them. This will inform future studies on which method is appropriate for their application. A group of rovers searching the Martian landscape with reduced sensor footprints is used as a benchmark. The generated paths are highly irregular due to collision avoidance and inter-rover communications creating a challenging 2D polygon to integrate over. The PDM for this scenario is analogous to the probability distribution of a meteor that has crashed on the surface and that the rovers are now searching for.

The top-level mission planning and environment generation is introduced in Sect. II-A, with modelling and path planning in Sect. II-B. Sampling-based and cubature integration is defined in Sect. II-C.1 and Sect. II-C.2 respectively. The results are presented and discussed in Sect. III with a conclusion being drawn in Sect. IV.

II. Method

A. Mission Planning

The PDM is modelled as a sum of G bivariate Gaussian [22] such that a point on the surface of Mars at coordinate $\vec{x} \in \mathbb{R}^2$ has a probability of containing the meteor p

$$p(\vec{x}) = \frac{1}{G}\sum_{i=0}^{G} \frac{\exp\left[-\frac{1}{2}(\vec{x}-\vec{\mu}_i)^T\vec{\sigma}_i^{-1}(\vec{x}-\vec{\mu}_i)\right]}{\sqrt{4\pi^2\det\vec{\sigma}_i}} \quad (2)$$

where $\vec{\mu}_i$ and $\vec{\sigma}_i$ are the mean location and covariance matrix of the ith bivariate Gaussian respectively. For this study, these values are randomly generated with $G = 4$. [23] used a similar method to approximate a discrete PDM through a Gaussian mixture model which employs Eq. (2) for the final $p(\vec{x})$.

This PDM is then used in conjunction with LHC_GW_CONV [20] to generate 64 mission waypoints with a search radius of 5m. The area is $150\text{m} \times 150\text{m}$ and is thus segmented into a 30×30 grid such that each cell is the size of the search radius.

LHC_GW_CONV is, at its core, a greedy algorithm by employing the local hill climb (LHC) optimization method. This evaluates all eight surrounding cells and moves to the highest-scoring one. If there is a tie, a convolution kernel ω centred around the offending cells is evaluated and the cell with the highest score

$$p^{conv}(\vec{x}) = \omega * p(\vec{x}) = \sum_{i=-1}^{1}\sum_{j=-1}^{1} \omega(i,j)p\left(\vec{x} - \begin{bmatrix} i \\ j \end{bmatrix}\right) \quad (3)$$

is selected. A 3×3 normalized box blur kernel is typically used and is defined as

$$\omega = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad (4)$$

Furthermore, LHC does not handle multi-modal problems well as it attempts to cover one mode completely before being able to move to the next. This often leads to LHC ignoring much more rewarding modes. To solve this, global warming (GW) is introduced. This works by subtracting C from the PDM a l number of times, where $C = p_{\max}/l$ and $p_{\max}$ is the global maxima. This then gives the updated PDM

$$p'(\vec{x}) = \begin{cases} p(\vec{x}) - C, & p(\vec{x}) > C \\ 0, & else \end{cases} \quad (5)$$

which the LHC uses to generate a new path. After all l GW steps are evaluated, they are scored using the original PDM $p(\vec{x})$ and the path with the highest accumulated probability is selected.

Finally, each visited cell is marked a *seen*, and subsequent evaluations of $p(\vec{x})$ at this cell will return 0. This discourages revisiting a position but does not entirely forbid it.

B. Rover Modelling and Path Planning

1) Mars Map Generation: A 3D terrain model has been created using data from the High-Resolution Imaging Science Experiment (HiRISE) on the Mars Reconnaissance Orbiter. The map is a matrix of latitude, longitude, and elevation data, and has a resolution of 0.3m per pixel [24]. For this work, a mission site has been selected from within the Jezero crater. The generated mission site consists of a

600×600 block grid. Fig. 1 shows the 3D terrain model that has been generated, based on previous work at the University of Glasgow [25].

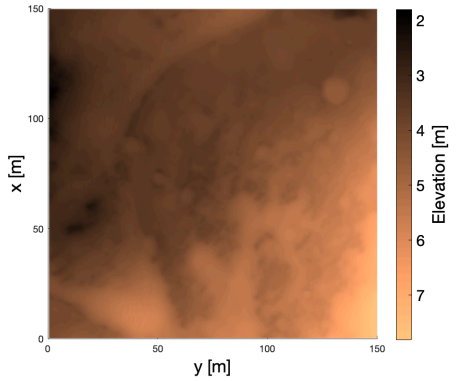

Fig. 1. 3D terrain model of the selected mission site

The 3D terrain model is analysed in terms of the elevation of neighbouring blocking in the grid. A given block inherits the worst-case slope angle, θ, from its eight neighbouring blocks. Three traversability categories have been defined: traversable ($\theta < 10°$), high-risk ($10° \leq \theta \leq 15°$), and impassable ($\theta > 15°$). The thresholds of the three traversability categories have been set in line with the nominal operational limits of the Perseverance rover [26]. Fig. 2 shows the resulting traversability map for the selected mission site.

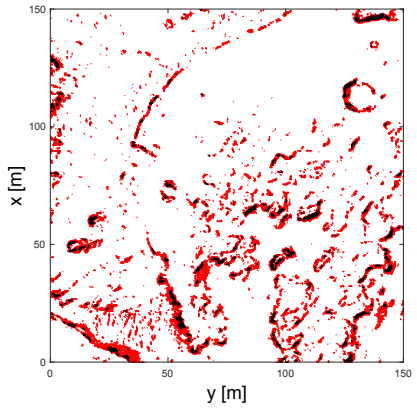

Fig. 2. Traversability analysis of the selected mission site. Traversable terrain is shown in white, high-risk terrain is shown in red, and impassable terrain is shown in black.

2) Rover Model: The robot utilised for analysis and testing in this work is the rocker bogie runt (RBR). This rover has a small form factor ($0.271\text{m} \times 0.251\text{m} \times 0.144\text{m}$), and a six-wheel rocker bogie suspension in line with the baseline mobility characteristics of current planetary exploration robotos [27]. The RBR has six wheels; three wheels on each side, where a front wheel is connected to the rocker, and the middle and rear wheels are connected to the bogie. Each of the six wheels is driven by its own 6V DC motor. Unlike NASA's Mars rovers, which utilise steerable wheels, all RBR wheels are fixed and cannot rotate. Therefore, the differential drive steering method is used. For this work, a team of five rovers is utilised, each with a search footprint diameter of $d_{rover} = 1\text{m}$.

3) Generating Safe Paths: The paths of each member of the rover team must be coordinated such that no collisions occur between them as they traverse paths toward their respective targets. As planetary exploration rovers operate in such remote and hazardous environments, collisions could cause the loss of the rovers involved and severe degradation to the group's data collection capabilities.

For each target point generated by LHC_GW_CONV, a set of safe paths is generated using prioritised planning coordination [28]. Each rover path is given an arbitrary priority number. A path planning attempt is then made for the first rover, using RRT*. A simulation of the rover is run to obtain 4D pose data. The path of the second rover the then planned, and its behaviour is simulated. The 4D pose data for both rovers is compared to check for collisions. If a collision occurs, the path of the lower priority rover is re-planned until a safe path is found. If no collisions occur, the path is saved and the algorithm moves to the next rover. This process continues until all five rovers have safe paths.

C. Path Analysis

Each rover generates a unique path as it executes the mission which is buffered by the search footprint diameter d_{rover}. The total search area H_{total} is then the union of the set of buffered paths H such that

$$H_{total} = \bigcup_{h \in H} h \tag{6}$$

This can be extended to be a function of time t by

$$H_{total}(t) = \bigcup_{h \in H(t)} h \tag{7}$$

where $H(t)$ is the set of buffered paths covered by the rovers up until ts. If $t_{\max}$s is the time taken for the entire mission to be completed, it then follows that $H_{total} = H_{total}(t_{\max})$ and $H_{total} \neq H_{total}(t) \ \forall \ t < t_{\max}$.

1) Sampling-based Integration: Accessing the value of $p(x, y)$ at a given coordinate is trivial, and is as computationally expensive as the definition of $p(x, y)$ itself. A harder task is determining if the point at (x, y) is within $H_{total}(t)$.

A popular method is ray casting along a straight line from a point [29]. If this ray intersects $H_{total}(t)$ (geometry B in the example seen in Fig. 3) an odd amount of times then the point is contained by the buffered search paths.

Another popular method uses the Dimensionally Extended Nine Intersection Matrix (DE-9IM)[30]. This is calculated using

$$DE-9IM(A,B) = \begin{bmatrix} D(A^I \cap B^I) & D(A^I \cap B^B) & D(A^I \cap B^E) \\ D(A^B \cap B^I) & D(A^B \cap B^B) & D(A^B \cap B^E) \\ D(A^E \cap B^I) & D(A^E \cap B^B) & D(A^E \cap B^E) \end{bmatrix} \tag{8}$$

where $D(x)$ is the dimension of x ($D(x) \in (0, 1, 2)$ if $x \neq \emptyset$, and $D(x) = -1$ if $x = \emptyset$), and the superscripts I,

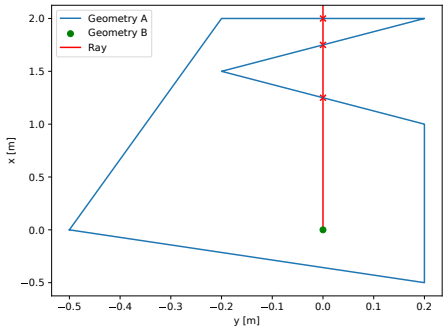

Fig. 3. Ray casting method to check if a point (geometry A) is within a non-convex polygon (geometry B). If the ray originating from A intersects B an odd amount of times (three, in this example), then A is within B.

B, and E denote the interior, boundary, and exterior regions respectively. This matrix can be used to calculate 47 different predicates for A and B to be in. The predicate of interest here is the topological *within*, which is described by the pattern matrix

$$within = \begin{bmatrix} T & * & F \\ * & * & F \\ * & * & * \end{bmatrix} \tag{9}$$

where an entry in the matrix is marked T if $D(x) \in (0, 1, 2)$, F if $D(x) = -1$, and $*$ if $D(x) \in (-1, 0, 1, 2)$.

The DE-9IM approach is 10.05% faster than the ray-casting method when dealing with the high-vertex polygons generated by the buffered paths H and is the method used for the rest of this study.

As touched on in Sect. I, the sampling method is the sum of pixels within the polygon written as

$$I(f, H) = \frac{A}{NM} \sum_{n=0}^{N} \sum_{n=0}^{M} \begin{cases} 0, & (n, m) \text{ not within } H \\ f(n, m), & else \end{cases} \tag{10}$$

where N and M are the dimensions of the rasterisation, A is the area enclosed by the rasterisation, $f(n, m)$ is the integrand (akin to $f(\vec{x})$ from Eq. (1)) which is a function accessing the pixel value at (n, m), and H is a single polygon. This method can be fast with low values of N and M, and accurate with high values alike. However, this method highly suffers from aliasing around the borders of H. Methods to get around the aliasing problem have been tried [31] with major performance penalties. Nonetheless, due to its simplicity, it is prominently used in the literature [23], [32], [33], [34], [35].

2) Cubature Integration: In order to convert the simplex H into something usable for *cubpack++*, it must first be subdivided into a set of the 15 primitive simplices, with the simplest and most ubiquitous of these being the triangle. As H is likely to have many holes and unlikely to be convex, the constrained Delaunay triangulation [36] is used to get the set of triangles T.

At the beginning of the algorithm, the integration estimate $\hat{q}$ and error estimate $\hat{e}$ is evaluated for every triangle in T where the unit triangle is transformed to the given triangle,

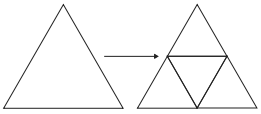

Fig. 4. Subdivision of the unit triangle into four smaller triangles.

while the cubature formula maintains its given degree. To calculate $\hat{q}$, the cubature formula of polynomial degree 13 with 37 points [37] is used. $\hat{e}$ is calculated using several null rules as defined in [38]. If $\hat{E} = \sum \hat{e} > max(\epsilon_a, \epsilon_r |\hat{Q}|)$ (where ϵ_a, ϵ_r are user-defined absolute and relative error tolerances respectively) is above the tolerance set, the triangle with the highest error is subdivided into four smaller triangles as seen in Fig. 4. Each new triangle has its integral and error estimates evaluated, and the four replace the original triangle. This process is continued until either $\hat{E}$ is below the tolerance, or a maximum number of calculations has been exceeded.

III. Results

As discussed in Sect. II-A, LHC is a greedy algorithm that prioritizes local modes. This can be seen in Fig. 5 at $(80, 80)$m where the mission abruptly leaves the saddle between the modes to ascend the local probability mode. The rovers attempt to avoid collision and obstacles, creating small holes in the buffered search path as well as a highly irregular exterior. This creates a challenging benchmark to compare and contrast the methods.

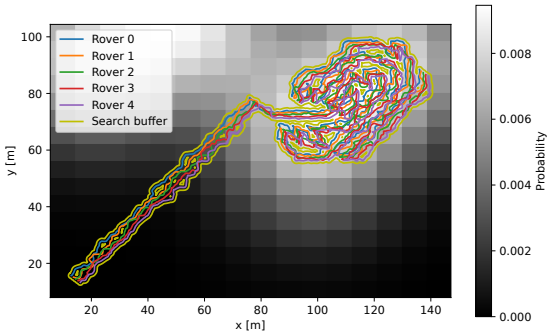

Fig. 5. Rover trajectories over the area given the mission planned by LHC_GW_CONV with a random PDM $p(\vec{x})$. $N = 16$ was selected to emphasize the issue with sampling-based integration methods.

To benchmark the methods, the cubature method is evaluated for each PDM and corresponding paths and the mean of this result (time and integration result) was deemed the baseline. Overall, 105 missions have been simulated using the mission planning from Sect. II-A and the rover simulation from Sect. II-B. In this case, the sampling method is evaluated from $N = 10$ to $N = 300$, and the relative error and relative time are calculated with

$$e_{rel} = \frac{|c - s|}{c} \tag{11}$$

where c and s are the cubature and sampling values respectively.

From Eq. (10), it can be seen that the grid that the sampling method uses grows in $\mathcal{O}(NM)$. For the remainder of this study, $M = N$ for simplicity making this algorithm $\mathcal{O}(N^2)$. This is evident from Fig. 6 where it can be seen that after $N = 16.78$, the cubature method is as fast to compute as the sampling method. After $N = 300$, this value is at 25000%. For reinforcement learning, where billions of steps are taken in the modelled environments during training, this can quickly add days to the learning time.

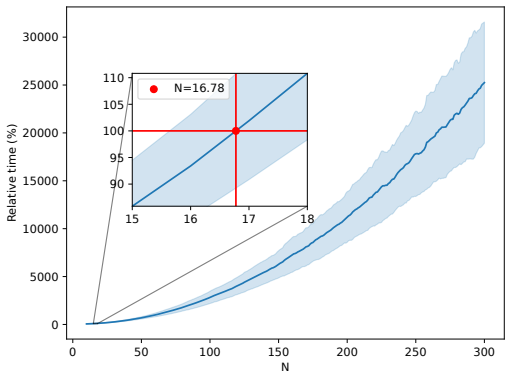

Fig. 6. The relative time as N increases showing a trend towards an exponential increase as $N \to \infty$.

Fig. 7 shows the exponential decay of the relative error as N increases. At the crossing point of $N = 16.78$, the relative error is 14.75%. Depending on the application, this level of accuracy might be deemed sufficient. However, for reinforcement learning, as outlined in Sect. I, any noise is to be mitigated as it can lead to unstable training and therefore diminished results. Even at the high value of $N = 300$, the relative error is still above 0.5%. This size of error is likely to be acceptable, however the computational cost is too high to justify this method.

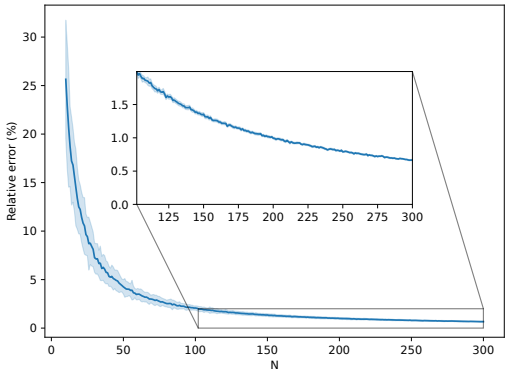

Fig. 7. The relative error as N increases showing a trend towards 0% as $N \to \infty$.

A smaller N might lead to faster computation, the physical meaning of N must be considered. The individual rovers have a sensor diameter of 1m and the area has dimensions $150\text{m} \times 150\text{m}$. To ensure that each grid cell is the same size as that of a single rover's sensor, $N = \frac{150}{1^2} = 150$ which would lead to a 800% relative time difference between the two methods.

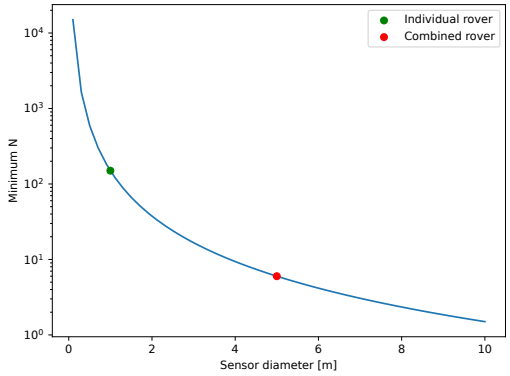

Fig. 8. Minimum N required as a function of the sensor diameter.

The impact of the varying maximum number of coordinates per trial, denoted as N_{coords}, on the relative error was examined to ensure it was not a dependent variable. The null hypothesis posited that N_{coords} does not affect the relative error. Following the application of an ordinary least squares, a p-value of 0.1404 was computed surpassing the level of significance (0.05). Thus, the null hypothesis cannot be rejected indicating that there is no significant evidence to suggest that N_{coords} influences the relative error.

IV. Conclusion

This study presents a comprehensive analysis comparing cubature and sampling-based integration of an integrand over a 2D polygon. An example scenario requiring the search of an area on Mars by a coordinated group of rovers was used as a test bed. These rovers completed a search mission created from the integrand; a random PDM. The union of the resultant buffered paths was then used as the 2D polygon. Using the trajectories from 105 missions, the aggregated data analysis showed that the cubature approach is substantially more accurate and significantly faster. Both of these qualities are important within reinforcement learning, where marginal performance gains can quickly snowball into days of saved training time. Furthermore, these results also translate to embedded devices where the computational budget is typically limited. This could be the case for the likes of onboard navigation systems on a self-sufficient rover similar to the one used in this research.

Overall, the results of this study highlight the superiority of the cubature approach in terms of accuracy and computational efficiency, particularly in scenarios requiring coordinated search missions like those encountered in reinforcement learning and embedded systems for autonomous navigation. However, it's important to acknowledge the limitations of this work. While the focus was on comparing two prevalent integration methods, numerous other techniques

warrant exploration. As highlighted in Sect. I, other potentially faster but less adaptable cubature algorithms exist which could result in an even great performance increase. Additionally, the requirement of a continuous integrand for the cubature method may pose challenges in certain scenarios, though potential solutions using approximations have been discussed. Furthermore, it is important to note that the influence of problem characteristics (deterministic or stochastic, complexity, etc.) on the comparison remains unexplored. By recognizing these limitations, future research can build upon these findings to further enhance integration techniques for a wide range of applications.

REFERENCES

[1] I. H. Sarker, "Machine Learning: Algorithms, Real-World Applications and Research Directions," *SN Computer Science*, vol. 2, no. 3, p. 160, May 2021.

[2] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, vol. 349, no. 6245, pp. 255–260, Jul. 2015.

[3] C. Stokel-Walker and R. Van Noorden, "What ChatGPT and generative AI mean for science," *Nature*, vol. 614, no. 7947, pp. 214–216, Feb. 2023.

[4] D. F. Nettleton, A. Orriols-Puig, and A. Fornells, "A study of the effect of different types of noise on the precision of supervised learning techniques," *Artificial Intelligence Review*, vol. 33, no. 4, pp. 275–306, Apr. 2010.

[5] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, no. 1, pp. 81–106, Mar. 1986.

[6] Y. Guo and W. Wang, "A robust adaptive linear regression method for severe noise," *Knowledge and Information Systems*, vol. 65, no. 11, pp. 4613–4653, Nov. 2023.

[7] R. Fox, A. Pakman, and N. Tishby, "Taming the Noise in Reinforcement Learning via Soft Updates," Mar. 2017.

[8] K. Sun, Y. Zhao, S. Jui, and L. Kong, "Exploring the Training Robustness of Distributional Reinforcement Learning Against Noisy State Observations," in *Machine Learning and Knowledge Discovery in Databases: Research Track*, ser. Lecture Notes in Computer Science, D. Koutra, C. Plant, M. Gomez Rodriguez, E. Baralis, and F. Bonchi, Eds. Cham: Springer Nature Switzerland, 2023, pp. 36–51.

[9] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor," Aug. 2018.

[10] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," Jul. 2019.

[11] J. Hollenstein, S. Auddy, M. Saveriano, E. Renaudo, and J. Piater, "Action Noise in Off-Policy Deep Reinforcement Learning: Impact on Exploration and Performance," Jun. 2023.

[12] T. Everitt, V. Krakovna, L. Orseau, M. Hutter, and S. Legg, "Reinforcement Learning with a Corrupted Reward Channel," Aug. 2017.

[13] J. Lyness and R. Cools, "A Survey of Numerical Cubature over Triangles," *Proceedings of Symposia in Applied Mathematics American Mathematical Society Providence, RI*, vol. 48, Apr. 1994.

[14] R. Cools, L. Pluym, and D. Laurie, "Algorithm 764: Cubpack++: A C++ package for automatic two-dimensional cubature," *ACM Transactions on Mathematical Software*, vol. 23, no. 1, pp. 1–15, Mar. 1997.

[15] I. Robinson and M. Hill, "Algorithm 816: R2d2lri: An algorithm for automatic two-dimensional cubature," *ACM Trans. Math. Softw.*, vol. 28, no. 1, pp. 75–100, Mar. 2002.

[16] M. Hill and I. Robinson, "A Comparison of Strategies for the Automatic Computation of Two-Dimensional Integrals over Infinite Domains," *Numerical Algorithms*, vol. 34, no. 2-4, pp. 325–338, Dec. 2003.

[17] K. Farley, K. Williford, and K. e. a. Stack, "Mars 2020 mission overview," *Space Science Reviews*, vol. 216, no. 142, 2020.

[18] T. Fong, B. D. Allen, and S. Azimi, "Autonomous systems & robotics for lunar surface infrastructure," in *AIAA ASCEND*, 2021.

[19] T. Vaquero, M. Troesch, and S. Chien, "An approach for autonomous multi-rover collaboration for mars cave exploration: Preliminary results," in *International Symposium on Artificial Intelligence, Robotics, and Automation in Space (i-SAIRAS 2018)*, Madrid, Spain, June 2018.

[20] L. Lin and M. A. Goodrich, "UAV intelligent path planning for wilderness search and rescue," *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2009*, vol. 0, no. 1, pp. 709–714, 2009.

[21] T. Furukawa, F. Bourgault, B. Lavis, and H. Durrant-Whyte, "Recursive Bayesian search-and-tracking using coordinated uavs for lost targets," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* Orlando, FL, USA: IEEE, 2006, pp. 2521–2526.

[22] P. Yao, Z. Xie, and P. Ren, "Optimal UAV Route Planning for Coverage Search of Stationary Target in River," *IEEE Transactions on Control Systems Technology*, vol. 27, no. 2, pp. 822–829, Mar. 2019.

[23] L. Lin and M. A. Goodrich, "Hierarchical Heuristic Search Using a Gaussian Mixture Model for UAV Coverage Planning," *IEEE Transactions on Cybernetics*, vol. 44, no. 12, pp. 2532–2544, Dec. 2014.

[24] J. W. Bergstrom, W. Delamere, and A. McEwen, "Mro high resolution imaging science experiment (hirise): Instrument test, calibration and operating contrainst," in *55th International Astronautical Congress of the International Astronautical Federation, the International Academy of Astronautics, and the International Institute of Space Law*, 2004, pp. Q–3.

[25] M. Baxter, "Modelling and visualisation of the mars terrain for rover simulations," Master's thesis, University of Glasgow, 2022.

[26] A. Rankin, M. Maimone, J. Biesiadecki, N. Patel, D. Levine, and O. Toupet, "Driving curiosity: Mars rover mobility trends during the first seven years," in *2020 IEEE Aerospace Conference*. IEEE, 2020, pp. 1–19.

[27] T. Flessa, E. W. McGookin, and D. G. Thomson, "Taxonomy, systems review and performance metrics of planetary exploration rovers," in *2014 13th International Conference on Control Automation Robotics & Vision (ICARCV)*. IEEE, 2014, pp. 1554–1559.

[28] S. Swinton and E. McGookin, "A Novel, RRT* Based Approach to the Coordination of Multiple Planetary Rovers," in *2022 UKACC 13th International Conference on Control (CONTROL)*, Apr. 2022, pp. 88–93.

[29] I. E. Sutherland, R. F. Sproull, and R. A. Schumacker, "A Characterization of Ten Hidden-Surface Algorithms," *ACM Computing Surveys*, vol. 6, no. 1, pp. 1–55, Mar. 1974.

[30] E. Clementini, P. Di Felice, and P. Van Oosterom, "A small set of formal topological relationships suitable for end-user interaction," in *Advances in Spatial Databases*, G. Goos, J. Hartmanis, D. Abel, and B. Chin Ooi, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1993, vol. 692, pp. 277–295.

[31] J.-H. Ewers, D. Anderson, and D. Thomson, "Optimal path planning using psychological profiling in drone-assisted missing person search," *Advanced Control for Applications*, p. e167, Sep. 2023.

[32] M. Morin, I. Abi-Zeid, and C.-G. Quimper, "Ant colony optimization for path planning in search and rescue operations," *European Journal of Operational Research*, vol. 305, no. 1, pp. 53–63, Feb. 2023.

[33] A. Subramanian, R. Alimo, T. Bewley, and P. Gill, "A Probabilistic Path Planning Framework for Optimizing Feasible Trajectories of Autonomous Search Vehicles Leveraging the Projected-Search Reduced Hessian Method," in *AIAA Scitech 2020 Forum*, AIAA. Orlando, FL: American Institute of Aeronautics and Astronautics, Jan. 2020.

[34] D. Ebrahimi, S. Sharafeddine, P.-H. Ho, and C. Assi, "Autonomous UAV Trajectory for Localizing Ground Objects: A Reinforcement Learning Approach," *IEEE Transactions on Mobile Computing*, vol. 20, no. 4, pp. 1312–1324, Apr. 2021.

[35] L. Hong, Y. Wang, Y. Du, X. Chen, and Y. Zheng, "UAV search-and-rescue planning using an adaptive memetic algorithm," *Frontiers of Information Technology & Electronic Engineering*, vol. 22, no. 11, pp. 1477–1491, Dec. 2011.

[36] L. P. Chew, "Constrained delaunay triangulations," *Algorithmica*, vol. 4, no. 1-4, pp. 97–108, 1987.

[37] S. Wandzurat and H. Xiao, "Symmetric quadrature rules on a triangle," *Computers & Mathematics with Applications*, vol. 45, no. 12, pp. 1829–1840, Jun. 2003.

[38] J. Berntsen and T. O. Espelid, "Algorithm 706: DCUTRI: An algorithm for adaptive cubature over a collection of triangles," *ACM Transactions on Mathematical Software*, vol. 18, no. 3, pp. 329–342, Sep. 1992.