Real-Time Navigation using Virtual Magnetic Fields

Majda Moussa, Giovanni Beltrame

Abstract— **Humans and animals have learned or evolved to use magnetic fields for navigation. Knowing how to model and estimate these fields can be used for motion planning. However, computing the propagation of electromagnetic fields in a given environment requires solving complex differential equations with advanced numerical methods, and therefore it is not suitable for real-time decision making. In this paper, we present a real-time approximator for Maxwell's equations based on deep neural networks that predicts the distribution of a virtual magnetic field. We show how our approximator can be used to perform autonomous 2D navigation tasks, outperforming state-of-the-art navigation algorithms, ensuring completeness, and providing a near-optimal path up to 200 times per second without any post processing stage. We demonstrate the effectiveness of our method with physics-based simulations of an unmanned aerial vehicle, an autonomous car, as well as real-world experiments using a small off-road autonomous racing vehicle. Furthermore, we show how the approach can be applied to multi-robot systems, video game technology, and can be extended to 3D problems.**

I. Introduction

Autonomous vehicles have sparked a wide interest in recent years, and pushed the many advances in software, artificial intelligence and machine learning. Applications such as delivery, inspection, and monitoring require intelligent robots that are capable of navigating autonomously to their destinations, detecting and avoiding obstacles in real-time[1]. This type of path planning is a challenging problem, especially for systems that operate in a highly dynamic environment. Several start-ups (e.g. Embodied Intelligence and Realtime Robotics) have resorted to dedicated hardware solutions to solve the path planning problem in real-time, with a market estimated in the billions of dollars [2]. Although the path planning problem has some practically demonstrated solutions such as Waymo self-driving cars [3] and the Skydio drone [4], these implementations are not necessarily optimal, they require precise external positioning (e.g. GPS), high computational performance, and have a high energy cost.

In the last decade, a massive number of real-time path planning algorithms have been proposed, studied, and discussed [5], [6]. The most common approaches are graph-based methods [7], sampling-based planners [8], artificial potential fields [9], and heuristic methods [10].

In graph methods, the environment is represented by a grid where the robot can only move between adjacent grid cells and can take position only in discrete locations (grid nodes). Standard solutions in this category include Dijkstra's algorithm and A* [7]. A major drawback of graph methods is that they become computationally expensive with larger environments and higher resolutions. To address this problem, many solutions use a sampling approach: originally, Rapidly-Exploring Random Trees (RRT) [11] exploits randomness to quickly explore a large search space with iterative refinement. Variants of RRT added probabilistic guarantees that a path would be found if it existed – e.g. RRT* [12], RRT*-smart [13] – as well as real-time performance – RRT^X [14], ERRT [15], CL-RRT [16], and RT-RRT* [17]. However, these solutions are far from optimal: guarantees are only asymptotic (i.e. assuming we have infinite computation time) and real-time solutions trade off accuracy by interleaving planning and acting on partial plans. More recent works [18], [19] resort to neural networks to generate paths using graph-based planners as "training experts". Although these approaches enable fixed-time path generation, they have the same limitations as the algorithms they try to imitate.

To solve this issue, we found inspiration from the natural world. Evolution has addressed the problem using existing features of our planet: animals have long since learned to take advantage of Earth's magnetic field to navigate their environment. Various species, ranging from birds and mammals to reptiles and insects, are able to sense the magnetic field and, along with sunlight, use it for navigation [20]. Wu and Dickman [21] show that some neurons in animal brains encode information on the magnetic field's direction, intensity and polarity. This reveals that the animals have a magnetic sense that provides them with an internal global positioning system and a magnetic compass for directional heading. This natural skill allows animals to navigate towards their destination without getting lost.

In essence, knowing how the magnetic field propagates, humans and animals alike can use the field as a map. For instance, Earth's magnetic field is well modeled and studied [22], [23], [24], and currently used for satellite navigation [25]. This is possible because magnetic field model [26], [27] *does not have local maxima* [28] and *its gradient can be used for global navigation*.

While we can use an existing magnetic field (e.g. Earth's) for outdoor navigation (e.g. satellite; aircrafts), we surmise that a robot could navigate any kind of environment by constructing a model of the magnetic field, i.e. a *virtual magnetic field*, and following its gradient. By properly selecting the conditions in which the virtual field propagates, one can have no local maxima, which makes the gradient path *optimal* and *guarantees completeness* –that is, if a path exists, it will be found. This operation requires solving Maxwell's equations to determine the value of the virtual magnetic field across a simplified model of the environment. However,

Majda Moussa and Giovanni Beltrame are with the Department of Computer and Software Engineering, École Polytechnique de Montréal, Québec, Canada giovanni.beltrame@polymtl.ca

solving Maxwell's equations is complex and computationally expensive, and usually demands numerical simulations such as finite elements or finite differences methods [29]. The computational resources needed to run these simulations in real-time far exceeds the capabilities of a mobile robot or autonomous vehicle. Hussein and Elnagar [30] attempted the use of a magnetic potential field for path planning using the finite differences method. Their work showed promise, but they were not able to achieve real-time performance.

In this paper, we present a data-driven solver of Maxwell's equations that can compute a *virtual* magnetic field in any 2D environment up to 200 frames per second, making our solution suitable for path-planning in highly-dynamic and time-varying environments. We use an auto-encoder convolutional neural network exclusively trained in simulation that can predicts the magnetic field distribution in unknown environments with high accuracy. We show how this method is readily applicable to many navigation tasks such as outdoor/indoor path planning and gaming.

II. Approach

Our main result is a learning-based path planning system that provides near-optimal results for 2D environments. Interestingly, our path planner is not trained on existing path planners or other kinds of oracles, but rather on the well-known physics of the electromagnetic field.

Electrical conductivity, the ability of a medium to carry electrical currents, is strongly related to the magnetic distribution in a given environment [31]. Intuitively, one can describe an environment as a conductivity map to compute the propagation of the magnetic field. However, the relation between the environment's conductivity and the propagation of the magnetic field is often indirect and strongly non-linear [32]. Convolutional neural networks, CNNs [33], are known to be very powerful approximators for arbitrary non-linear functions [34]. Often used in computer vision, CNNs work by learning patterns and making decisions at the level of pixels, based on local spatial information. A vast range of computer vision problems, the likes of image segmentation [35], [36], [37] and pixelwise prediction [38], have been solved using CNNs.

We extend this idea and create a deep learning framework that approximates the relationship between the conductivity of an environment and its corresponding magnetic field distribution. Given that the behavior of the magnetic field is well understood and easily simulated, we build an (arbitrarily) large dataset of conductivity maps and their corresponding field distributions using a commercial simulator COMSOL[1]. We train a CNN on this dataset and obtain an approximator of the magnetic field distribution. We use the magnetic field to generate a feasible path to any set point in the environment. The overall framework of our method is shown in Fig. 1.

The generation of conductivity maps is a key feature: we use 2D images where each pixel represents the conductivity value of a corresponding discrete area of the environment

[1]https://www.comsol.com/

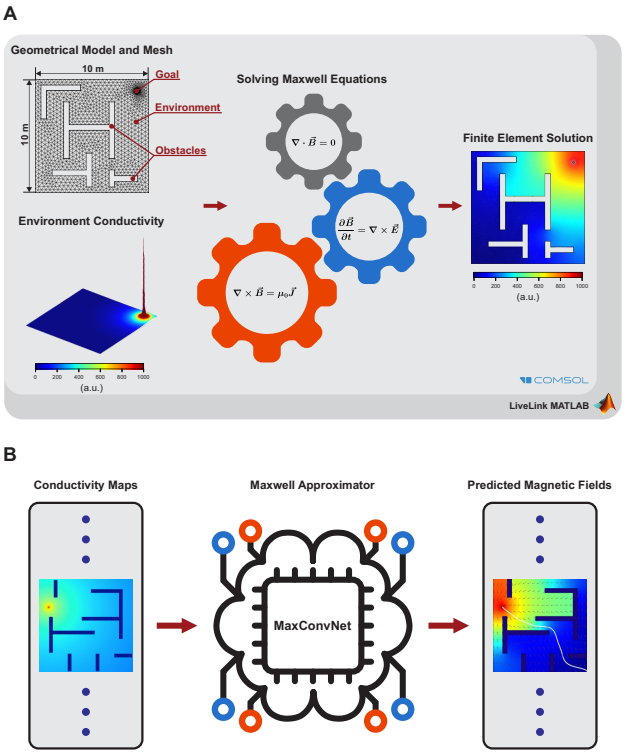

Fig. 1. **Deep learning framework for solving Maxwell's equations:** (A) Schematic representation of the data collection process (conductivity data maps and the corresponding magnetic field distributions). (B) The conductivity images from multiple environments are used to train a deep neural network using the error backpropagation algorithm. The trained network accurately predicts the corresponding magnetic solution when environment conductivity images are shown.

(of arbitrary size, chosen according to the desired accuracy). Studying the simplest practical form of this problem, we encode *obstacles* (anything that cannot be traversed) with zero conductivity ($\sigma_o = 0$), while we assign the highest conductivity to a *goal* location (σ_g). For the rest of the environment, we assign conductivity values to each pixel according to its distance to the goal. For every pixel $p(x, y)$, the conductivity $\sigma_e(x, y)$ is defined as:

$$\sigma_e(x,y) = \frac{c}{\sqrt{(x-x_g)^2 + (y-y_g)^2}} \tag{1}$$

where x_g and y_g are the goal coordinates and c is a fixed parameter, the environment conductivity constant.

We transform the conductivity map into a triangular mesh (the geometrical model and mesh in Fig. 1A) and compute the magnetic field distribution via finite elements [29]. Please refer to Sections 1 and 2 in the supplementary material [39] for more details about magnetic field computation.

III. Training

We train our neural network, shown in Fig. 2 and called MaxConvNet, as a supervised pixelwise regressor using the conductivity data as input and the magnetic distributions as their corresponding ground truth (see Fig. 1B). Pixelwise regression is similar to the well-known pixelwise classification

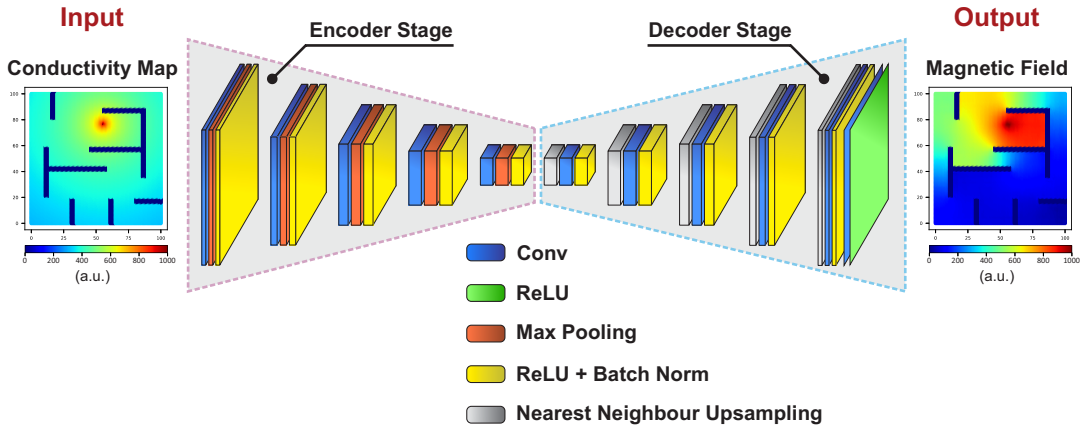

Fig. 2. **MaxConvNet structure**: When a conductivity image of an environment is taken as input, the network first processes the image through an encoder stage (five rounds of convolution, ReLU nonlinearity, and pooling layers). Then, a decoder stage follows with 5 rounds of up sampling, convolution, and ReLU nonlinearity. Dropout and L2 regularization are used to avoid overfitting. We compute ground truth magnetic field distributions using a commercial simulator (COMSOL) to obtain the loss function. See supplementary material [39] Section 3 and Table S2 for details on the architecture.

problem [40], however, each pixel is assigned to a continuous value.

A well-trained neural network learns the general relation between input and output data and can accurately predict never-seen-before inputs. We tested MaxConvNet on new, randomly generated environments and compared the prediction with the ground truth magnetic field distribution per pixel. Fig. 3A depicts the loss curves of both training and testing phases: the testing loss curve matches the slope of the training curve, which means that the model is able to generalize well on the testing dataset.

We use a relative error metric $err_s(x, y)$ at the pixel level to assess the accuracy of the model for each sample input s:

$$err_s(x,y) = \frac{|\phi_s(x,y) - \hat{\phi}_s(x,y)|}{\phi_s(x,y)} \tag{2}$$

where $\hat{\phi}_s$ is the predicted magnetic field for the input sample s and ϕ_s is the ground truth generated with a commercial simulator. The average relative error per-pixel for a batch of n samples is defined as the mean value of the relative per-pixel error. In logarithmic scale (decibels):

$$err_{avg}(x,y) = \left. \frac{\sum_s err_s(x,y)}{n} \right|_{dB} \tag{3}$$

Fig. 3B shows the relative error of MaxConvNet for the testing dataset: less than -25dB for almost every pixel. This shows that the model is able to learn the relationship between magnetic field and conductivity, and is able to generalize to unseen samples. An attentive reader might notice that the average relative error around the center of the map is lower (i.e., below -35dB) than the error at its boundaries. This can be explained by the structure of our dataset, which places most of the obstacles far from the map boundaries, meaning that MaxConvNet is especially good at determining the magnetic fields around obstacles.

Fig. 3C shows the performance of MaxConvNet for three sample scenarios with increasing complexity. Fig. 3C.i is a simple case where the environment is free of obstacles. The

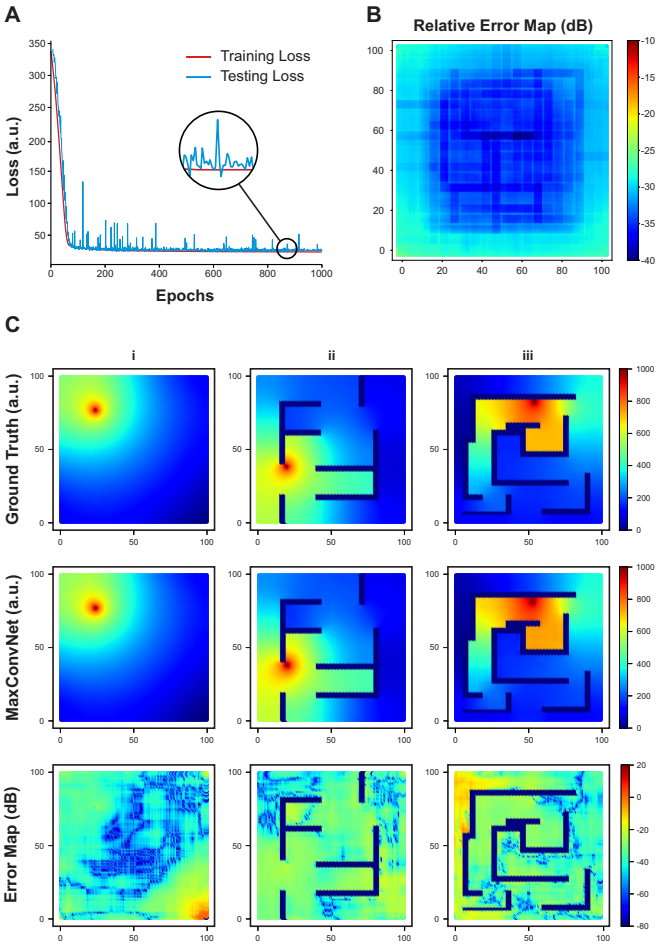

Fig. 3. **Performance of MaxConvNet:** (A) We use the mean square error to train MaxConvNet. The testing loss decreases conjointly with the training loss. The network is able to generalize to unseen environments. (B) We calculate the average relative error per pixel for the testing set (4000 samples). The average relative error is under -25dB per pixel. (C i to iii) depict three environment scenarios with increasing complexity. MaxConvNet (middle) is able to reproduce the ground truth (top). The error map (bottom) shows the relative error per pixel (less than -20dB for almost all pixels in the map). The x-y axes of each sub-figure refer to the abstract positions of pixels in the maps i.e., in [0, 100] each dimension.

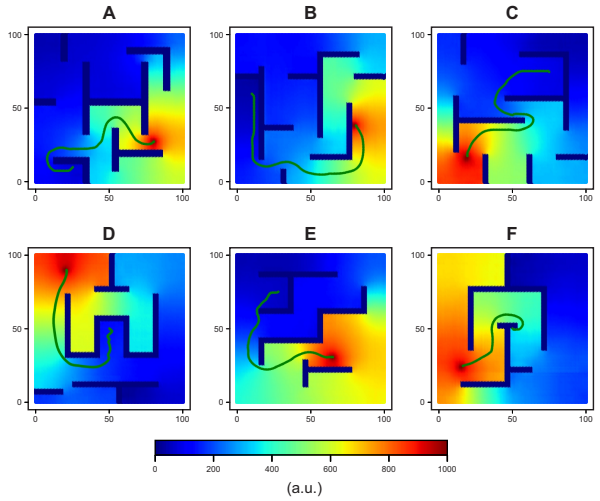

Fig. 4. **Gradient-based path:** (A to F) The gradient information of machine-predicted magnetic fields can be used to compute a continuous path from a starting point to a goal position. The predicted magnetic distribution is agnostic to local maxima problems as it is based on Maxwell's equations for the magnetic field. The x-y axes of each sub-figure refer to the abstract positions of pixels in the maps i.e., in [0, 100] each dimension.

relative error map (bottom) shows that MaxConvNet is able to reproduce the ground truth (top).

Fig. 3C.ii and Fig. 3C.iii present more complex cases where the environment is cluttered with obstacles. The error maps show an error rate less than -20dB for almost all pixels in the scenes. Overall, the model is able to learn the obstacle positions (i.e. with an error rate less that -80dB) and predict the correct magnetic field distribution.

IV. Path Planning

To compute a feasible path from the predicted distribution, we have implemented Adam's algorithm [41]. Assuming a robot acting in discrete time steps, at each time step the algorithm takes the gradient of the predicted field as well as the current position of the robot and returns a path to the goal (see supplementary material [39], Section 4 and Table S3 for details). The algorithm does not need any information about the goal since it is already encoded in the gradient information given as input.

Fig. 4 shows the computed path for some scenarios. The length of the path depends on the learning rate parameter that controls how much to change the position of the robot in response to the estimated gradient each time the path is updated towards the goal. Fig. 5 shows that the higher the learning rate the longer the path. An excessively large learning rate results in a coarse exploration of the gradient information and consequently generate a sub-optimal set of positions towards the goal. A too small learning rate could, in turn, lead to a long optimization process that might get stuck. To avoid these issues, we set the learning rate to 0.01.

Fig. 6 compares the paths computed on 4000 machine-predicted distributions and their ground truth in terms of path length. The ground truth path is produced from a strictly monotonic gradient [42], implying it is the optimal path [43].

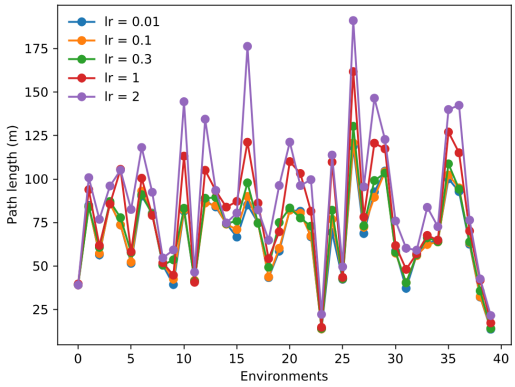

Fig. 5. Impact of Adam's learning rate on the path length. The higher the learning rate the longer the path

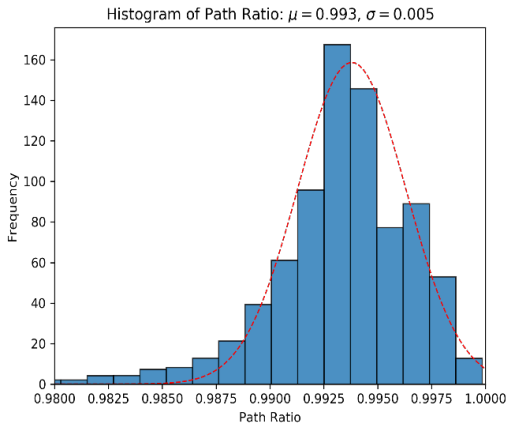

Fig. 6. The ratio between the path computed using the real magnetic distribution and the one computed using the machine predicted distribution: the length ratio variable follows a Normal Distribution N(μ=0.993, σ=0.005), The median value is 0.993.

The average ratio between optimal and the MaxConvNet-computed path length is 0.993$\pm$ 2.97e-05 which confirms the high accuracy of MaxConvNet in predicting the correct magnetic distributions. The main advantage of our approach is that it produces complete paths without local maxima, a problem present in other potential fields methods [9] that can cause a robot to get stuck. Fig. 4 shows some classic examples where planners can get stuck, but where MaxConvNet can find a path to the goal location.

V. Experimental Results

We used MaxConvNet for the autonomous navigation of flying and wheeled robots. We used realistic models of the Parrot AR.Drone 2.0 quadcopter and of a sport utility vehicle (SUV) in a high-fidelity visual and physical simulator, AirSim [44]. Fig. 7 depicts the simulation setup. Fig. 9A shows two simulated scenarios where a quadcopter and an SUV with a realistic LiDAR model, controlled by MaxConvNet, are making their way to a goal position while avoiding obstacles. Fig. 9A also presents the reconstructed maps of the virtual environments.

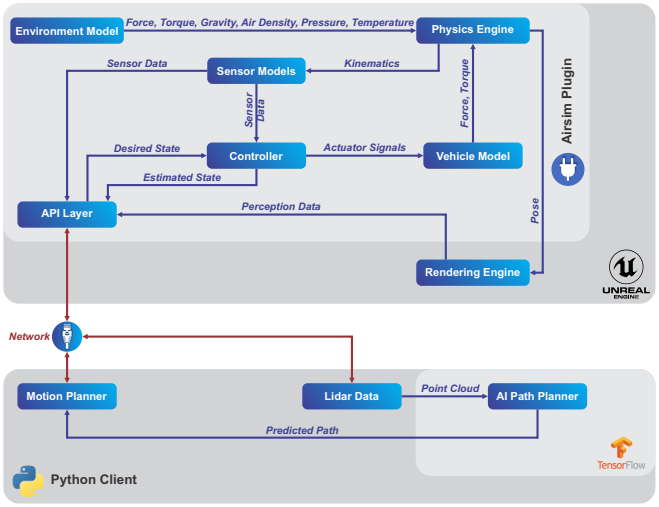

Fig. 7. AirSim physics-based simulation: AirSim server computes the physics of the vehicle model and simulate the sensors and the environment to mimic real-world behaviors (such as sensor drift, position errors). On the client side, MaxConvNet runs inside the AI path planner module at 100 fps. The AI path planner requests lidar data and estimated kinematics from the server and produces a feasible path. The motion planner converts the computed path into velocity commands using a modified version of Reynold's algorithms.

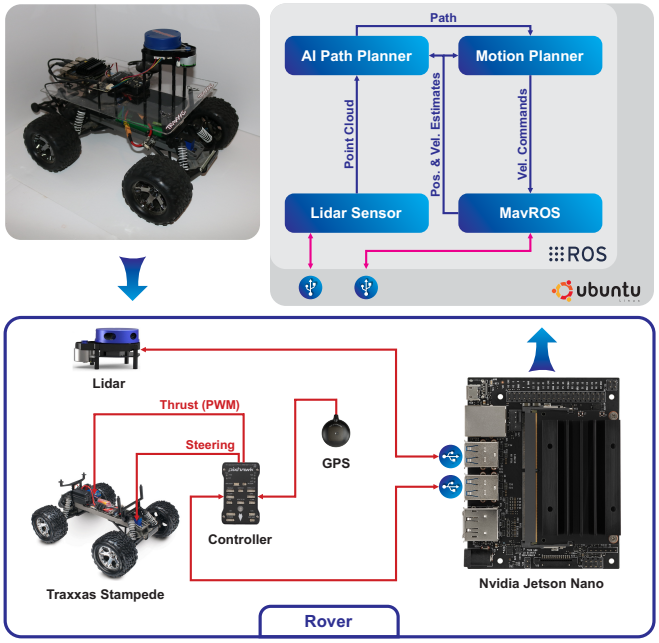

Fig. 8. **Experimental setup:** We set up a wheeled platform using Traxxas stampede frame with a servo motor in the front of the vehicle for steering and a throttle motor at the back connected to the ESC (electronic speed control VXL). We used YdLidar X4 to collect 2D environment data and run our path planner on an NVIDIA Jetson Nano.

Moreover, we have implemented a game-based scenario where an AI tries to catch a human-controlled player using MaxConvNet in a highly dynamic environment. The position of the player is encoded as the goal and used along with the obstacle information by the AI to predict a feasible path to track the player (see the attached video for an example).

Furthermore, we use MaxConvNet running on a NVIDIA Jetson Nano computing board to control a Traxxas Stampede rover equipped with Ydlidar X4 lidar and a PixRacer autopilot (see Fig. 8) for the outdoor experiments in Fig. 9B. The autopilot uses GPS information along with its inertial measurement unit to estimate its current position.

A. Comparison with RRT* and A*

We compare MaxConvNet, RRT* [13], and A* [45] in terms of path cost and runtime. A* provides an optimal path, and performs well on small maps. However, its runtime depends on the map and the position of the initial and goal locations, and A* scales exponentially with map size, which makes it less than ideal for real time operation. Some real-time variants exist, but they compute partial paths that are optimal only in a static environment. RRT* is widely used in robotics, can be computed in fixed time, and tends towards the optimal solution.

Qualitatively, Figs. 10A and 10B show two scenarios where the path computed by MaxConvNet is shorter and smoother than the one produced by RRT*. The quality of the path generated by RRT-based approaches depends on the number of nodes (a parameter). The denser the tree, the smoother the path. However, this significantly slows down the algorithm. In Fig. 10A.ii, RRT* takes 30 seconds to compute a 125-meter path using 1000 nodes, while using 3000 nodes in Fig. 10A.iii leads to a path length of 103m, but at the expense of 217 seconds of computation time. For the same scenario, MaxConvNet was able to compute a shorter path in only 0.07 seconds, and similarly for Fig. 10B. Overall, MaxConvNet can compute a shorter path 400 times faster than RRT* with 1000 nodes.

Quantitatively, we compare RRT* (with 300 and 1000 nodes), MaxConvNet, and A* over 10 different environments running on an Intel i7 M620 running at 2.67GHz. Fig. 10C shows that the path length for MaxConvNet is *always* shorter than RRT*, very close to A*, and Fig. 10D shows that the model has a constant, very high speed, 400 times faster than RRT* with 1000 nodes, and up to 20 times faster than A* (times include the full pipeline, from sensor reading to path generation). Note that, as shown in the following, MaxConvNet can be run on GPUs, increasing the speed by orders of magnitude (see Table I), which is more difficult to do for RRT* and A*.

We can conclude that MaxConvNet is more efficient than state-of-the-art sampling-based approaches: it is faster, finds near-optimal paths, has constant runtime, and its results do not depend on scene complexity or path length.

B. Performance on Different Computers

We have conducted a statistical study to investigate the performance capabilities of MaxConvNet on different computers. We used a set of 1000 input environments on the most common Nvidia Graphical Processing Units GPUs–namely Jetson NANO, Jetson TX2, GTX 1050, GTX 1060, GTX 1080 Ti–, as well as on a Raspberry Pi 3 and a standard Intel i7-8750H. Table I depicts the minimum, maximum and mean values of the execution time of our model on the

TABLE I

MAXCONVNET'S FULL PIPELINE EXECUTION TIME ON DIFFERENT PLATFORMS ON A SET OF 1000 SAMPLES

Platforms		**Nvidia GPUs**					**CPUs**	
		Jetson Nano	Jetson TX2	GeForce GTX1050	GeForce GTX 1060	GeForce GTX 1080 Ti	RPI 3	Intel i7
MaxConvNet Execution Time (s)	**Min**	0.0920	0.0455	0.0132	0.0086	0.0045	1.901	0.039
	Max	0.1306	0.298	0.0199	0.019	0.006	2.885	0.081
	Mean	0.1065	0.074	0.0164	0.0093	0.005	1.982	0.033

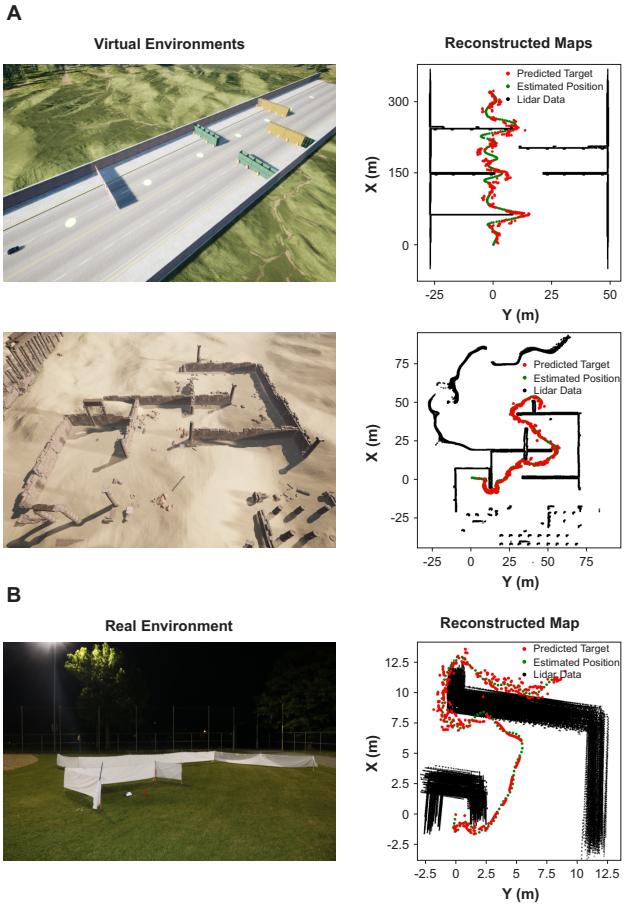

Fig. 9. **AirSim physics-based simulations and real-world experiment:** (A) presents two virtual environments in AirSim: MaxConvNet predicts the next path to follow at 100 fps. The lidar data captured by the drone and the ground vehicle as well as the predicted path and the estimated positions has been superimposed to reconstruct the environments. (B) presents a real-world environment. MaxConvNet predicts the next path to follow at 7 fps (limited by the lidar performance). Obstacle data is collected using a Ydlidar X4. Lidar data from different frames are superimposed along with predicted future positions and estimated positions to reconstruct the map.

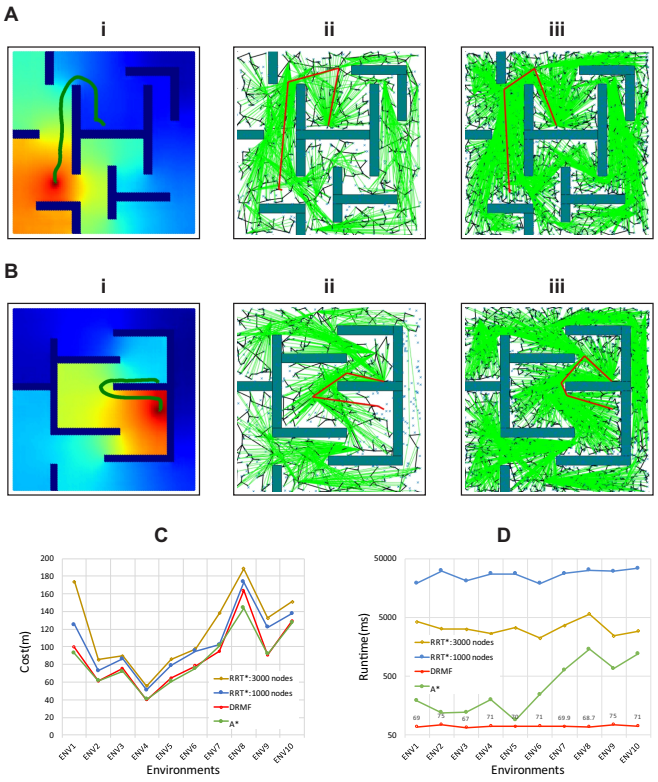

Fig. 10. **MaxConvNet vs RRT*:** (A i) MaxConvNet runtime = 0.07s, cost = 99.27m, (A ii) RRT* number of nodes 1000, runtime = 30.12s, cost 122.01m, (A iii) RRT* number of nodes 3000, runtime = 217s, cost 111.33m, (B i) MaxConvNet runtime = 0.07s cost 75.17m, (B ii) RRT* number of nodes 1000, runtime = 27.45s, cost 104.24m, (B iii) RRT* number of nodes 3000, runtime = 329s, cost 84.64m. We compared MaxConvNet and RRT* in 10 different environment (C) Cost comparison, (D) Runtime comparison.

aforementioned platforms. Results show that MaxConvNet can run at up to 15 Hz on an embedded computer (the Jetson TX2) and up to 200 Hz on an advance GPU (the GTX 1080 Ti). It is worth noting that our model prototype is implemented in Python, and we expect better performance with a more optimized implementation.

DISCUSSION AND CONCLUSIONS

We propose an AI based approximator of Maxwell's equations that enables their use in applications with tight resource constraints. We also show how this new strategy can be applied to robots to perform navigation tasks. This application is of wide interest as it allows a low-cost navigation in a highly dynamic and unknown environment, and ensures optimal paths in complex real world scenarios with static and dynamic obstacles.

MaxConvNet works by automatically recognizing the distribution of a virtual magnetic field in a 2D environment defined as a conductivity map. On top of MaxConvNet, we use an optimization algorithm to compute a feasible path given the gradient information of the magnetic distribution. The attentive reader might have realized that the path generated by MaxConvNet does not take into account the kinematic constraints of the robot, but these can be taken into account when executing the optimization algorithm for the generation of the path.

Results show that MaxConvNet makes correct predictions and avoids the problem of local maxima. In addition, MaxConvNet outperforms state-of-the-art navigation algorithms by ensuring completeness and providing an optimal path in real time. Additionally, the present method can be readily extended to other navigation problems, such as flocking or pursuit (see attached video).

The neural network presented in this paper can be an inspiration to understand how the problem of visual path-planning in a dynamic, unknown environment can be solved efficiently by a neural architecture, given a simple model of the environment. Most importantly, the environment model and conductivity could be modified to account for the *dynamics* of the robot being considered to achieve specific trajectories such as those needed for drone or car racing. For example, adding magnetic shells to the obstacles can shape the magnetic field to have smoother turns, thus modelling limitations in turn radius for the vehicle. These implicit optimizations likely come at *no performance cost*: the map, model, and inference time stay the same, and only require additional training. We plan to study magnetic shaping in future work.

The low computational cost of MaxConvNet allows implementations on small, low-cost, single-board computers as the Jetson Nano or the Raspberry Pi, paving the way for pervasive robot applications. In addition, one can train the system using *exclusively* simulation data leading to straightforward implementation on physical robots.

MaxConvNet can be extended to 3D to enable a more efficient navigation for flying and under-water robots. The magnetic field distribution of a 3D environment computed using our physics-based model and the gradient information that can be used to compute a free path from any starting position to the goal (see supplementary material [39], Fig. S2). In addition, the network can be trained with an arbitrarily large number of environment data with different obstacle shapes and the performance can be further improved increasing the environment resolution (i.e. 512x512).

REFERENCES

[1] F. Bonin-Font, A. Ortiz, and G. Oliver, "Visual navigation for mobile robots: A survey," *Journal of intelligent and robotic systems*, vol. 53, no. 3, pp. 263–296, 2008.

[2] D. Sorin and G. Konidaris, "Enabling faster, more capable robots with real-time motion planning," 2018, https://spectrum.ieee.org/automaton/robotics/robotics-software/enabling-faster-more-capable-robots-with-real-time-motion-planning, last accessed on 2019-10-14.

[3] C. Fisher, "Waymo's fully-automated shuttles are picking up riders around phoenix," 2019, https://www.engadget.com/2019/10/28/waymo-rider-only-autonomous-taxis-phoenix/, last accessed on 2019-10-24.

[4] F. Jon, "Skydio's station lets self-flying drones work around the clock," 2019, https://www.engadget.com/2019/10/16/skydio-2-dock-autonomous-drone/, last accessed on 2019-10-24.

[5] M. N. Rastgoo, B. Nakisa, M. F. Nasrudin, A. Nazri, and M. Zakree, "A critical evaluation of litterature on robot path planning in dynamic environment," *Journal of Theoretical & Applied Information Technology*, vol. 70, no. 1, 2014.

[6] M. Mohanan and A. Salgoankar, "A survey of robotic motion planning in dynamic environments," *Robotics and Autonomous Systems*, vol. 100, pp. 171–185, 2018.

[7] D. González, J. Pérez, V. Milanés, and F. Nashashibi, "A review of motion planning techniques for automated vehicles." *IEEE Trans. Intelligent Transportation Systems*, vol. 17, no. 4, pp. 1135–1145, 2016.

[8] M. Elbanhawi and M. Simic, "Sampling-based robot motion planning: A review," *Ieee access*, vol. 2, pp. 56–77, 2014.

[9] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Autonomous robot vehicles*, pp. 396–404, 1986.

[10] T. T. Mac, C. Copot, D. T. Tran, and R. De Keyser, "Heuristic approaches in robot path planning: A survey," *Robotics and Autonomous Systems*, vol. 86, pp. 13–28, 2016.

[11] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[12] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[13] F. Islam, J. Nasir, U. Malik, Y. Ayaz, and O. Hasan, "RRT*-smart: Rapid convergence implementation of RRT* towards optimal solution," *Mechatronics and Automation (ICMA), 2012 International Conference on*, pp. 1651–1656, 2012.

[14] M. Otte and E. Frazzoli, "Rrt^X: Real-time motion planning replanning for environments with unpredictable obstacles," *Algorithmic Foundations of Robotics XI*, pp. 461–478, 2015.

[15] J. Bruce and M. M. Veloso, "Real-time randomized path planning for robot navigation," *Robot Soccer World Cup*, pp. 288–295, 2002.

[16] Y. Kuwata, J. Teo, G. Fiore, S. Karaman, E. Frazzoli, and J. P. How, "Real-time motion planning with applications to autonomous urban driving," *IEEE Transactions on Control Systems Technology*, vol. 17, no. 5, pp. 1105–1118, 2009.

[17] K. Naderi, J. Rajamäki, and P. Hämäläinen, "Rt-rrt*: a real-time path planning algorithm based on rrt," *Proc. of the 8th ACM SIGGRAPH Conference on Motion in Games*, pp. 113–118, 2015.

[18] M. J. Bency, A. H. Qureshi, and M. C. Yip, "Neural path planning: Fixed time, near-optimal path generation via oracle imitation," *arXiv preprint arXiv:1904.11102*, 2019.

[19] T. Dang, S. Khattak, C. Papachristos, and K. Alexis, "Anomaly detection and cognizant path planning for surveillance operations using aerial robots," *International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 667–673, 2019.

[20] D. Barrie, *Supernavigators: Exploring the wonders of how animals find their way*. The Experiment, 2019.

[21] L.-Q. Wu and J. D. Dickman, "Neural correlates of a magnetic sense," *Science*, vol. 336, no. 6084, pp. 1054–1057, 2012.

[22] J. Davis, "Mathematical modeling of earth's magnetic field," *Technical Note, Virginia Tech, Blacksburg*, 2004.

[23] S. Edwards, C. Parnell, L. Harra, J. Culhane, and D. Brooks, "A comparison of global magnetic field skeletons and active-region upflows," *Solar Physics*, vol. 291, no. 1, pp. 117–142, 2016.

[24] W. H. Campbell, *Earth magnetism: a guided tour through magnetic fields*. Elsevier, 2001.

[25] G. Huang, B. K. Taylor, and D. Akopian, "A low-cost approach of magnetic field-based location validation for global navigation satellite systems," *IEEE Transactions on Instrumentation and Measurement*, 2019.

[26] P. Huray, *Maxwell's Equations*. Wiley, 2009, vol. 9780470542767.

[27] K. Yosida, *Theory of magnetism*, ser. Springer series in solid-state sciences. Springer-Verlag Berlin Heidelberg, 1996, vol. 122.

[28] J. R. Cary and S. G. Shasharina, "Omnigenity and quasihelicity in helical plasma confinement systems," *Physics of Plasmas*, vol. 4, no. 9, pp. 3323–3333, 1997.

[29] O. C. Zienkiewicz, R. L. Taylor, P. Nithiarasu, and J. Zhu, *The finite element method*. McGraw-hill London, 1977, vol. 36.

[30] A. M. Hussein and A. Elnagar, "Motion planning using maxwell's equations," *Intelligent Robots and Systems*, vol. 3, pp. 2347–2352, 2002.

[31] J. G. Van Bladel, *Electromagnetic fields*. John Wiley & Sons, 2007, vol. 19.

[32] P. G. Huray, *Maxwell's equations*. John Wiley & Sons, 2011.

[33] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, and M. S. Lew, "Deep learning for visual understanding: A review," *Neurocomputing*, vol. 187, pp. 27–48, 2016.

[34] X. Guo, W. Li, and F. Iorio, "Convolutional neural networks for steady flow approximation," *International Conference on Knowledge Discovery and Data Mining*, pp. 481–490, 2016.

[35] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[36] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241, 2015.

[37] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 834–848, 2017.

[38] M. I. Meyer, A. Galdran, A. M. Mendonça, and A. Campilho, "A pixel-wise distance regression approach for joint retinal optical disc and fovea detection," *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 39–47, 2018.

[39] M. Moussa and G. Beltrame, "Supplementary material," https://mistlab.ca/papers/MaxConvNet, 2021, [Online].

[40] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," *Proc. of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, 2015.

[41] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *Proc. International Conference on Learning Representations (ICLR)*, 2015.

[42] M. N. Ã-zisik, M. N. Özısık, and M. N. Özışık, *Heat conduction*. John Wiley & Sons, 1993.

[43] N. Endou, K. Narita, and Y. Shidama, "The lebesgue monotone convergence theorem," *Formalized Mathematics*, vol. 16, no. 2, pp. 167–175, 2008.

[44] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," *Field and service robotics*, pp. 621–635, 2018.

[45] S. Russel, P. Norvig *et al.*, *Artificial intelligence: a modern approach*. Pearson Education Limited, 2013.