POLITECNICO DI TORINO
Repository ISTITUZIONALE

Assessing an Image-to-Image Approach to Global Path Planning for a Planetary Exploration

(Article begins on next page)

27 April 2024

Article

Assessing an Image-to-Image Approach to Global Path Planning for a Planetary Exploration

Guglielmo Daddi *,† [ID] **, Nicolaus Notaristefano †, Fabrizio Stesina †** [ID] **and Sabrina Corpino †**

Department of Aerospace and Mechanical Engineering, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129 Torino, TO, Italy

* Correspondence: guglielmo.daddi@polito.it

† These authors contributed equally to this work.

Abstract: This work considers global path planning enabled by generative adversarial networks (GANs) on a 2D grid world. These networks can learn statistical relationships between obstacles, goals, states, and paths. Given a previously unseen combination of obstacles, goals, and an initial state, they can be asked to guess what a new path would look like. We performed experiments on a 64×64 pixel grid that generated a training set by using randomly positioned obstacles and goals. The heuristic search algorithm A* was used to create training paths due to its significant presence in the literature and ease of implementation. We experimented with architectural elements and hyperparameters, converging to a pix2pix-based architecture in which the generator was trained to generate plausible paths given obstacles and two points. A discriminator tried to determine whether these maps were real or fake. Additionally, we defined a qualitative path-generation "success rate" metric derived from the Fréchet inception distance (FID) and optimized our architecture's parameters, ultimately reaching a 74% success rate on the validation set. Furthermore, we discuss the applicability of this approach to safety-critical settings, concluding that this architecture's performance and reliability are insufficient to offset the downsides of a black-box approach to path generation.

**Keywords:** path planning; machine learning; generative adversarial networks

1. Introduction

Many of the questions posed by planetary scientists rely on analyses that cannot be performed remotely, but require in situ samples [1]. It has also been argued that in situ mobility systems generally exhibit greater scientific return when capable of some form of mobility [2] due to the increased sampling opportunities and sample diversity compared to those of a system fixed in space. For these reasons, path-planning techniques for planetary mobility systems are an area of active interest, as seen in the large body of literature that has already been published on the subject [3]. In this paper, we assess the performance of a general path-planning technique based on deep generative networks that is capable of approximating the results of any path-planning algorithm.

Rovers are some of the most common systems for in situ science, and their utility is roughly proportional to the area that they can cover throughout their operations phase (all else being equal). A major area of focus for increasing the scientific return from mobility systems is speed (for planetary rovers, the aim for next-generation systems is to move hundreds of meters per hour, as compared to the tens of meters in current systems). Effective autonomous mobility requires path planning that leverages both "local" and "global" environmental information. Local information is used for high-resolution short-term planning, while global information is used to avoid local optima and choose the most effective path. This global information typically comes from images that are segmented and processed into traversability maps and taken by supporting systems or orbiters.

1.1. Mission Reference Architecture

As shown in Figure 1, we imagine an observer that senses the environment and generates a path that is subsequently communicated to the mobility system. The ground-based system executes on-board local path planning and receives high-level movement suggestions from the airborne observer that are appropriately integrated into local path planning. This paper is structured as follows: Section 1.2 defines the path-planning problem and briefly reviews the concepts used throughout this work. Section 2 describes the image-to-image approach to path planning. Section 3 contains architectural details relating to our implementations. Section 4 outlines the results of our experiments and discusses what these results can say about this approach's applicability to a real-world scenario. Despite success rates of 74%, we will see that a statistical approach to generative path planning is not well suited for safety-critical applications. Offline network training would also complicate any mission in uncertain or unknown environments, as thorough generalization tests would have to be performed before deploying pre-trained networks.
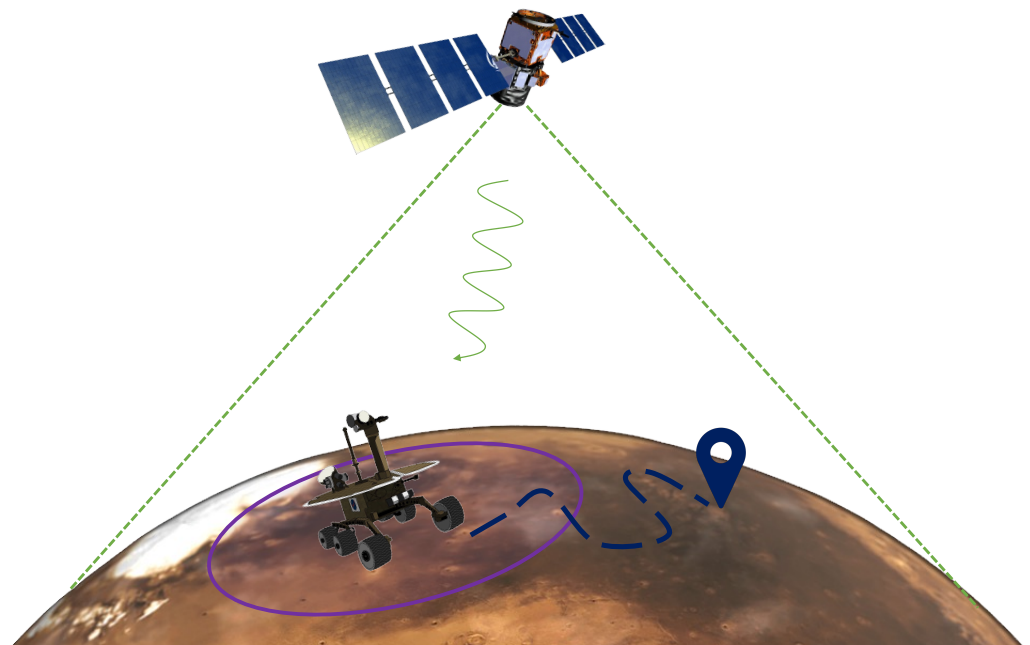
Figure 1. Sketch of the reference architecture. A rover with limited visibility receives a global path from a space-based asset.

1.2. Path Generation

Path planning is the generation of a sequence of states from an initial state to an end goal through continuum space. An agent is in state A and desires to achieve state B; the path-planning process generates a sequence of intermediate states that join A and B while complying with the dynamic constraints acting on the mobility system. Constraints come under many different forms. For problems described only by a pose vector, constraint examples could be the system's performance envelope (e.g., curve radius not inferior to x meters) and obstacle avoidance. For larger state spaces, there can be a number of additional constraints, such as resource limits, communication windows, etc. Since infinite feasible paths might connect any two points, the objective of minimizing a cost metric (e.g., distance, time) is added in order to reduce the space of solutions. There is a vast literature on motion planning, and we will provide the reader with pointers to relevant review papers. Researchers have focused on aspects including finding the ideal representation for physical space [4], algorithm design [5], optimal path generation [6], three-dimensional environments [7], information-based approaches [8], aerial applications [9], and underwater applications [10]. A system may encounter undetected issues, such as small obstacles, unexpected hazards, or shifts in the terrain over time. Therefore, global path planning has been

adapted to dynamic situations [11] and supported by local path-planning algorithms [12] that perform obstacle avoidance and generate trajectories based on information gathered from on-board sensors [13]. These local algorithms have been shown to be incapable of guaranteeing success in particularly challenging environments [14,15]. Additionally, for traversals beyond the on-board sensor's range, finding the optimal path requires global knowledge. In its absence, the agent might take a wrong turn or have to backtrack.

Path planning is a deeply studied problem for terrestrial robotics [5,16], and many of the techniques used for path planning in terrestrial robots apply to space robotics without much modification. Some critical challenges that are particularly marked for space systems are related to the environment. The space environment is often partially observable, highly uncertain, and extremely remote. The remoteness leads to costly systems, therefore placing even greater safety constraints when compared to typical Earth-based systems. Environmental uncertainty and the lack of complete observability place an emphasis on online planning techniques, such as obstacle avoidance and re-planning, rather than offline techniques, which may be based on poor data [17]. However, online planning alone cannot reach globally optimal solutions.

Most work on global path planning for space systems is focused on Martian applications, starting with work in preparation and support of the Mars Exploration Rover (MER) program [18,19] and similar work performed for the Mars Science Laboratory (MSL) program and Mars 2020 (M2020). Path-planning techniques have also been assessed by many other actors [20–23] for different mission architectures. A growing number of similar investigations that are centered around lunar applications can be found [24,25], where both governmental and commercial entities are pushing to deploy robotic systems in support of the next phase of human space exploration. Other path-planning examples range from balloon missions on Venus [26] to asteroid-hopping robots [27]. With the proliferation of increasingly affordable space-based imaging systems, the quality, quantity, and availability of global maps over which to perform planning are likely to grow, making path-planning techniques inspired by mission architectures, such as that in Section 1.1, worth investigating. The clearest example of a similar mission architecture can be found in the Ingenuity helicopter, which was used as a Scout for the Perseverance rover [28]. In this context, our research is aimed at exploring novel path-planning techniques that may be applied to mission architectures in which a mobility system is aided by global maps generated by a supporting agent. Nevertheless, we want to be clear that the concept of learning a computationally expensive algorithm and embedding this knowledge in a neural network for fast evaluation is not limited to path planning, but can be applied with varying degrees of success to any field in which there is a clear distinction among inputs, outputs, and training data. An example of another aerospace field in which similar algorithms could be applied is spacecraft trajectory planning and optimization [29].

In general, high-dimensional motion-planning problems with many constraints are solved with sampling-based approaches that do not guarantee optimal solutions in a finite runtime. Recent advances have framed similar problems as mixed-integer optimization problems, which lead to both optimality and computational efficiency [18,30]. Such approaches inevitably restrict the class of problems that can be efficiently tackled. Our approach forsakes optimality in favor of generality, as it can applied to spaces with very complex configurations that would not be compatible with mixed-integer formulations.

1.3. Machine-Learning-Based Approaches to Path Planning

There are many review papers that have described approaches to path planning, such as randomized techniques [31], artificial potential fields [32], heuristic-search-based techniques [33], any-angle techniques [34], and many more. Most of the attempts to include machine learning in motion planning have historically been focused on augmenting the planning process through data processing and state estimation techniques [8]. This tendency to use machine learning to complement existing algorithms is unsurprising because of the fragility and proneness to unexpected failure modes of data-driven approaches. Of

the few attempts to use machine learning to perform global path planning that we could find, most approaches have focused on reinforcement learning [35] and, more recently, generative adversarial networks. In data-driven approaches, path optimality and completeness are rarely supported by theoretical guarantees; therefore, machine learning is generally used to decrease the path generation's computational burden from the points of view of both time complexity and program memory. Neural networks are valuable tools for learning complex relationships between data, allowing the mapping of an input state directly to a path or action without the state exploration entailed by classical algorithms (e.g., breadth-first search). The planning problem can be formulated as a sequence of decisions that are not influenced by past actions and, therefore, falls under the theoretical framework of Markov decision processes (MDPs) [36]. Reinforcement learning is a way to learn effective policies for MDPs and, therefore, has been applied to global path-planning problems [37]. Q-learning has been used to discover a collision-free path between two states [12]. Similar results have been achieved with inverse reinforcement learning in conjunction with convolutional neural networks and value iteration [38]. Another inverse reinforcement learning approach using the maximum entropy paradigm to approximate reward functions was presented in [39]. Computational experiments and theoretical advances have proceeded in parallel, as seen in the efficient framework for cost function learning developed in [40] or that for risk-averse planning formulation presented in [41]. Another approach to machine-learning-based planning was proposed in [42], where deep CNNs trained with standard backpropagation effectively approximated value iteration. A dual-branch CNN architecture that avoids passing through an environmental mapping was used to generate paths through value function approximation [42]. In the literature, we found three previous works attempting to tailor generative adversarial networks to global path planning and a similar approach for circuit routing. One approach [43], used conditional generative adversarial networks (CGANs) [44] that took environmental maps as input and paths generated using the rapidly exploring random tree (RRT) algorithm to increase the underlying algorithm's efficiency. Their neural architecture was based on an encoder–decoder with skip connections (U-NET) generator [45] and a deep convolutional neural network discriminator. RRT was run 50 times for every training point pair instance. The probabilistic nature of RRT allowed for the generation of an "area of interest" rather than a single path. The generative network learned to approximate this area of interest, which could then be used as a new state space to run RRT. This newly learned domain allowed faster RRT convergence compared to the reference state space. Another RRT-based approach can be found in [46]. Their architecture had an encoder–decoder generator made from convolutional and residual network [47] layers that enabled less detail loss compared to that with a U-NET architecture. They also proposed a double discriminator, one for the initial/target point pair and the other for the region of interest, to improve the generator's effectiveness. RRT's inherent stochasticity led to the generation of these "areas of interest", and the approach was robust to noise in the generated images. Note that this approach enhanced a path-planning algorithm rather than substituting it. A conceptually different approach was proposed in [48], where a path planner based on the A* algorithm was approximated with a CGAN using the pix2pix architecture [49]. Their generator used a U-NET architecture with skip connections between layers. Their loss function was made from a cross-entropy component for the generator summed with the adversarial loss. At the same time, their discriminator focused solely on the path's quality rather than also taking obstacles as inputs.

In the realm of circuit routing, a partially successful generative approach based on the pix2pix architecture can be found in [50]. It is interesting to note the similarities between circuit design and motion planning, as the constraints posed on the circuit traces are comparable to those in our application.

2. Materials and Methods

Our approach to path planning is based on Ian Goodfellow's work on generative adversarial networks (GANs) [51]. These unsupervised models aim to learn patterns in data in order to generate reasonable approximations of novel data points. A multitude of research projects and commercial tools have been developed since the introduction of generative networks, with use cases ranging from image super-resolution, face and texture synthesis, natural language processing, medical diagnosis [52,53], computational fluid dynamics [54], and many other engineering applications [55]. All GANs share a high-level architecture in which a generator G is trained to produce samples starting from a random noise vector. At the same time, a discriminator D tries to determine whether the samples are real (taken from the training instances) or fake (generated). In other words, the discriminator outputs the conditional probability label y that predicts if a sample is part of the training set or the generator's output [56] (Chapter 10.4). The two networks are trained concurrently in a game-theoretic adversarial game that reaches higher fidelity than generative modeling alone. The adversarial game can be formulated as a two-player min–max problem in which the discriminator's weights are changed at each training step to decrease the likelihood of being fooled. In contrast, the generator's weights are changed in the direction that is more likely to mislead the discriminator. For the particular case of two multi-layered perceptrons where the generator is passed a noise vector as input, this adversarial game can be formulated as [51]:

$$\min_G \max_D \left(\mathbb{E}_{x\,p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\,p_z(z)}[\log(1 - D(G(z)))] \right) \tag{1}$$

where x is the data, and $D(x)$ is the probability of a sample belonging to the data or the generator's output distribution p_g. The idea behind a generative approach to path planning is to use a GAN to learn the relationships between environmental maps, start and finish points, and paths generated through a classical algorithm. This knowledge is embedded into the network's weights and allows the generation of plausible paths given a new map and start/end points. The previous sentence implies two main points. Firstly, the path-planning GAN should have an internal structure that can process images, and secondly, the GAN should be able to condition its inputs with obstacle maps and goals.

2.1. Dealing with Images

High image generation or discrimination performance requires a different approach from that of multi-layer perceptrons. Since the advent of modern GPUs, deep convolutional neural networks (DCNNs) have been shown to be a very efficient and effective way to learn features from images [57], and their use when paired with GANs was first shown to be effective in [58]. Convolution layers are the central feature of convolutional neural networks, as they embed an image into representations of decreasing size with each layer. The inverse process can pass from a smaller representation to a larger one, thus passing from a noise vector to an image. The central point of CNNs is that smaller representations entail higher levels of abstraction, allowing the network to reason about more complex structures than single pixels.

2.2. Conditioning Input

Another aspect of Goodfellow's GAN paper was random noise input vectors. Solving a path-planning problem cannot be done solely with noise vectors, but requires the conditioning of the generator with an input map and target goals. Without input conditioning, the generated path would be unaware of goals and obstacles and would resemble a random path. Conditional generative adversarial networks (CGANs) are the solution to this issue, as they were developed as an extension to GANs that accepts input conditioning. These networks were introduced in order to automatically generate image tags [44], but have been used for image-to-image translation tasks, such as passing from labels to images [49],

colorizing black and white pictures, transforming sketches into photos, changing a picture's lighting conditions, and generating maps from satellite images and vice versa [59].

In practice, CGANs work as normal GANs, but with an additional input layer that combines the noise vector with some other information *y* in a hidden representation. The problem formulation is remarkably similar to that for normal GANs, with the addition of conditional probabilities in the log loss computations [44].

$$\min_G\max_D\Big(\mathbb{E}_{x\ p_{data}(x)}[\log D(x|y)]+\mathbb{E}_{z\ p_z(z)}[\log(1-D(G(z|y)))]\Big) \quad (2)$$

2.3. Pix2pix

A widespread and successful CGAN architecture that has been used in most generative approaches to path planning is the pix2pix architecture, which was first presented in [49]. The two main reasons for this architecture's widespread use are its effectiveness and the code base's open availability. In pix2pix, both the generator and discriminator follow a pattern where convolution, batch normalization, and Relu activation layers are alternated. The convolution layers progressively abstract the image into smaller and smaller feature maps; the Relu layers serve as rectifying activation functions and effectively allow the approximation of nonlinear functions, while batch normalization improves the chances of convergence for large networks by normalizing the inputs. The generator follows the general U-net shape [45], that is, a reinterpretation of the encoder–decoder network, where the input is first scaled down into a hidden representation up to a bottleneck point, after which it is decoded and expanded to its original dimension. The discriminator used in pix2pix is also a refinement of classical GAN discriminators. Rather than outputting a single real/fake label for an image, Patch-GAN discriminators consider $N \times N$ pixel patches across the generated image and create real/fake labels for each patch via a convolution operation. The result is an average of the resulting labels. This patched approach to discriminators is used in pix2pix because it encourages high-frequency sharpness in the generated images, which is better than that with single labels [49]. Pix2pix's objective formulation differs slightly from a traditional CGAN by adding (from Equation (2)) a more traditional L1 loss term to the CGAN loss $\mathbb{L}_{CGAN}$.

$$\min_G\max_D(\mathbb{L}_{CGAN}+\lambda(\mathbb{E}[\|y-G(x|z)\|_1])) \quad (3)$$

3. Generative Path-Planning Approach

Our approach to assessing the GAN-based path planning that was described in Section 2 leverages the results obtained in previous work (Section 1.3) by implementing the most promising aspects and experimenting with architectural and hyperparameter variations in an attempt to tease out as much performance as possible from current widely available technology. The main decisions made were related to the architecture, algorithm, and performance metrics. From the architectural perspective, we followed the general framework provided by pix2pix. With the algorithmic choices, we refer to the path-planning algorithm implemented. From this point of view, we chose the A* heuristic global search algorithm [33], rather than an anytime algorithm, as in [44]. Our path planner aims to approximate an optimal solution rather than perform a speedup that would reduce the search space for a stochastic algorithm. Mimicking an external agent suggests a fuzzy global path to a mobility system that is only aware of its close surroundings. The environment in our problem is described as a grid map [60]. We chose to represent this map as a three-channel tensor (RGB image) rather than a smaller representation to allow for goals and path cells to have a larger distance in the latent space from the map cells than the distance that they would have if they were just a shade of gray. Additionally, we chose RGB images because only three-channel images can be used to compute the success metric described in Section 3.2. As seen in Figure 2, this comprised empty, obstacle, goal, and path cells. The robot's state consisted in its coordinates in the world. We assumed

that the generated obstacles were increased in size by an amount equal to the robot's size. Therefore, any valid path was guaranteed to be safe.
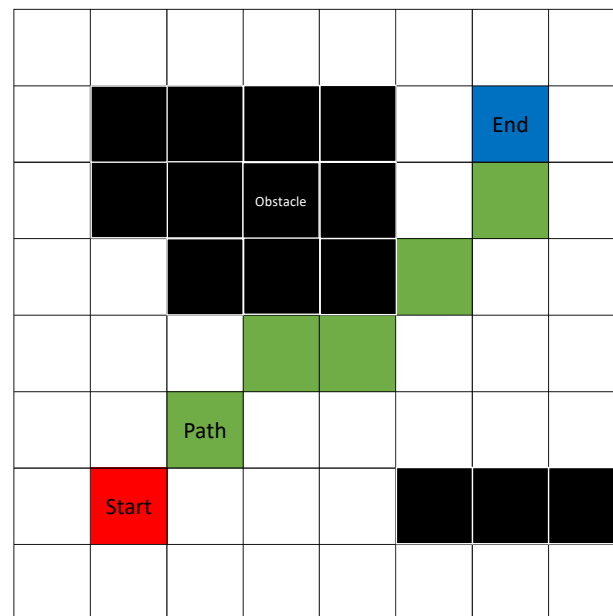
Figure 2. Grid world representation. Cells with a value of [0,0,0] (black) are obstacles, while cells with [255,255,255] (white) can be traversed. Red [255,0,0] and blue [0,0,255] cells are, respectively, starting and ending positions, while green [0,255,0] cells represent path cells.
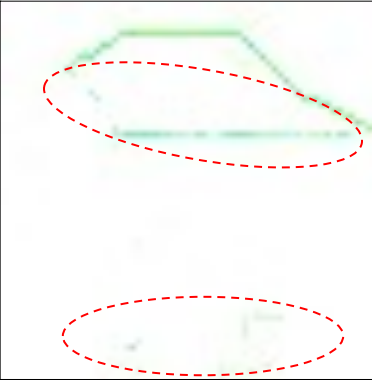
3.1. Input and Outputs

This method takes as an input a traversability map alongside a starting point and an end goal. The output is a matrix that describes a path that imitates the line connecting the starting point to goal generated via the A* algorithm. In real applications, the traversability map would be generated from images of the terrain in conjunction with the robot's mobility model, as described in Section 1.2, although our case considers a toy problem in which the maps are synthetically generated.

3.2. Success Metrics

As success metrics, we chose the Fréchet inception distance (FID) [61] due to its consistency with human judgement and robustness to noise [62]. This metric was obtained by feeding the ground-truth and generated images into the Inception V3 network [63] to get feature maps. These features were compared, and a Weierstrassian "distance" metric was derived. Computing the FID metric requires a three-dimensional tensor, as InceptionV3's architecture is built around RGB images. This is a limitation of our method, as the occupancy information could be encoded into a much simpler space. In fact, a 2D matrix whose positions were taken from the set {empty, occupied, start, end, path} would be sufficient. We used the FID metric in two ways depending on the efficiency required. Working with large image distributions leads to a slow evaluation time for every image pair using the InceptionV3 network. This slowness is acceptable when evaluating the final result once training is over. With each image's FID, we can plot a distribution that can be used to assess how effective the network is. We call each pairwise FID computation a Particular-FID (PFID). The FID use case that requires efficiency rather than completeness is network training. We would like to know how "good" the network is after each epoch without pairwise comparisons. Still, since the loss is not a particularly informative parameter for generative adversarial architectures [62], we use the FID metric taken by comparing an aggregate of all ground truths with a total of all generated maps. This leads to a single InceptionV3 evaluation and is significantly faster than pairwise comparisons. We call this

aggregated FID the Overall-FID (OFID). As shown in Figure 3, the FID metric can be tied to the types of artifacts that appear in the generated images. We noted from our results that for FID values that were smaller than 100, most errors could be easily filtered out. These errors consisted of pixels that were missing or slightly moved from the ground-truth path and noise pixels that were not red, green, or blue. The errors changed in nature for FID values that were greater than 100, and the way became irrecoverable. This happened because multiple adjacent pixels could miss or be misplaced from the path, the origin and destination could vary from their intended coordinates, and ghost sections of the path could be generated at random positions on the map. Due to these observations, we introduced a heuristic evaluation metric, the "success rate", which was computed by taking the percentage of the PFID over the validation set of a trained net below an FID value of 100. This metric can quickly give an idea of how many generated paths can lead to helpful guidance, but is limited by the heuristic nature of the "success FID" setpoint. A more accurate version of this metric that could be implemented in future work would be computed by applying a filter to the generated path and seeing what proportion of paths are feasible after filtering.

FID < 50	50 < FID < 100	FID > 100
Noise	Noise + Deviations, Missing pixels	Noise + Deviations, Missing pixels + Missing chunks, Wrong start or end points, Incomplete sections

Figure 3. Heuristic representation of what errors are generated depending on the FID metric.

3.3. Dataset Generation

Neural networks need training data and validation data. In our case, these data were procedurally generated. A script created a set of $64 \times 64 \times 3$ RGB grids of white space that signaled empty cells. As shown in Figure 4, obstacles were signaled by black rectangles, triangles, circles, and semicircles that were randomly placed throughout each map with randomly varying sizes that were bounded by a maximum and minimum characteristic length. Vacant and occupied cells together formed the map space M. We chose black and white to maximize the vector distance between empty and occupied cells. Each map was paired with a set of randomly generated starting points and end goals that formed the point space P. These were, respectively, red and blue pixels. For each pair of goals, a ground-truth track was generated by running the A* algorithm and saving the generated path as green pixels. The set of paths was called the path space S. We split the generated datasets, using 80% for training and 20% for validation. The number of maps, number of goal pairs for each map, and sizes of the end and start goals were parameters that we experimented with.
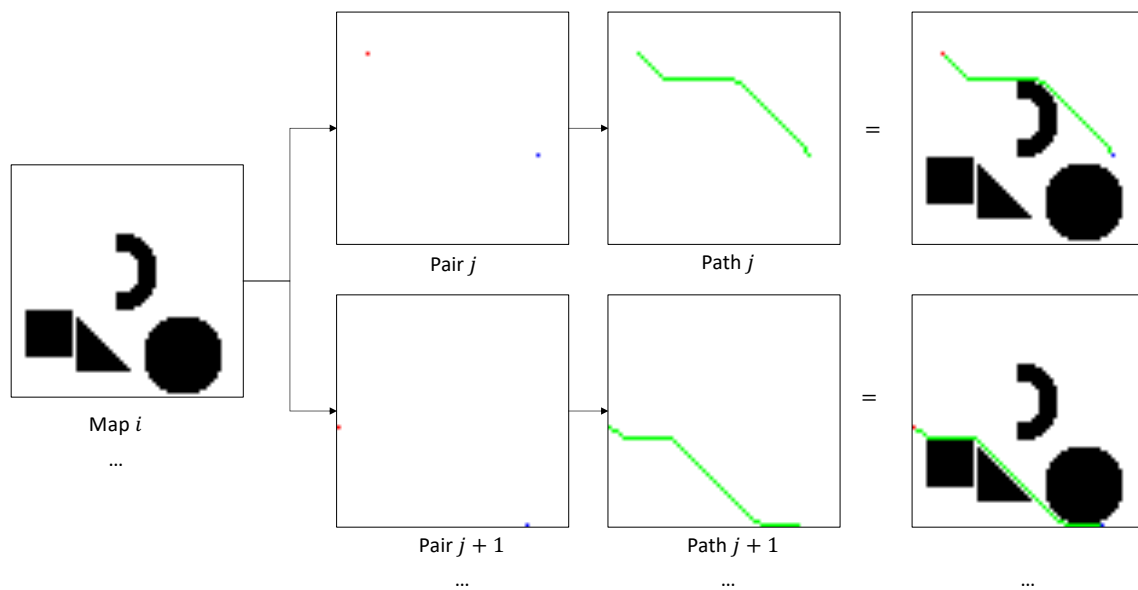
Figure 4. Dataset structure. Each map of procedurally generated obstacles was subjected to multiple goal pairs that were used to generate a path via A*.

3.4. Architecture

Our network's architecture was based on pix2pix (Section 2) and can be observed in Figure 5. The generator took as input a map and a point image (*M* and *P*), which were each passed through a convolution layer with 32 channels. The results of this operation were concatenated into a single map with $32 \times 32 \times 64$ dimensions and fed to a U-Net. The U-Net's encoder was made from an alternation of Convolution–LeakyRelu and Convolution–BatchNormalization–LeakyRelu layers with filters of 4×4 pixels. On the other hand, the decoder network used Deconvolution–BatchNormalization–LeakyRelu and Deconvolution–Tanh layers. Skip connections were established between encoder and decoder layers of the same size as anticipated in Section 2, skip connections improve performance for deep architectures by providing alternative pathways for the gradient during backpropagation. The main parameter that we experimented with was the U-Net's bottleneck size by varying the number of layers in the U-Net. The discriminator took as input the map *M*, goals *P*, and either the ground-truth or generated path *P*. Similarly to the generator's encoder, the discriminator concatenated the input maps into a single data structure that was passed through Convolution–LeakyRelu and Convolution–BatchNormalization–LeakyRelu layers with 4×4 filters. The final layer was a pure convolution layer that generated a 6×6 truth map. The 36 truth values were averaged to determine whether the overall map was true or false.

The training algorithm that we used was outlined in the original pix2pix paper [49]. It is helpful to acknowledge that the essential hyperparameters were batch size and several training epochs. We varied these hyperparameters during the experiments to look for an optimal combination. The optimizer used to update the generator and discriminator weights at each epoch was the Adam optimizer [64].

We implemented the architecture using the PyTorch machine learning framework and used Anaconda virtual environments for code portability. Training and validation maps and paths were generated in external Python scripts, where the PIL and NumPy libraries were used to facilitate operations with images.
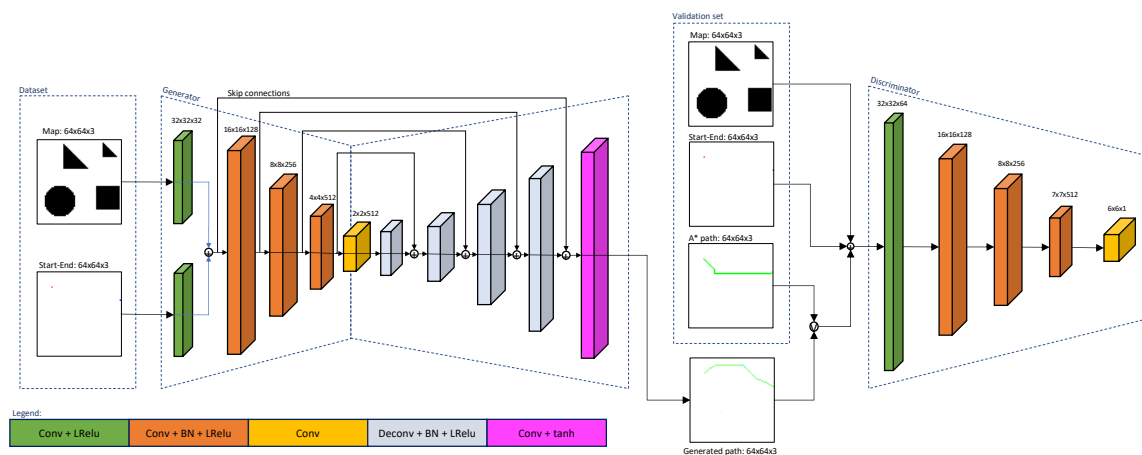
Figure 5. High-level GAN architecture sketch. The number of layers in the generator changed throughout the experiments.

To streamline the presentation of the results, we labeled our tests with names that describe the network architecture, gain, number of epochs, batch size, and dataset size. The naming convention is outlined in Equation (4).

$$number\_of\_epochs.batch\_size.network\_ID.dataset\_ID \qquad (4)$$

The number of epochs and network size have a one-to-one correspondence between the name and ID, while the network and dataset ID can be read in Table 1.

Table 1. Labeling conventions for the experiments.

| Network ID	Network Description	Dataset ID	Dataset Size	Goal Size
01	Baseline	01	44,500	1×1
02	$4 \times 4 \times 512$ bottleneck	02	110,500	1×1
03	$1 \times 1 \times 512$ bottleneck	03	110,500	4×4

4. Results

In the next subsections, we explore our efforts to improve the network's performance by varying the hyperparameters and some architectural details. For a quantitative view, we refer to Table 2. Each experiment ID describes the value of each hyperparameter according to the relationship described in Equation (4). Table 1 can be used to infer the sizes of the datasets, goals, and bottlenecks starting from the experiment ID. No changes in the GAN algorithm were attempted in order to stay in line with our intention to assess what performance can be expected from existing generative algorithms.

Table 2. Success metrics of the various experiments.

Experiment ID	Final Epoch OFID	Success Rate
50.16.01.01	38.31	38.45%
150.16.01.01	30.29	43.68%
50.16.01.02	29.21	47.52%
200.16.03.02	27.07	48.62%
200.128.02.02	27.16	N/A
80.16.02.02	19.88	56.69%
150.16.02.03	10.70	73.88%

4.1. Epoch Variation

Mostly, we saw that the OFID metric improved over each training epoch with an asymptotic behavior. This was wholly expected and compatible with neural network training patterns. The asymptotic behavior implied that the training process was able to reach a point of diminishing returns, after which the computer resources were inefficiently employed, and overfitting became more probable. This phenomenon was tracked through the loss function over time for classical neural networks. Still, for GANs, we need to use a success metric, as loss alone is not sufficient to completely understand what is happening to the results due to the adversarial nature of the game. As shown in Figure 6, the point at which improvements were negligible was around 100–150 epochs depending on the specifics of each test. We experimented with 50, 80, and 200 epochs, but eventually converged on 150 epochs as a reasonable tradeoff between time and accuracy. In future iterations, we will consider implementing early stopping based on the variation in the rolling average of the OFID metric.
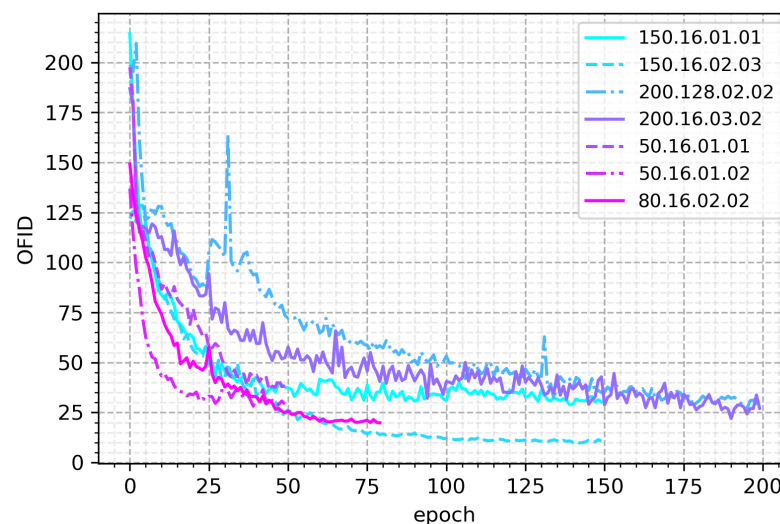
Figure 6. Training process: overall FID trend.

4.2. Dataset Size

We tested only two dataset sizes due to their long generation time. The first dataset consisted of 44,500 paths, while the second one comprised 110,500 paths. We noted that the larger dataset improved the generalization capacity at a constant epoch number. This behavior was expected, as the number of batches used for each training step increased for the larger dataset. The trade-off here was that larger datasets led to more training time for each epoch and the generation of the dataset increased. Choosing the optimal dataset size can be tricky, since larger is always better for neural network datasets. Still, since we noted an OFID increase of 10% with a doubling of the training set, we decided that further increases were not worth the additional computational time. This is in line with our intention to probe the effectiveness of this path-planning strategy, not to try to extract the maximum possible performance from our network. Therefore, costly optimizations leading to a few percentage points in OFID gain were not considered.

4.3. Bottleneck Size

The bottleneck represents the degree of compression applied to the generator's input maps. Optimizing the size of the bottleneck layer is a matter of trial and error, as it is a tradeoff between complexity and compression, where both a lack of or excessive compression in the feature map can lead to non-satisfactory performance. We experimented with this metric by adding or removing convolution blocks, starting from a baseline architecture with a $2 \times 2 \times 512$ bottleneck. We first tried shrinking the bottleneck to a

$1 \times 1 \times 512$ block and then expanding it to a $4 \times 4 \times 512$ element. We noted no significant improvement for the $1 \times 1 \times 512$ case, while increasing the bottleneck size to $4 \times 4 \times 512$ yielded a 10% increase in the success rate. This is symptomatic of excessive compression in the smaller bottleneck architectures.

4.4. Initial and End Goal Size

We noted in the experiments that the attempted architectural changes did not lead to success rates greater than 50%. This means that over half of the validation paths were not sufficiently clear or coherent to allow navigation. A common source of error that we observed was incorrect starting points and endpoints. This was where coherence within the path was respected (no missing pixels or incomplete sections), but the beginning and end points were incorrect, leading to a high PFID. This led us to believe that information about the start and end goal pixels was being compressed into an unusable state. Subsequently, random noise artifacts were mistaken as goals, and a path was plotted between these two points. These considerations led to the idea of increasing the goal's size from a 1×1 square to a 4×4 square on the grid. The goal itself did not change, meaning that the A* path was still a computer starting from a single pixel, but the goal's representation increased from 1.6% to 6.2% of the map's characteristic length. In principle, this would decrease the probability of excessive compression while introducing a topological risk of goals and obstacles intersecting. Due to this risk, we rejected the map instances where the goal pixel would be placed on vacant space, but the 4×4 representation of the goal would intersect with an obstacle. This goal size change significantly affected the network's performance, which jumped from hovering below 50% to 74%. This is visible in Figures 6 and 7, where the 150.16.02.03 line had a significantly better OFID asymptote, and its PFID distribution was visibly skewed towards lower FID than those in the other experiments.
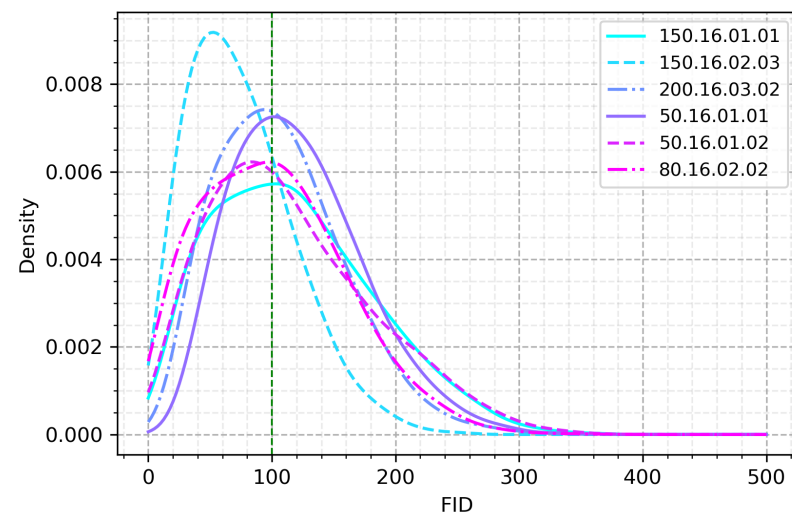
Figure 7. Trained networks: FID distribution over the validation set.

5. Discussion

While a 74% success rate is an impressive result for a path-planning method that knows very little about both paths and planning, the critical point is that these experiments were conducted in ideal circumstances. The maps were small at 64×64 pixels, and the obstacles were limited to four shapes. The limited obstacle set aided the network by diminishing the number of features that it had to learn to navigate. In a sense, the problem's dimension was reduced by limiting the shapes of obstacles encountered. In an actual application, even considering perfect terrain classification, the "obstacle pixels" would be arranged in more diverse patterns that would require additional training. We chose to limit the dataset's scope to provide an upper bound to the performance that can be expected. If a network does not generalize well with a limited obstacle set, its performance can only worsen when

exposed to a real environment. This success-rate reduction effect can only worsen when considering an exploration setting, where the training set will only be an educated guess on the operational environment's details. In future work, it would be interesting to train and validate this approach on real-world traversability maps to measure the performance degradation that can be expected when transitioning from ideal to realistic settings. It is also worth noting that the training process embedded the capability of generating paths around a specific and limited obstacle set into the network's weights. Real-world traversability maps are not made from such a small set of obstacle geometries; therefore, the network would not generalize well with the current weights to more realistic examples. Moving to real-world testing would require training the network from scratch on a more realistic traversability map dataset. Such a task would add the additional step of converting the dataset into a traversability map and was, therefore, avoided in this work so as to avoid potential pitfalls in terrain labeling or traversability estimation. Our intention was solely to probe the performance limits of the path-planning approximation algorithm under ideal circumstances. In other words, we overfitted our network to a fictitious world where only simple obstacles of varying sizes exist in order to bound our expectations of performance in more realistic scenarios. Real traversability maps are not made of a limited set of regular shapes, which poses challenges to the network's generalization capacity. Thus, these results can be interpreted as a higher bound to this architecture's success rate. Larger input images pose an additional challenge, as they are likely to lead to training instabilities for an unmodified pix2pix architecture [65]. Extensions and modifications have been proposed to apply CGANs to high-resolution images, with mechanisms such as improved objective functions, coarse to acceptable generators, and multi-scale discriminators [66]. Recent developments in high-resolution, real-time adversarial network usage for image generation can be seen in the computer graphics industry [67]. These impressive improvements signal ever-increasing image translation capabilities and warrant further investigations into this path-planning approach. Another way to tackle high-resolution maps (that can also be applied to classical planning algorithms) is to down-sample high-resolution images into a lower-resolution space over which the algorithm can plan a path. The down-sampled path can then be up-scaled to the original resolution. This method leads to further sub-optimalities and potentially missing small-scale obstacles that may get deleted during the down-sampling process.

The main issue with this approach is that unsuccessful paths are generated with neither success guarantees nor failure awareness. When an infeasible path that leads to a dangerous state is generated, the pix2pix architecture is unaware that it is generating a path that violates safety constraints. Therefore, the robot would unknowingly guide itself to a risky state if not interrupted by additional reasoning layers that check and enforce these constraints. The minimum acceptable success rate would likely change depending depending on the mission's risk posture. This is why we framed our architecture as a navigation aid that guides global path planning for agents with limited perception. In this setting, there could be instances in which high-success-rate networks could learn to imitate a global path-planning algorithm and be helpful to the ground-based agent without excessively endangering it by providing low-level guidance. However, due to a failure rate of 26% despite being trained and validated on small maps with a limited obstacle set, we can say that our pix2pix-based architecture (and associated hyperparameters) requires further tuning before being deployed to any path-planning approximation task. Higher success rates and robust generalization capacity would have to be shown before transitioning to real-world hardware experiments. Given the reference architecture described in Section 1.1, the GAN-based path-planning algorithm that we propose would not be tasked with online local path-planning, but would rather be tasked with generating a path suggestion aimed at improving the quality of the local path planner offline. Safety-critical algorithms deployed on the rover would be proven, verifiable, and highly safe algorithms, while the global path suggestion could be fuzzier and less reliable, as even incorrect suggestions would merely delay the local path planner's solution when extended to a global context.

Deep learning and computer vision are fast-moving fields with vast swaths of knowledge being added each year; therefore, it is not unreasonable to expect significant improvements with architecture changes or hyperparameter tweaking. For this reason, it is worth revisiting this approach with improved network architectures and more realistic datasets in the future. Furthermore, it is worth attempting to adapt this technique to other levels of planning (e.g., mission planning) to either achieve a speedup of existing search algorithms or approximate the results of said algorithms.

6. Conclusions

We implemented and tested a generative approach to global path planning based on the pix2pix-based conditional generative adversarial network architecture. We trained the architecture on paths generated on procedural obstacles maps of 64×64 RGB pixels via the A* heuristic search algorithm. We experimented with the effects of several training epochs and the sizes of the goals, bottleneck, and dataset. The most effective combination of architecture and hyperparameters that we found was capable of successfully approximating the path generated with the A* algorithm by 73.88% on validation maps generated with the same procedural rules as those of the training maps. While this specific approach to path planning is not suitable for safety-critical path planning, the results are promising enough to recommend attempting future iterations of this approach as generative network technology improves.

Abbreviations

The following abbreviations are used in this manuscript:

GAN	Generative Adversarial Network
CGAN	Conditional Generative Adversarial Network
FID	Fréchet Inception Distance
OFID	Overall Fréchet Inception Distance

References

1. Board, Space Studies, and National Research Council. *Vision and Voyages for Planetary Science in the Decade 2013–2022*; National Academies Press: Washington, DC, USA, 2011.
2. Carpenter, K.; Cable, M.L.; Choukroun, M.; Iacoponi, S.; Moreland, S.; Heverly, M.; Nayar, H.; Kennedy, B.; Bowkett, J.; Reid, W. The Science Case for Surface Mobility on Icy Worlds. *Bull. Am. Astron. Soc.* **2021**, *53*, 186.
3. Sánchez-Ibáñez, J.R.; Pérez-del-Pulgar, C.J.; García-Cerezo, A. Path Planning for Autonomous Mobile Robots: A Review. *Sensors* **2021**, *21*, 7898. [CrossRef] [PubMed]
4. Galceran, E.; Carreras, M. A survey on coverage path planning for robotics. *Robot. Auton. Syst.* **2013**, *61*, 1258–1276. [CrossRef]
5. Gasparetto, A.; Boscariol, P.; Lanzutti, A.; Vidoni, R. Path Planning and Trajectory Planning Algorithms: A General Overview. In *Motion and Operation Planning of Robotic Systems*; Springer: Berlin/Heidelberg, Germany, 2015; pp. 3–27.
6. Raja, P.; Pugazhenthi, S. Optimal path planning of mobile robots: A review. *Int. J. Phys. Sci.* **2012**, *7*, 1314–1320. [CrossRef]
7. Yang, L.; Qi, J.; Song, D.; Xiao, J.; Han, J.; Xia, Y. Survey of robot 3D path planning algorithms. *J. Control Sci. Eng.* **2016**, *2016*, 7426913. [CrossRef]

8. Bai, S.; Shan, T.; Chen, F.; Liu, L.; Englot, B. Information-Driven Path Planning. In *Current Robotics Reports*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 1–12.

9. Aggarwal, S.; Kumar, N. Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges. *Comput. Commun.* **2020**, *149*, 270–299. [CrossRef]

10. Panda, M.; Das, B.; Subudhi, B.; Pati, B.B. A comprehensive review of path planning algorithms for autonomous underwater vehicles. *Robot. Auton. Syst.* **2020**, *17*, 321–352. [CrossRef]

11. Stentz, A. Optimal and efficient path planning for partially known environments. In *Intelligent Unmanned Ground Vehicles*; Springer: Berlin/Heidelberg, Germany, 1997; pp. 203–220.

12. Gao, P.; Liu, Z.; Wu, Z.; Wang, D. A global path planning algorithm for robots using reinforcement learning. In Proceedings of the 2019 IEEE International Conference on Robotics and Biomimetics (ROBIO), Dali, China, 6–8 December 2019; pp. 1693–1698.

13. Ellery, A. *Planetary Rovers: Robotic Exploration of the Solar System*, 3rd ed.; Springer: Berlin/Heidelberg, Germany, 2015.

14. Carsten, J.; Rankin, A.; Ferguson, D.; Stentz, A. Global Path Planning on Board the Mars Exploration Rovers. In Proceedings of the 2007 IEEE Aerospace Conference, Big Sky, MT, USA, 3–10 March 2007; pp. 1–11.

15. Otsu, K.; Matheron, G.; Ghosh, S.; Toupet, O.; Ono, M. Fast approximate clearance evaluation for rovers with articulated suspension systems. *J. Field Robot.* **2019**, *37*, 768–785. [CrossRef]

16. Patle, B.; Pandey, A.; Parhi, D.; Jagadeesh, A. A review: On path planning strategies for navigation of mobile robot. *Def. Technol.* **2019**, *15*, 582–606. [CrossRef]

17. Wong, C.; Yang, E.; Yan, X.; Gu, D. Adaptive and intelligent navigation of autonomous planetary rovers—A survey. In Proceedings of the 2017 NASA/ESA Conference on Adaptive Hardware and Systems (AHS), Pasadena, CA, USA, 24–27 July 2017; pp. 237–244

18. Marcucci, T.; Petersen, M.; von Wrangel, D.; Tedrake, R. Motion Planning around Obstacles with Convex Optimization. *arXiv* **2022**, arXiv:2205.04422.

19. Tompkins, P.; Stentz, A.; Wettergreen, D. Global path planning for mars rover exploration. In Proceedings of the 2004 IEEE Aerospace Conference (IEEE Cat. No. 04TH8720), Big Sky, MT, USA, 6–13 March 2004; pp. 801–815.

20. Rekleitis, I.; Bedwani, J.-L.; Dupuis, E.; Allard, P. Path planning for planetary exploration. In Proceedings of the 2008 Canadian Conference on Computer and Robot Vision, Windsor, ON, Canada, 28–30 May 2008; pp. 61–68.

21. Garcia, A.; Barrientos Cruz, A.; Medina, A.; Colmenarejo, P.; Mollinedo, L.; Rossi, C. 3D Path planning using a fuzzy logic navigational map for Planetary Surface Rovers. European Space Agency-Automation and Robotics (A&R) Section. In Proceedings of the 11th Symposium on Advanced Space Technologies in Robotics and Automation, ESTEC Conference Centre, 12–14 April 2011; ESA/ESTEC: Noordwijk, The Netherlands, 2011.

22. Roncagliolo, P. Evaluation and Implementation of Modern Path Planning Algorithms for Planetary Exploration Rovers. Master's Thesis, Univeristy of Genoa, Genoa, Italy, 2020. [CrossRef]

23. Pérez-del-Pulgar, C.J.; Sánchez, J.R.; Sánchez, A.J.; Azkarate, M.; Visentin, G. Path planning for reconfigurable rovers in planetary exploration. In Proceedings of the 2017 IEEE International Conference on Advanced Intelligent Mechatronics (AIM), Munich, Germany, 3–7 July 2017; pp. 1453–1458.

24. Yu, X.; Wang, P.; Zhang, Z. Learning-based end-to-end path planning for Lunar Rovers with safety Constraints. *Sensors* **2021**, *21*, 796. [CrossRef]

25. Sutoh, M.; Otsuki, M.; Wakabayashi, S.; Hoshino, T.; Hashimoto, T. The right path: Comprehensive path planning for lunar exploration rovers. *IEEE Robot. Autom. Mag.* **2015**, *22*, 22–33. [CrossRef]

26. Blackmore, L.; Kuwata, Y.; Wolf, M.T.; Assad, C.; Fathpour, N.; Newman, C.; Elfes, A. Global reachability and path planning for planetary exploration with montgolfiere balloons. In Proceedings of the 2010 IEEE International Conference on Robotics and Automation, Anchorage, AK, USA, 3–7 May 2010; pp. 3581–3588

27. Jiang, J.; Zeng, X.; Guzzetti, D.; You, Y. Path planning for asteroid hopping rovers with pre-trained deep reinforcement learning architectures. *Acta Astronaut.* **2020** *1*, 265–279. [CrossRef]

28. Tzanetos, T.; Aung, M.; Balaram, J.; Grip, H.F.; Karras, J.T.; Canham, T.K.; Kubiak, G.; Anderson, J.; Merewether, G.; Starch, M.; et al. Ingenuity Mars Helicopter: From Technology Demonstration to Extraterrestrial Scout. In Proceedings of the 2022 IEEE Aerospace Conference (AERO), Big Sky, MT, USA, 5–12 March 2022; pp. 1–19.

29. Zavoli, A.; Federici, L. Reinforcement learning for robust trajectory design of interplanetary missions. *J. Guid. Control Dyn.* **2021**, *44*, 1440–1453. [CrossRef]

30. Marcucci, T.; Umenberger, J.; Parrilo, P.A.; Tedrake, R. Shortest paths in graphs of convex sets. *arXiv* **2021**, arXiv:2101.11565v3.

31. Noreen, I.; Khan, A.; Habib, Z.T. A comparison of RRT, RRT* and RRT*-smart path planning algorithms. *Int. J. Comput. Sci. Netw. Secur. (IJCSNS)* **2011**, *16*, 20.

32. Warren, C.W.E.F. Global path planning using artificial potential fields. In Proceedings of the 1989 IEEE International Conference on Robotics and Automation, Scottsdale, AZ, USA, 14–19 May 1989; pp. 316–317.

33. Ferguson, D.; Likhachev, M.; Stentz, A. A guide to heuristic-based path planning. In Proceedings of the International Workshop on Planning under Uncertainty for Autonomous Systems, International Conference on Automated Planning and Scheduling (ICAPS), Monterey, CA, USA, 5–10 June 2005; pp. 9–18.

34. Daniel, K.; Nash, A.; Koenig, S.; Felner, A. Theta*: Any-angle path planning on grids. *J. Artif. Intell. Res.* **2010**, *39*, 533–579. [CrossRef]

35. Otte, M.W.T. *A Survey of Machine Learning Approaches to Robotic Path-Planning*; University of Colorado at Boulder: Boulder, CO, USA, 2015.
36. Zhang, J.; Xia, Y.; Shen, G. A novel learning-based global path planning algorithm for planetary rovers. *Neurocomputing* **2019**, *361*, 69–76. [CrossRef]
37. Zhao, Y.; Zhang, Y.; Wang, S. A Review of Mobile Robot Path Planning Based on Deep Reinforcement Learning Algorithm. *Proc. J. Phys. Conf. Ser.* **2021**, *2138*, 1. [CrossRef]
38. Pflueger, M.; Agha, A.; Sukhatme, G.S. Rover-IRL: Inverse reinforcement learning with soft value iteration networks for planetary rover path planning. *IEEE Robot. Autom. Lett.* **2019**, *4*, 1387–1394. [CrossRef]
39. Markus, W.; Peter, O.; Ingmar, P. Rover-IRL: Maximum Entropy Deep Inverse Reinforcement Learning. *arXiv* **2015**, arXiv:1507.04888.
40. Ratliff, N.D.; Silver, D.; Bagnell, J.; Andrew, T. Learning to search: Functional gradient techniques for imitation learning. *Auton. Robot.* **2009**, *27*, 25–53. [CrossRef]
41. Ahmadi, M.; Ono, M.; Ingham, M.D.; Murray, R.M.; Ames, A.D. Risk-averse planning under uncertainty. In Proceedings of the 2020 American Control Conference (ACC), Denver, CO, USA, 1–3 July 2020; pp. 3305–3312.
42. Tamar, A.; Wu, Y.; Thomas, G.; Levine, S.; Abbeel, P. Value iteration networks. *arXiv* **2016**, arXiv:1602.02867.
43. Ma, N.; Wang, J.; Liu, J.; Meng, M.Q.-H. Conditional Generative Adversarial Networks for Optimal Path Planning. *IEEE Trans. Cogn. Dev. Syst.* **2021**, *14*, 662–671. [CrossRef]
44. Mirza, M.; Osindero, S. Conditional generative adversarial nets. *arXiv* **2014**, arXiv:1411.1784.
45. Ronneberger, O.; Fischer, P.; Brox, T. U-net: Convolutional networks for biomedical image segmentation. In Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention, Munich, Germany, 5–9 October 2015; pp. 234–241.
46. Zhang, T.; Wang, J.; Meng, M.Q.-H. Generative Adversarial Network based Heuristics for Sampling-based Path Planning. *arXiv* **2020**, arXiv:2012.03490.
47. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
48. Soboleva, N.; Yakovlev, K. GAN Path Finder: Preliminary Results. In Proceedings of the Joint German/Austrian Conference on Artificial Intelligence (Künstliche Intelligenz), Kassel, Germany, 23–26 September 2019; pp. 316–324.
49. Isola, P.; Zhu, J.-Y.; Zhou, T.; Efros, A.A. Image-to-image translation with conditional adversarial networks. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 1125–1134.
50. Dalmia, S. Constraint-Based Circuit Design Using Generative Adversarial Networks. Master's Thesis, Rijksuniversitait Groningen, Groningen, The Netherlands, 2020.
51. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial networks. *Commun. ACM* **2020**, *63*, 139–144. [CrossRef]
52. Pan, Z.; Yu, W.; Yi, X.; Khan, A.; Yuan, F.; Zheng, Y. Recent progress on generative adversarial networks (GANs): A survey. *IEEE Access* **2013**, *7*, 36322–36333. [CrossRef]
53. Jabbar, A.; Li, X.; Omar, B. A survey on generative adversarial networks: Variants, applications, and training. *ACM Comput. Surv. (CSUR)* **2021**, *54*, 1–49. [CrossRef]
54. Liu, Y.; Wang, Y.; Deng, L.; Wang, F.; Liu, F.; Lu, Y.; Li, S. A novel in situ compression method for CFD data based on generative adversarial network. *J. Vis.* **2019**, *22*, 95–108. [CrossRef]
55. Dash, A.; Ye, J.; Wang, G. A review of Generative Adversarial Networks (GANs) and its applications in a wide variety of disciplines–From Medical to Remote Sensing. *arXiv* **2021**, arXiv:2110.01442.
56. Aggarwal, C.C. *Neural Networks and Deep Learning*, 1st ed.; Springer: Berlin/Heidelberg, Germany, 2018.
57. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 2012; Volume 25.
58. Radford, A.; Metz, L.; Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv* **2015**, arXiv:1511.06434.
59. Liu, M.-Y.; Breuel, T.; Kautz, J. Unsupervised image-to-image translation networks. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 2017; Volume 30.
60. Lee, T.-K.; Baek, S.-H.; Choi, Y.-H.; Oh, S.-Y. Smooth coverage path planning and control of mobile robots based on high-resolution grid map representation. *Robot. Auton. Syst.* **2011**, *59*, 801–812. [CrossRef]
61. Gretton, A.; Borgwardt, K.M.; Rasch, M.J.; Schölkopf, B.; Smola, A. A kernel two-sample test. *J. Mach. Learn. Res.* **2012**, *13*, 723–773.
62. Borji, A. Pros and cons of gan evaluation measures. *Comput. Vis. Image Underst.* **2019**, *179*, 41–65. [CrossRef]
63. Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; Wojna, Z. Rethinking the inception architecture for computer vision. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 2818–2826.
64. Duchi, J.; Hazan, E.; Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.* **2011**, *12*, 7.
65. Chen, Q.; Koltun, V. Photographic image synthesis with cascaded refinement networks. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 1511–1520.

66. Wang, T.-C.; Liu, M.-Y.; Zhu, J.-Y.; Tao, A.; Kautz, J.; Catanzaro, B. High-resolution image synthesis and semantic manipulation with conditional gans. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 8798–8807.

67. Richter, S.R.; Al Haija, H.A.; Koltun, V. Enhancing photorealism enhancement. *IEEE Trans. Pattern Anal. Mach. Intell.* **2022**, 1. [CrossRef]