On finding low complexity heuristics for path planning in safety relevant applications

Krutsch Robert, Intel

***Abstract*— In many robotics applications path planning has safety implications that need to be addressed and understood. Approaches based purely on learning algorithms have today no strong guarantees that the path found, even given perfect environment model, is safe. In contrast, search based methods have strong theoretical guarantees but are significantly slower and hard to parallelize. In this paper we present a method of obtaining heuristics for search based algorithms targeting to reduce the search complexity by combining the strengths of the two paradigms. We show that a complexity reduction of more than 30% is achievable with less than 1% drop in path optimality. As a consequence of the complexity reduction we also measure a performance boost of more than 30%.**

I. INTRODUCTION

Comprehensive environment models are an essential part of robotics application and driver assistance. A significant number of publications [1,2,3,4] have concentrated on developing strategies for producing predictive environment models that incorporate information from multiple sensors. The typical approach in robotics has been based on the sense – plan – act paradigm, where environment models constructed based on the sensed information are used in the planning phase. Other paradigms, based on reactive control need also a representation of the environment yet concentrate more on reaction timeliness than on environment model correctness [5].

Path planning has been studied extensively in the past years and many of the algorithms have been implemented and tested in various robotics applications [6]. Most of the modern strategies perform well in environments where no adversarial interaction occur or/and where sensing is still acceptable accurate. In driver assistance systems for instance path planning based on a grid based representation of the environment is very often used. Such a representation allows the designer to choose from multiple methods of path planning as for instance probabilistic roadmap, rapidly exploring random trees, search based methods (e.g. variants of A*) , reinforcement learning methods etc.

In automotive application functional safety is one of the critical aspects needed to be considered in system design. The ISO26262 [7] gives guidelines and requirements at the system level, defining the framework on how hardware and software is built. The framework does not mandate specific implementation features, so the designer has the freedom and the responsibility of how to achieve the functional safety for the final product. Typically, the system safety goals are achieved by decomposing the system and applying redundancy and fault detection mechanisms. In such systems multiple environment models are used, different modalities for path planning are employed to achieve the system level safety goals. One possible architecture is presented in Figure 1, where two different environment models are used, different possible paths are obtained and then the cross check between environment paths is performed. Note that typically a significant number of paths is produced by each path planner (e.g. 100), with different characteristics and after validating this paths one both environment models one path will be selected for the execution.

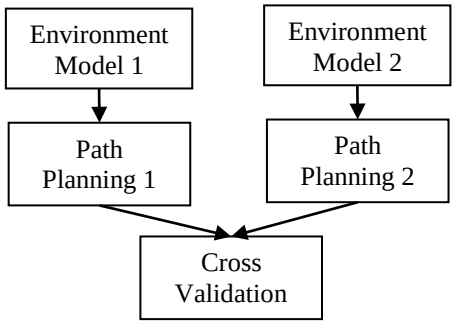

Figure 1: Functional safety decomposition of the planning and environment modeling

The algorithmic diversity is a very important part of the architecture that allows such a decomposition to be valid. For instance, different methods of path planning can be used or different heuristics can be employed. For the environment model different settings can be used such that one environment model has a longer aggregation time while the other is more refined for quick environment changes.

In this paper we will concentrate mainly on search based path planning and propose methods on how to reduce the search complexity by producing search heuristics based on convolutional neural networks. We will study also the tradeoff between complexity reduction and optimality. It is expectable that such heuristics might louse the optimality guarantees yet in such systems the optimality is in itself very hard to quantify. For autonomous driving applications the optimality does not only imply the shortest path but might be formulated as a combination of shortest path, lowest energy consumption and comfort.

Due to the increased number of paths that need to be found a reduction in search complexity is highly desirable. Typically neural networks run best on accelerators while search based schemes have better run times on cores

allowing for a pipelining of producing heuristics and searching a path as depicted in Figure 2. To be noted is also that if on the target hardware multiple neural network compute accelerators and cores are available the process can be parallelized and so provide more diversity to the solution and potentially faster runtime.

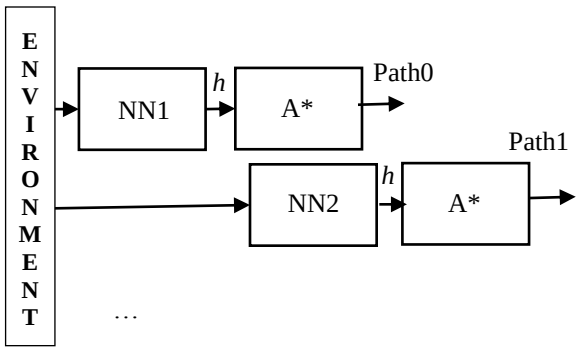

Figure 2: Pipelining the neural network execution with the A* search. Each neural network (NN1, NN2...) produces a new heuristic denoted as *h* that is used by A* to produce the paths Path0, Path1 etc.

The search scheme based on A* has strong guarantees since it test each cell if it is occupied or not and is as such guaranteed not to have in the path produced any obstacle, as long as the environment model is accurate. Mitigation of the sensorial imperfections and adversarial behaviors is typically based on continuous and timely replanting. The practical assumption is that the sensors get more accurate as the robot is approaching the obstacle and that the physical constraints of the other agents prohibits them from changing strategies with a very low time granularity (the behavior is stationary in the re-planning interval). This assumptions are reasonable for most use-cases and can be fairly accurate for instance in parking use-cases. Approaches such in [7] have no guarantee on correctness of the path and generally reinforcement learning approaches are very susceptible to catastrophic forgetting and can be used in practice only after a validation step where correctness and physical constraints are taken into consideration.

II. Experimental Setup

For our experiments we have constructed a dataset of 500000 occupancy grids of size 32x32 where each cell is marked either as free or as occupied (the probability of a cell to be occupied is 0.4, an example is provided in Figure 3). The starting location and the goal are encoded also as a grid with one in the respective locations and zero in the rest. An agent is placed in the start location and is able to take an action from the action set *A = {left, right, up, down, left-up, right-up, left-down, right-down}*. A move up, down, left or right is associated with a cost of one while a move diagonally has a cost of 1.41. The path cost is the summation of the cost of the individual moves. The data set contains also the Euclidian distance heuristic encoded as grid and the optimal heuristic that is found after running the A* algorithm from every location to the goal. As can be expected, computing the optimal heuristic is the most demanding part in the dataset generation and takes a few weeks on our system setup. We also keep the number of opened nodes by A* for the Euclidian distance heuristic.

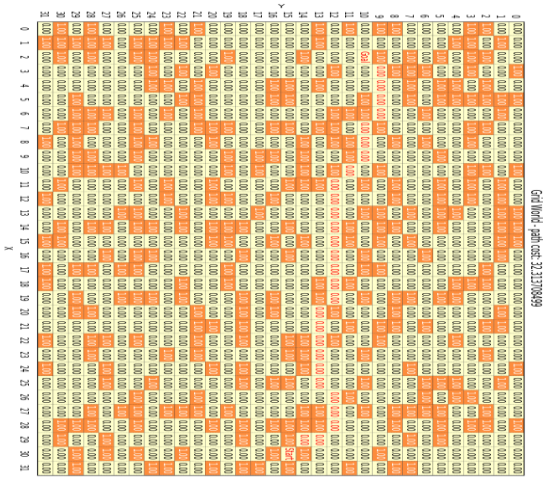

Figure 3: Occupied grid cells are marked with orange, the path from start to goal is marked with light orange

Comparing the complexity associated with two heuristics implies comparing the number of open nodes during the A* search. The second dimension of interest is the optimality of the path; it is well known that in case the heuristic overestimates the cost to the goal there are no optimality guarantees. We have used the following two performance criteria to assess the quality of the new heuristic:

$$C[\%] = \frac{\sum(OE-ONN)}{\sum OE} * 100 \quad (1)$$

$$O[\%] = \frac{\sum(DNN-OD)}{\sum OD} * 100 \quad (2)$$

, WHERE:

OE = Number of opened nodes by A* given Euclidian distance as heuristic
ONN = Number of opened nodes by A* given output of the neural network as heuristic
OD = optimal path length
DNN = path length found by A* given the heuristic obtained with a neural network
The summation if (1) and (2) are over the test dataset.

In our experiments we obtain new heuristics with a neural network that has as input three channels (Figure 4):

- Start – Goal channel that has the same dimensions as the grid with one in the start and goal location and zero everywhere else
- Grid channel that has one in every cell where an obstacle is present and zero where no obstacle is present
- Euclidian distance channel that has the Euclidian distance from each cell to the goal. This channel can be obtained in an offline computation.

The network output is a matrix of the same size as the grid in that each element represents an estimate of the path length from that specific location to the goal. This matrix is used as

a heuristic for A* and used to compute the performance metrics presented in (1) and (2).

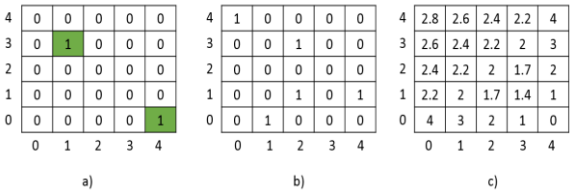

Figure 4: Input channels to the neural network: a) Start-Stop channel, b) Grid Channel, c) Euclidian distance channel

III. Obtaining new heuristics

The goal is to obtain a new heuristic that is less complex and that loses as less as possible from the optimality characteristic. Complexity and optimality cannot be obtained directly by a differential function that can be introduced in the neural network so we will concentrate on finding cost functions that are correlated to this indicators such that optimizing this cost functions will improve the two hidden indicators.

In [8] Theorem 12 from Chapter 3 indicates that given two admissible heuristics $h_2(n)$ and $h_1(n)$ for that $h_2(n) \geq h_1(n)$ for every *n* imply that an A* search using h_2 is more efficient than A* search using h_1 (this is intuitively obvious since we have a better approximate of the optimal heuristic). It is also shown in [8], Chapter 7, that when non-admissible heuristics are used the accuracy with that we approximate the optimal path is not the most critical aspect since less accurate estimations can lead to more informed heuristics (this is somehow surprising but has as background the idea that overestimates outside the path can be benefic to the search complexity). Given the insides above and the fact that the neural network is not guaranteed to produce an admissible heuristic it is clear that a simple formulation of the cost function where we try to approximate as much as possible the optimal path is prone to produce uncorrelated results to the complexity and optimality metrics.

In our experiments we test the following cost functions:

$$L_1 = \frac{|h_o - h|}{|h_e - h|} \quad (3)$$

$$L_2 = |h_o - h| + Relu(|h_e - h|) \quad (4)$$

, where:

h_o- The optimal heuristic, true distance to the goal

h - Output of the neural network

h_e - Euclidian distance heuristic

The denominator in L_1 encourages the output of the neural network to be greater than the initial Euclidian heuristic while the numerator in L_1 is responsible for guiding the network towards the optimal policy and penalizes severe overestimates. We use in the computation of the loss all cells that are not occupied and give equal weighting to cells that are on path and off path.

L_2 is constructed out of two terms , the first term encourages the output of the neural network to be close to the optimal heuristic while the second term pushes the output to be higher than the Euclidian distance heuristic.

We have experimented also with several other loss function formulations where the terms of the loss function are weighted or where we treat the loss differently if it is off the minimal path or on the minimal path (inspired by the fact that overestimate on path can be highly detrimental to complexity). The results obtained where similar to the vanilla formulation from (3) and (4) but the training procedure was observed to be more unstable.

The neural network architecture used in our experiments is depicted in Figure 5 and has a contracting part and an expending part similar to topologies employed in pixel labeling applications. The intuition behind comes from [9] where the grid is split into blocks and the planning is done hierarchically, first between blocks and then inside blocks. Similarly, the network contracts the information into blocks given by the receptive field of the convolution and pooling and in the later stage incorporates more fine grained information. The parameters for the topology are detailed in Table 1. As it is custom we have split the data set into training, validation and test and optimized the network with the Adam optimizer with learning rate 1e-4. The batch size used throughout the experiments is 128.

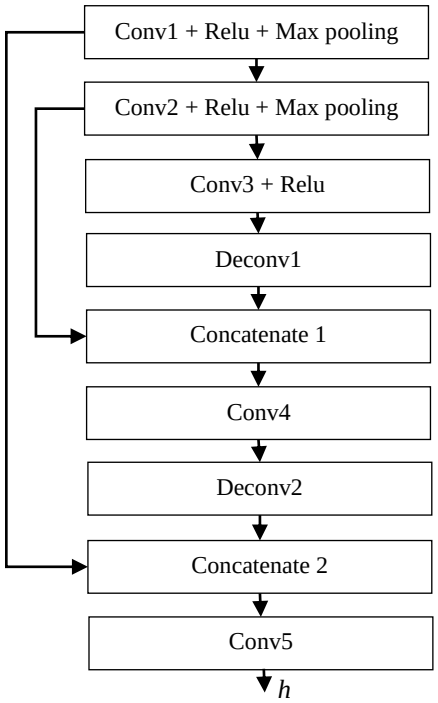

Figure 5: Neural network topology used in the experiments. Conv1:5 are two dimensional convolution layers while Deconv1:2 are two dimensional transpose convolutions

Layer	Kernel Size	Stride	Out Channels
Input	-	-	3
Conv1+Relu	3x3	1	32
Max Pooling	3x3	2	32
Conv2+Relu	3x3	2	16
Max Pooling	3x3	2	32

Conv3+Relu	3x3	2	8
Deconv1	3x3	2	8
Concatenate 1	-	-	24
Conv4	3x3	1	8
Deconv2	3x3	2	8
Concatenate 2	-	-	40
Conv5	3x3	1	1

Table 1: Neural network parameters

IV. Results

One of the first questions that arise is if the loss functions are correlated with the algorithmic complexity and optimality of the A* algorithm. In Figure 6 and 7 the training process statistics are shown (last 105 epochs of the training), the correlation between (1) , (2) and the L_1 and respectively L_2 loss functions are immediate to spot since as the loss decreases the complexity reduction has an increasing trend while the loss in optimality decreases. The metrics of (1) and (2) are computed over the validation test that represents about 5% of the overall dataset.

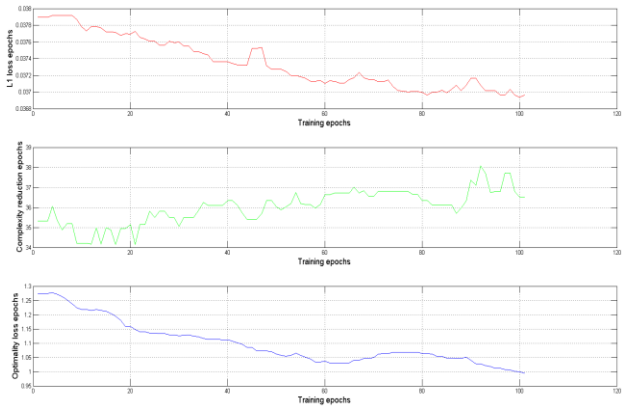

Figure 6: Upper plot is the L_1 loss; middle plot the complexity reduction (1); lower plot the optimality lost (2). The x axis for all plots represents the training epoch

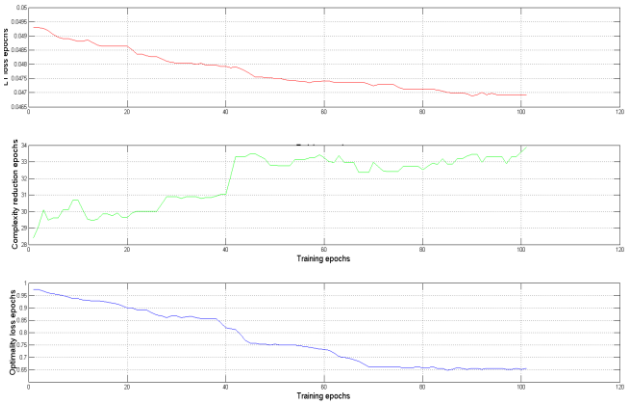

Figure 7: Upper plot is the L_2 loss; middle plot the complexity reduction (1); lower plot the optimality lost (2). The x axis for all plots represents the training epoch

The metrics (1) and (2) for the two loss functions are presented in Table 2 and are obtained based on the test subset that has 128000 grid samples. The computational time gain as measured on our system is presented in the 3rd column of the Table 2 and it is based on the formula (5). As can be observed the computational time reduction follows closely the complexity reduction. In our experiments we run first the A* algorithm for the heuristic obtained with the neural network and after the search based on the Euclidian distance heuristic. While the data cache is flushed in between the runs of A* the instruction cache is not flushed making it highly likely that the run of the Euclidian distance based search benefits from a better hit ratio, nevertheless the runtime is reduced by more than a 3rd.

$$T[\%] = \frac{\sum(T_{old} - T_{new})}{\sum T_{old}} * 100 \quad (5)$$

, where:
T_{old} – Runtime for the Euclidian distance heuristic A*
T_{old} – Runtime for the neural network heuristic A*

Loss Function	C[%]	O[%]	T[%]
L_1	37.64	1.01	36.32
L_2	35.82	0.71	32.45

Table 2: Performance metrics for L_1 and L_2

V. Conclusion

In this paper we have shown a novel method of obtaining new heuristics for A* that have lower complexity than the starting Euclidian distance with a minimal loss in complexity. The loss functions used follow the intuitions obtained from theoretical results obtained in [8]. We show that such indirect cost functions are correlated to the hidden variables in the A* algorithm and that joint optimization is possible.

Future work will investigate higher dimensional graphs and introduce robot poses in the search process.

References

[1] Florian Homm et al.,Efficient Occupancy Grid Computation on the GPU with Lidar and Radar for Road Boundary Detection. IEEE Intelligent Vehicles Symposium, 2010, San Diego, 10.1109/IVS.2010.5548091.

[2] Dominik Nuss, Doctoral Dissertation: A Random Finite Set Approach for Dynamic Occupancy Grid Maps with Real-Time Applications, 2016, University Ulm

[3] Alberto Elfs et al., Using Occupancy Grids for Mobile Robot Perception and Navigation, IEEE Computer Volume:22 Issue:6, 1989, 10.1109/2.30720

[4] Sebastian Thrun. Learning Occupancy Grids with Forward Models. International Conference of Intelligent Robots and Systems, 2001, Maui, 10.1109/IROS.2001.977219

[5] Daniel Kappler et al. Real time perception meets Reactive Motion Generation , IEEE Robotics and Automation Latters, 2018, 10.1109/LRA.2018.2795645

[6] Eric Galcern et al. A survey on coverage path planning for robotics Robotics and Autonomous Systems 61 1258-1276, 2013

[7] Aviv Tamar et al. ,Value Iteration Networks, NIPS , 2016

[8] Pearl, Judea. Heuristics: Intelligent Search Strategies for Computer Problem Solving , Michigan, Addison-Wesley, 1984, Chapter 3,7

[9] Adi Botea et al. Near Optimal Hierarchical Path-Finding, Journal of Game Development, Volume 1, Pages 7-28, 2004