IMG-SMP: Algorithm and Hardware Co-Design for Real-time Energy-efficient Neural Motion Planning

Lingyi Huang*
Xiao Zang*
Rutgers University
Piscataway, New Jersey, USA

Yu Gong
Rutgers University
Piscataway, New Jersey, USA

Chunhua Deng†
ScaleFlux Inc
San Jose, California, USA

Jingang Yi
Rutgers University
Piscataway, New Jersey, USA

Bo Yuan
Rutgers University
Piscataway, New Jersey, USA

ABSTRACT

Motion planning is a fundamental and critical task in modern autonomous systems. Conventionally, motion planning is built on uniform sampling that causes long planning procedure. Recently, built upon the powerful learning and representation abilities of deep neural network (DNN), neural motion planners have attracted a lot of attention because of the better biased sampling strategy learned from data. However, the existing NN-based motion planners are facing several limitations, especially the insufficient exploit of critical spatial information and the high computational cost incurred by neural network models. To overcome these limitations, in this paper we propose IMG-SMP, an algorithm and hardware co-design framework for neural sampling-based motion planner. At the algorithm level, IMG-SMP is an end-to-end neural network that can efficiently capture and process the critical spatial correlation to ensure high planning performance. At the hardware level, by properly rescheduling the computing scheme, the dataflow of IMG-SMP architecture can eliminate the unnecessary computations without affecting planning quality. The IMG-SMP hardware accelerator is implemented and synthesized using CMOS 28nm technology. Evaluation results across different planning tasks show that our proposed hardware design achieves order-of-magnitude improvement over CPU and GPU solutions with respect to planning speed, area efficiency and energy efficiency.

CCS CONCEPTS

• **Computer systems organization → Embedded hardware**.

*Both authors contributed equally to this research.
†This work was done when the author was with Rutgers University.

This work is partially funded by National Science Foundation Award CNS-1932370.

GLSVLSI '22, June 6–8, 2022, Irvine, CA, USA

ACM ISBN 978-1-4503-9322-5/22/06…$15.00
https://doi.org/10.1145/3526241.3530367

KEYWORDS

VLSI, Hardware architecture, Motion planning

ACM Reference Format:
Lingyi Huang, Xiao Zang, Yu Gong, Chunhua Deng, Jingang Yi, and Bo Yuan. 2022. IMG-SMP: Algorithm and Hardware Co-Design for Real-time Energy-efficient Neural Motion Planning. In *Proceedings of the Great Lakes Symposium on VLSI 2022 (GLSVLSI '22), June 6–8, 2022, Irvine, CA, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3526241.3530367

1 INTRODUCTION

Motion planning, which targets to compute a high-quality and collision-free path from a start configuration to a goal configuration under the given environment, is a fundamental and important task in autonomous systems. Among various motion planning approaches, sampling-based motion planner (SMP) is a very popular solution that has been widely used in many applications, such as robot arm manipulation, drone navigation and autonomous driving. However, the sampling process in most of the existing SMPs, e.g., RRT [1] and RRT* [2], is typically performed in a uniform way, thereby inevitably causing high exploration costs and slow convergence for finding the near-optimal path solutions.

To solve this issue, recently several learning-based SMPs [3–5] have been proposed to provide biased sampling schemes to find near-optimal path in a fast way. Benefited from the strong learning and representation abilities of deep neural networks (DNNs), these neural planners are able to bias the sampling towards the promising waypoints that most probably lie on the shortest collision-free paths. Consequently, both planning cost (in term of planning time) and path cost (in term of path length) can now be significantly reduced.

Despite these encouraging benefits, the widespread deployment of neural SMPs are still hindered by some limitations. First, in order to extract the map information, most neural planners [3][5] adopt contractive auto-encoder (CAE) [6] to encode the spatial environment into a vertorized embedding. One key drawback of such CAE-based solution is that once the obstacles are embedded into an individual latent space, the critical spatial correlation between the obstacles and the robot state, which is not contained in the latent space, cannot not be fully exploited or leveraged in the learning procedure, thereby causing limited planning performance. Second, using DNN is not free – modern DNN models are typically computation intensive. Considering the equipped DNN models need to be executed for every sampling operation, the overall computational costs can be very high, thereby posing a severe challenge

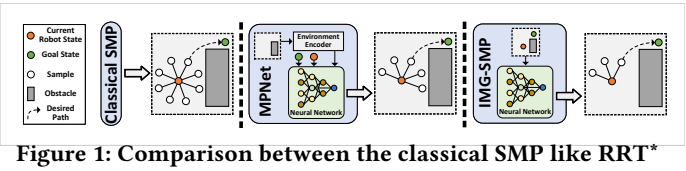


Figure 1: Comparison between the classical SMP like RRT* [2], existing NN-Based SMP like MPNet [3] and IMG-SMP proposed in this paper. Taking the raw image of the working status as the input of SMP can effectively capture the spatial correlation between obstacles and the robot state, thereby significantly improving the accuracy of biased sampling.

for the deployment of neural SMPs in the time-constrained power-constrained embedded systems.

To overcome these limitations, in this paper we propose IMG-SMP, an algorithm and hardware co-design framework towards real-time energy-efficient neural motion planning. The contributions of IMG-SMP are two-fold. At the algorithm level, IMG-SMP adopts a single CAE-free neural network, which directly takes the raw image of the entire working status, including the information of *both* the planning environment and the robot/goal states, as the input. As indicated in Fig. 1, such end-to-end learning procedure ensures that the obstacles and the robot state can be embedded in the same latent space, and therefore their spatial correlation can now be efficiently captured, which brings a remarkable improvement on the planning performance. At the hardware level, we further develop a customized hardware architecture to reap the algorithmic benefits of IMG-SMP and accelerate the execution in an energy-efficient way. By properly rescheduling the computing scheme, the proposed hardware architecture can eliminate all of redundant computations without affecting planning quality, thereby significantly improving the planning speed. To demonstrate the advantage of our proposed solutions, we design and implement IMG-SMP hardware accelerators using CMOS 28nm technology. Evaluations over different planning maps demonstrates that the proposed customized hardware for the neural planners significantly outperforms the CPU and GPU-based planner with respect to planning speed, area efficiency and energy efficiency.

2 THE PROPOSED IMG-SMP: ALGORITHM

2.1 Problem Statement

Denote a state space as $C \in \mathbb{R}^d$, where d is the dimension of the state space. The obstacle space and free space can be then represented as $C_{obs} \in C$ and $C_{free} = C \backslash C_{obs}$, respectively. Also, the start state and goal state of the robot can be denoted as $c_{start} \in C_{free}$ and $c_{goal} \in C_{free}$, respectively. The sampling-based motion planner aims to find a collision-free path solution $\mathcal{S}$ that connects the c_{start} and c_{goal}, i.e., $\mathcal{S}_0 = c_{start}$ and $\mathcal{S}_T = c_{goal}$, where a path is represented as a non-empty list of states $\mathcal{S} : [0, T] \in C$. Here a path is defined as collision-free if the trajectory connecting all consecutive states in $\mathcal{S}$ lies entirely in the free space C_{free}.

2.2 Key Idea

Motion Planning as Image Prediction. As analyzed in Section 1, the classical SMP approach and the existing neural planners suffer from extensive sampling and limited planning performance, respectively. To solve these problems, our proposed IMG-SMP aims to develop a fast end-to-end NN-based planner, which can learn

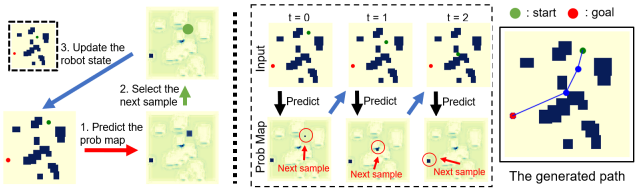


Figure 2: The 2-D planning task from an image prediction perspective. Note that the robot/goal states are denoted as green/red dots, while the obstacles are represented as dark blue patches. The left part of the figure summarizes the high-level idea of determining the next robot state via predicting the probability map. The right part describes the whole process of computing a feasible path.

and process the images directly (See Fig. 1). To be specific, a 2D planning task can be essentially interpreted as an image prediction task, where the environment along with the robot/goal states can be represented by an image (see Fig. 2). From this perspective, the entire motion planning task can be viewed as an iterative procedure of the following three steps. First, given a well-trained image prediction model and the current task image, the neural model predicts a probability map that shares the same size as the input (step 1). Afterwards, the collision-free sample with the highest probability is selected as the new robot state (step2). The task image is then updated by the selected sample and proceed to the next iteration (step 3). Notice that such 3-step operation is repeated until the robot reaches the desired goal state.

Data Representation. In order to ensure that motion planning can be realized as image prediction task, the corresponding data representation is needed. To be specific, the input to the image prediction network is a 1-channel image that describes the 2D planning task, where the image size is the same as the environment size. Here the free space and obstacle space are represented by pixel values -1 and 1, respectively. On the other hand, we denote the entries correspond to the robot and goal state as values -0.33 and 0.33, respectively. The output of the network is also a 1-channel image that shares the same size as the input, which represents each entry's probability of being the next sample by a pixel value between $0 - 1$. In addition, for the ground-truth probability maps, the pixel belongs to the next sample is marked as 1, otherwise 0.

Pixel Augmentation. Considering it is typically challenging for the neural network to capture and learn the single-pixel information that represents robot/goal states, we perform the pixel augmentation, which means also mark the neighbouring pixels of robot/goal states as "1", to enhance the representation of those important information. In this work, the augmentation square of size 3×3 is adopted. Such pixel augmentation is performed on both the input task images and the ground-truth probability maps.

Data Generation. We train and evaluate our network on tasks in three sizes of 2-D environments (32×32, 64×64 and 128×128). For each size, we generate 10 different environments via placing 15 obstacles of different shapes randomly. For each planning task, we use RRT* to compute its collision-free and near-optimal path as the training data. Overall for each size the training data consists of 10 environments with 1000 near-optimal paths in each environment. In the testing phase the neural planner is evaluated via solving 200 random planning tasks for each environment.

2.3 Offline Training Procedure

After performing the above described data preparation, the proposed IMG-SMP can be trained via minimizing the mean squared error (MSE) between the predicted probability maps O and their ground truths $\hat{O}$. Let $\mathcal{G}$ and $\mathcal{I}$ denote the sample predictor and task image, respectively, the optimization goal can be then written as:

$$w^* = \arg\min_{w} MSE(\mathcal{G}(\mathcal{I}; w), \hat{O}), \quad (1)$$

where w represents the network parameters.

Algorithm 1 The Proposed IMG-SMP Neural Motion Planner

1: **Input:** c_{start}, c_{goal}, map information C, sample predictor $\mathcal{G}$
2: **Output:** The path solution $\mathcal{S}$
3: $\mathcal{I} \leftarrow IMG\_generator(c_{start}, c_{goal}, C)$
4: **while** c_{start} not reach c_{goal} **do**
5: $O \leftarrow \mathcal{G}(\mathcal{I})$
6: **for** $c_{next} \in argsort(O, reverse = True)$ **do**
7: **if** $isValid(c_{start}, c_{next}, C)$ **then**
8: $c_{start} \leftarrow c_{next}$
9: $\mathcal{S}.add(c_{start})$
10: **Break**
11: **end if**
12: $\mathcal{I} \leftarrow IMG\_generator(c_{start}, c_{goal}, C)$
13: **end for**
14: **end while**
15: Return $\mathcal{S}$

2.4 Online Path Construction

With the well-trained neural network, we are then able to conduct the path construction by making use of its predicted samples. The details of this online path construction procedure are described in Algorithm 1. Initially, the function *IMG_generator* takes the state pairs and map information as input, and generates the corresponding image (Line 3). The search procedure iterates until the robot states reaches the goal state c_{goal} (Line 4). In each iteration, the neural network takes the current task image $\mathcal{I}$ as input, and outputs the probability map O (Line 5). The entries in the probability map are then sorted in an descending order, according to their probabilities of being the next sample (Line 6). Each sorted entry is visited sequentially (Line 6) and checked whether this is a valid sample (Line 7) or not. To be specific, the function $isValid$ returns True if there is no collision between the newest robot start state c_{start} and the next sample c_{next}, otherwise returns False. If a valid sample is found, the robot moves to the next state (Line 8), appends it to the path solution (Line 9) and stops visiting the following samples (Line 9). The task image $\mathcal{I}$ is then updated by the new start state (Line 12), thereby being prepared for the next iteration (Line 12). The path is returned after a valid solution is found (Line 15).

3 THE PROPOSED IMG-SMP: HARDWARE

Built upon the algorithm described in Section 2.3, in this section we further develop the customized hardware architecture to accelerate the execution of IMG-SMP.

Key Observation. Recall that the neural network used in IMG-SMP serves as the sample generator, which needs to be executed multiple times for biased sampling. Evidently, such repeated executions significantly increase the overall computational cost for the neural planner. Fortunately, our in-depth analysis on the computing

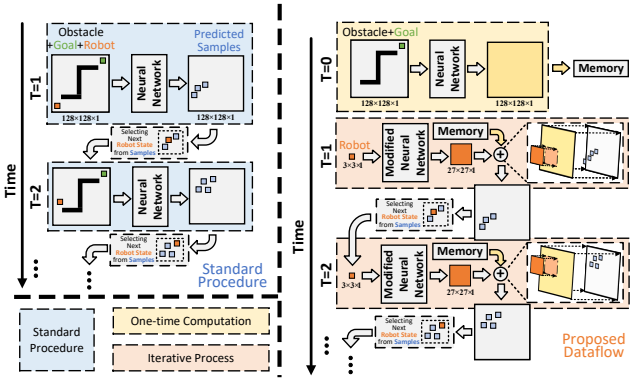


Figure 3: Dataflow for IMG-SMP. (a) Standard computing scheme via straightforward design. (b) Proposed optimized scheme. Here 128×128 map size is used as example.

flow shows the existence of numerous unnecessary computations. To be specific, consider in each prediction iteration the input of the sample predictor $\mathcal{G}$ consists of three information: obstacles, the goal state and the real-time state of the robot. Since the environment and the goal state of the static motion planning tasks are fixed, the only unique information for each iteration's input is the robot state, which only contributes to a very small portion of the input image. Therefore, the predictor indeed takes highly similar images as the inputs across all iterations. In other words, many repetitive computations are performed during the online planning process.

The Proposed Dataflow. Based on this observation, we propose a dataflow to eliminate the unnecessary computations incurred by the planning procedure. As illustrated in Fig. 3, our key idea is to separate the computations that are repeatedly performed across different iterations from the unique computations required in each iteration. To be specific, the originally repeated computation will now be executed only once at the start of the motion planning task, and then the corresponding results will be saved and reused in the future iterations. By this way, in each iteration the proposed hardware accelerator will only focus on performing the distinctive computations, thereby saving a lot of computational costs. Next, we describe the computing procedure in detail.

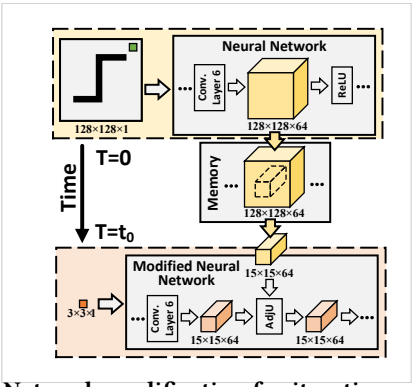


Figure 4: Network modification for iterative prediction. (1) Zero padding for feature maps. (2) Replacement of ReLUs by AdjUs (adjustment units).

One-time Computation. At the beginning of online planning, only the goal state and the map information, without the robot state, are

formed as the input and sent to the neural planner. Then the output activation of each layer, before the processing of the activation function, is saved into memories for future reuse. Specifically, the outputs of all the hidden layers are saved into off-chip DRAM, and the last convolution layer's output is saved into on-chip SRAM.

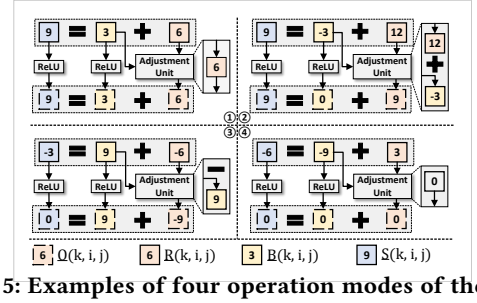


Figure 5: Examples of four operation modes of the adjustment unit.

Network Modification in Iterative Process. After the above described one-time computation, the neural planner will then execute on the input containing robot states in each iteration. As shown in Fig. 4, to guarantee mathematical equivalence, the ReLU activation units are replaced by the specifically designed *adjustment units* (AdjU) (see Fig. 5). To be specific, the output tensor $\underline{O}$ of the adjustment unit can be expressed as:

$$\underline{O}(k,i,j) = \begin{cases} \underline{R}(k,i,j) & \underline{S}(k,i,j) > 0 \text{ and } \underline{B}(k,i,j) > 0 \\ \underline{S}(k,i,j) & \underline{S}(k,i,j) > 0 \text{ and } \underline{B}(k,i,j) < 0 \\ -\underline{B}(k,i,j) & \underline{S}(k,i,j) < 0 \text{ and } \underline{B}(k,i,j) > 0 \\ 0 & \underline{S}(k,i,j) < 0 \text{ and } \underline{B}(k,i,j) < 0 \end{cases} \quad (2)$$

where $\underline{R}$, $\underline{B}$, $\underline{S}$ and $\underline{O}$ are four 3-rd order tensors with the same size of $K \times I \times J$, $k \in [1, K]$, $i \in [1, I]$ and $j \in [1, J]$, respectively. Here $\underline{R}$ is the output activation generated in the iteration, $\underline{B}$ is the pre-stored results from initial one-time computation, and $\underline{S}$ is the sum of $\underline{R}$ and $\underline{B}$. After that, the output of the adjustment units will then be added with these pre-stored results to recovery the correct activation outputs. Fig. 6 illustrates the benefits of the proposed dataflow. Here for the example motion planning tasks on 32×32, 64×64 and 128×128 grid maps, our proposed approach can bring a very significant saving in computational costs (in term of number of multiplications).

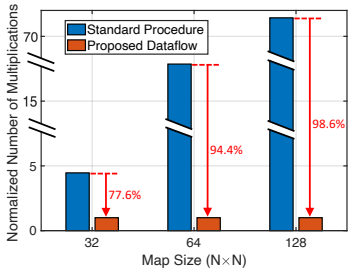


Figure 6: The numbers of required multiplications for one iteration of IMG-SMP on maps with size of 32×32, 64×64 and 128×128.

Overall Architecture & Processing Scheme. Based on the above described dataflow, the overall hardware architecture can be then developed. As shown in Fig.7, the proposed IMG-SMP

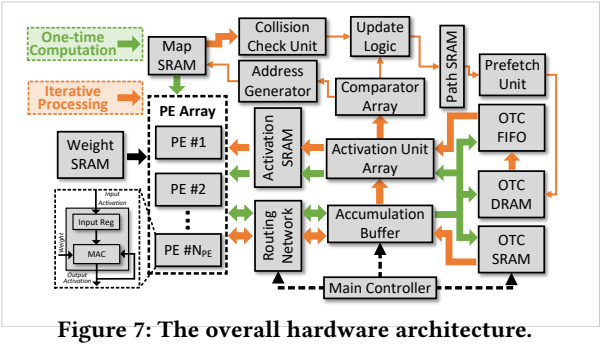


Figure 7: The overall hardware architecture.

hardware accelerator can be operated in two modes. In the *one-time computation mode*, at the beginning of the planning task, the map information and the goal state are loaded from the Input SRAM, and then, together with the DNN weights, they are sent into an array of N_{PE} processing elements (PEs) for DNN inference. The final outputs of PE arrays will be sent into the One-time Computation (OTC) DRAM. After the PE array has finished processing the current portion of inputs, the activation, which is loaded from the OTC DRAM and processed by the activation units (ActU), will be sent into the Activation SRAM, where the used-up data will be overwritten. Notice in order to support the requirements of different applications, each activation unit can be reconfigured to act as Rectified Linear Unit (ReLU), Sigmoid function unit, or the adjustment unit.

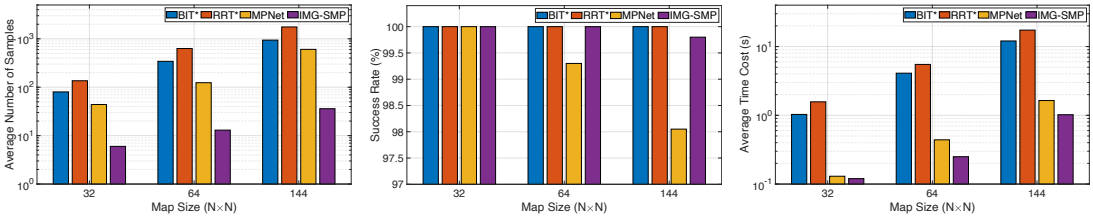
Iterative Processing Mode. In the subsequent iteration, the neural planner is operated in the iterative processing mode, which means only the impact of the current robot state will be calculated. At the beginning of each iteration, with the coordinates of the current robot state, the prefetch unit will produce the corresponding reading addresses that will be used to access OTC DRAM. After the processing of PE arrays, the adjustment units will be activated to provide the correct activation results. The output of the last layer will be added into the corresponding results from one-time computation to obtain the probability maps. The predicted samples, according to their corresponding probabilities, will be sorted by a comparator array, and their qualifications will also be checked in sequence via a specialized collision check unit. With the sequential visiting of the sorted samples, the foremost qualified sample will be finally selected as the robot state at the next timestamp by an update logic. The coordinates of the selected sample will then be stored in the path SRAM and sent to the prefetch unit to start the next iteration.

4 EVALUATION

4.1 Algorithm Evaluation

Network Architecture. We adopt a 12-layer convolutional neural network model with 3×3 kernel size. The first 11 convolutional layers has 64 filters and the last one has only 1 filter. Each convolutional layer is followed by batch normalization, and after the last convolution we apply a dropout layer with 10% drop rate. Zero padding is used in all the convolutional layers to keep the same dimension. The entire model is trained by ADAM optimizer [7] for 200 epochs with learning rate $1e^{-3}$.

Baselines and Implementation Details. We compare our IMG-SMP with three other baseline planners (BIT*, RRT* and MPNet). Here BIT* and RRT* are the classical planners, and MPNet is the state-of-the-art neural planners. The neural networks of MPNet

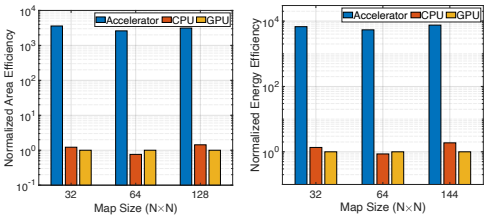


(a) The average number of samples. (b) Success rate. (c) Time cost.

Figure 8: The comparison with baseline planners, with respect to the average number of samples, success rate and time cost of finding a feasible path under 2-D environments of different sizes.

and our IMG-SMP are implemented using PyTorch framework, and all the motion planners are implemented with Python.

Experimental results. Fig. 8 shows the planning performance of different motion planners, with respect to the average number of samples, success rate and time cost, respectively. It is seen that that our proposed IMG-SMP outperforms all the other baselines significantly with respect to planning speed and number of samples with very similar success rate. For instance, IMG-SMP is at least 12× faster than the classical motion planners. Meanwhile, IMG-SMP achieves both faster speed and higher success rate than the state-of-the-art neural planner MPNet. In addition, IMG-SMP uses the fewest samples to compute a path, which demonstrates its strong capability of predicting the feasible samples.



(a) Area Efficiency. (b) Success rate.

Figure 9: The comparison with the CPU-based and GPU-based implementations w.r.t. area and energy efficiency.

4.2 Hardware Evaluation

Accelerator Configuration. To demonstrate the effectiveness of the proposed hardware architecture, we develop a design example for IMG-SMP hardware accelerator, which can perform planning tasks on maps with size up to 128×128. Here the overall architecture consists of 256 16-bit multipliers and 876.3 KB on-chip memory.

EDA Tools and Synthesis Results. A bit-accurate cycle-accurate simulator is designed in System Verilog to model the high-level behavior of the IMG-SMP hardware architecture, which is then used to verify the functional correctness of the RTL model implemented by Verilog HDL. The verified model is synthesized by Synopsys Design Compiler with CMOS 28nm technology. The synthesis report shows our design example occupies 4.2 mm^2 and consumes 206 mW under 800 MHz clock frequency.

Hardware Performance. We compare the performance of the accelerator with the CPU-based and the GPU-based implementations. Here the evaluated CPU platform is the AMD EPYC 7601 32-Core Processor with 2.2 GHz clock frequency. The GPU platform is NVIDIA RTX A6000 with 1410 MHz clock frequency. Table 1 shows the average time cost to complete a motion planning task

Table 1: Average time cost (ms) to complete a planning task.

Operation Platform	Time Cost on Different Map Size		
	32 × 32	64 × 64	128 × 128
IMG-SMP Accelerator	5	14.4	48.3
AMD EPYC 7601 CPU	290	970	2090
NVIDIA RTX A6000 GPU	120	250	1020

under environments of different sizes. On 32 × 32, 64 × 64 and 128 × 128 maps, our accelerator achieves average 58×, 67.4× and 43.3× speedups over the CPU implementation, respectively. It also archives average 24×, 17.4× and 21× speedups as compared to the GPU implementation on these three maps, respectively. In addition, Fig. 9 shows the benefits of the proposed hardware accelerators with respect to area and energy efficiency. It can be seen that, on maps of size 32 × 32, 64 × 64, 128 × 128, our design achieves 2941×, 3418× and 2196× area efficiency improvement and 3589×, 2602× and 3140× energy efficiency improvement over the CPU based implementation, respectively. Moreover, compared to the GPU implementation, our accelerator achieves 4964×, 6279×, 4017× area efficiency improvement and 6757×, 5406×, 7544× energy efficiency improvement on maps of size 32×32, 64×64, 128×128, respecitively.

5 CONCLUSION

In this paper, we present IMG-SMP, a neural sampling-based motion planner. IMG-SMP can efficiently capture the spatial correlation between obstacles and robot state to guarantee high planning performance; and meanwhile the dataflow of IMG-SMP is specifically optimized to reduce computational cost. Experiment shows that our proposed neural planner can achieve very high planning and hardware performance simultaneously.

REFERENCES

[1] Steven M LaValle and James J Kuffner Jr. Randomized kinodynamic planning. *The international journal of robotics research*, 20(5):378–400, 2001.
[2] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894, 2011.
[3] Ahmed H Qureshi, Anthony Simeonov, Mayur J Bency, and Michael C Yip. Motion planning networks. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 2118–2124. IEEE, 2019.
[4] Masaya Inoue, Takahiro Yamashita, and Takeshi Nishida. Robot path planning by LSTM network under changing environment. In *Advances in computer communication and computational sciences*, pages 317–329. Springer, 2019.
[5] Alassane M Watt and Yusuke Yoshiyasu. Pathnet: Learning to generate trajectories avoiding obstacles. In *2020 IEEE International Conference on Image Processing (ICIP)*, pages 3194–3198. IEEE, 2020.
[6] Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *Icml*, 2011.
[7] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.