

DDPEN: Trajectory Optimisation With Sub Goal Generation Model

Aleksander Gamayunov^{1,2}

Aleksey Postnikov^{1,2}

Gonzalo Ferrer²

Abstract—Differential dynamic programming (DDP) is a widely used and powerful trajectory optimization technique, however, due to its internal structure, it is not exempt from local minima. In this paper, we present Differential Dynamic Programming with Escape Network (DDPEN) - a novel approach to avoid DDP local minima by utilising an additional term used in the optimization criteria pointing towards the direction where robot should move in order to escape local minima.

In order to produce the aforementioned directions, we propose to utilize a deep model that takes as an input the map of the environment in the form of a costmap together with the desired goal position. The Model produces possible future directions that will lead to the goal, avoiding local minima which is possible to run in real time conditions. The model is trained on a synthetic dataset and overall the system is evaluated at the Gazebo simulator.

In this work we show that our proposed method allows avoiding local minima of trajectory optimization algorithm and successfully execute a trajectory 278 m long with various convex and nonconvex obstacles.

I. INTRODUCTION

The need for service professions - couriers, cleaners, etc. is growing along with the population of the planet. At the same time, mobile robotics continue to develop and can replace at least part of the monotonous work, while providing people with new highly skilled jobs.

We are witnessing news about novel robot couriers, and robot vacuum cleaners that have already become commonplace while autonomous tractors are starting to drive through the agricultural fields.

Navigation is one of the core subsystems for robot's autonomy which is largely dependent on path planning and trajectory planning algorithms.

In this work, we consider optimizing algorithms and propose a solution to the important but common problem of nonconvex trajectory optimization.

Optimization trajectory algorithms iteratively modify the initial assumption of the trajectory according to the specified criteria, and gradually reduce the error value. One of the important issues is that the free state space is usually non-convex and presents multiple local minima. An optimization algorithm can reach one of the local minima and stay in it without ever reaching the global minima.

For a trajectory optimization task, reaching local minima can lead to a situation where the algorithm will not lead the mobile robot to the goal, even if the optimal trajectory exists. In that case, the mobile robot will not be able to complete

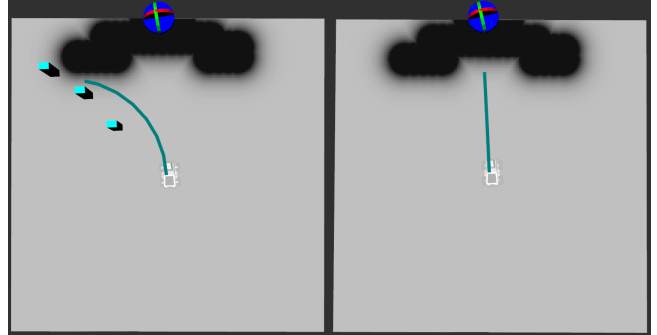


Fig. 1. Comparison of DDPEN (left side) with DDP (right side) on trajectory optimization with nonconvex obstacle. The robot is in the center of gray costmap. Black part of the costmap represents obstacles. Green line is the optimized trajectory. Green rectangles is predicted sub goals x_s . Blue circle is desired goal position x_{goal} .

its task.

A number of works are devoted to this problem. Zhaoming Xie et al. in his work of extension the differential dynamic programming trajectory optimization approach (DDP [1], [2]) with nonlinear constraints (CDDP) [3], mention that “like all nonconvex optimization, we need a good initial trajectory to ensure that the algorithm would not get stuck in a bad local minima”.

In that case sampling trajectory planning algorithm like RRT^* [4] or some sort of fast neural path planning A^* [5] algorithms for generating a safe initial trajectory for optimization can be used. Another approach of local minima avoidance is based on DDP maximum entropy formulation (MEDDP) [6], with multimodal policy exploration.

Escaping from local minima is a trivial task for human in case of 2d environment but the overall process of escaping from the local minimum is ill-posed with the current gradient-based optimisation tools in DDP.

In this work, we have proposed an approach for escaping from local minima that provides additional criteria of optimisation, calculated by a neural network, that “suggests” the best direction for the optimization process.

The Neural network learns to imitate optimal planning algorithm and provides intermediate goal as additional optimisation criteria that leads the optimisation process out from local minima.

We have evaluated vanilla DDP optimisation algorithm and our proposed approach (DDPEN) that takes additional optimization criteria from neural network at simulated scenarios in Gazebo [7] and have shown that our proposed approach is being able to complete complex trajectories even with nonconvex obstacles.

¹ The authors are with the Sberbank Robotics Laboratory, Moscow, Russia. {gamayunov.a.r, postnikov.a.l}@sberbank.ru.

²Skolkovo Institute of Science and Technology, Moscow, Russia. g.ferrer@skoltech.ru.

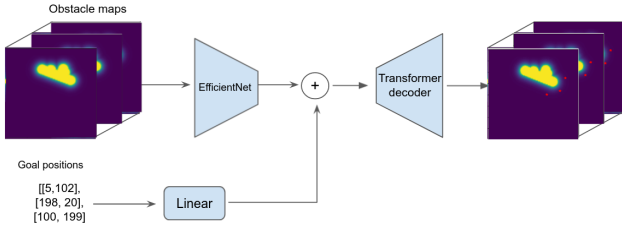


Fig. 2. Sub Goal Generation Model architecture.

II. METHOD

A. Trajectory optimisation problem

The state of an agent at time t is represented by x_t , where x is the two dimensional vector in the Cartesian coordinate system. Transitions between states are provided by the dynamic function $x_{t+1} = f(x_t, u_t)$ where u_t is a control action at time t . Sequence of control action $U = [u_{t_0}, u_{t_1}, \dots, u_{t_{N-1}}]$ applied to dynamic function forms the trajectory $X = [x_{t_0}, x_{t_1}, \dots, x_{t_N}]$ from initial time t_0 to prediction horizon t_N . The total cost of trajectory J_0 is the sum of running costs l and the final cost l_f . The solution of the optimal control problem is to find the minimum cost control sequence.

$$J_0(X, U) = \sum_{i=0}^{N-1} l(x_i, u_i) + l_f(x_N) \quad (1)$$

$$U^*(X) \equiv \arg \min_U J_0(X, U) \quad (2)$$

The current work focuses on the cost function $l(x, u)$ that forms the optimisation direction and can move the optimisation process into a local minimum.

The cost function, in our vanilla realisation of DDP optimisation method, includes c_g - cost of distance to the goal state x_{goal} , c_c - control cost and c_o - cost representing the distance to obstacles obs at distance less than d meters.

$$l(x, u) = c_c(u) + c_g(x, x_{goal}) + c_o(x, obs) \quad (3)$$

$$l_f(x) = c_g(x, x_{goal}) + c_o(x, obs) \quad (4)$$

$$c_c(u) = \sum u^2 \quad (5)$$

$$c_g(x, x_{goal}) = \|x - x_{goal}\|_2 \quad (6)$$

$$c_o(x, obs, d) = \sum ReLU(d - \|x - obs\|_2) \quad (7)$$

B. DDPEN

The proposed method DDPEN takes into account an additional cost c_s - distance to sub goal x_s , generated by the neural network Eq 10, which represents the movement direction of the mobile robot that allows avoiding local minima.

$$l(x, u) = c_c(u) + c_g(x, x_{goal}) + c_o(x, obs) + c_s(x, x_s) \quad (8)$$

$$c_s(x, x_s) = \|x - x_s\|_2 \quad (9)$$

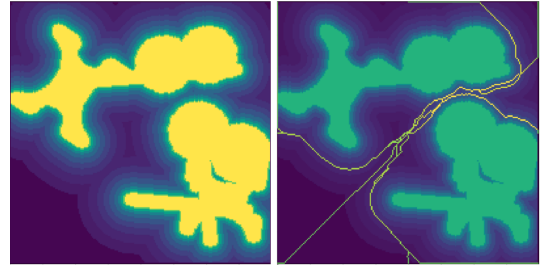


Fig. 3. Synthetically generated samples from dataset. *Left*: Example of randomly generated costmap *Right*: Example of paths obtained by A* algorithm to randomly sampled goals.

C. Sub Goal Generation Model

In order to generate future positions that will force the trajectory optimization algorithm to avoid local minima we propose to utilise deep model Eq. 10, that is trained to imitate goal proposals generated by A* algorithm, that would satisfy real-time restrictions and whose performance would not depend on the position of obstacles.

$$x_s = SG_\theta(map, x_{goal}), \quad (10)$$

where SG_θ - deep Sub Goal Generation Model, map - local costmap of environment, x_{goal} - desired goal position of robot.

Architecture of Sub Goal Generation Model, shown in Figure 2, consist of convolution based encoder of obstacle map, namely EfficientNet B0 [8], linear layer that embeds goal positions, and transformer based decoder that generates sub goal positions.

D. Dataset

We have created a synthetic dataset that captures randomly generated obstacle maps and optimal path to randomly sampled goal on that map. As a base for obstacle map generation it is taken a costmap with 200x200 cells with the cell side length of 5cm. Predefined obstacles are created at random positions with non-blocking center position constraint. Obstacles are positioned through a random affine transformation for augmentation purposes. After generation of obstacles, their costs are dilated from occupied cost to free moving space.

We have used A* path planning algorithm to generate examples of paths from starting position, which is in our case always the grid center, to randomly sampled goal position near borders of the costmap. After processing the optimal path from randomly sampled goal positions on a randomly generated costmap, intermediate positions of sub goals at 30, 50, 70 steps of the optimal path are saved together with goals and costmaps, forming the overall dataset. Synthetically generated samples from the dataset is shown in Figure 3.

E. Simulator

For conducting experiments and evaluation, we use Gazebo [8] simulator with an environment map reconstructed

TABLE I
COMPARISON OF DDP AND DDPEN TIME EXECUTION OF PATH. THE VALUES IS THE MEAN $\pm\sigma$ IN SECONDS

Type of obstacles	DDP		DDPEN	
	Forward pass	Backward pass	Forward pass	Backward pass
Free	191 \pm 1	188 \pm 1	191 \pm 1	198 \pm 1
Cylinders	243 \pm 1	313 \pm 27	255 \pm 8	281 \pm 32
Cubes without minima	284 \pm 54	262 \pm 31	243 \pm 15	266 \pm 6
Cubes with minima	fail	fail	245 \pm 7	275 \pm 20

from lidar scans and four wheels sidewalk robot model with a lidar sensor. Environment map and model of robot is shown in Figure 4. Point cloud data from simulated lidar sensor after processing by ROS2 [9] node provides costmaps that uses as an input to trajectory optimization algorithm and sub goal generation model. We use a global path passing through 8 points on the reconstructed map for experiments, as shown in Figure. 4. x_{goal} is located on the intersection of the local costmap border and the line between next and previous global path points, and is used as input to the trajectory optimization algorithm and sub goal generation model.

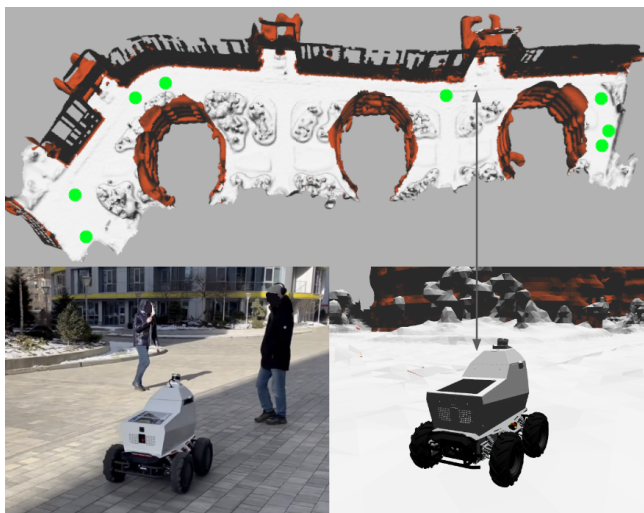


Fig. 4. Simulated street on the top on figure, simulated sidewalk robot in environment on the right down and the real robot at the same place on the left down. Green circles on the simulated street show the path points for experiments.

F. Running cycle

The implementation of a particular optimization algorithm takes 75ms for the optimization epoch, where generation of intermediate goal by neural network takes $9.03 \text{ ms} \pm 139 \mu\text{s}$. We use 4 optimisation epochs before executing calculated trajectory. For the next cycle of 4 optimisations we use previously generated trajectory with cut past trajectory predictions and added new assumption of trajectory by applying last control, as shown in Figure 6. Executor uses generated trajectory for sending controls to the robot at stable 10Hz and approximate controls between predictions while next optimisation cycle will end.

III. EVALUATION

We evaluate vanilla DDP and proposed DDPEN trajectory optimization algorithms by executing path of 278 meters in both directions 5 times. The path that the robot must follow passes through the green dots as shown in the Fig. 4.

We manually place different types of obstacles along the path which results in 4 different environments:

- *Free* - original obstacle free environment.
- *Cylinders* - environment with additionally placed cylinders with the radius 1m along the path.
- *Cubes without minima* - environment with additionally placed cubes with a side 2m forming convex obstacles.
- *Cubes with minima* - environment with additionally placed cubes with a side 2m forming nonconvex obstacles.

Resulting trajectories of DDP and DDPEN in an environment with *Cubes with minima* is shown in Figure 5.

Both DDP and DDPEN have equal configuration and dynamic model with linear max velocity limit of 1.5 m/s and angular velocity limit of 0.5 rad/s.

The main goal that we have focused on in this work is the ability to complete the entire path trajectory even with non-convex obstacles, and we also evaluated the path execution time as an indicator of system efficiency.

A. Results

Table I shows the average time \pm standard deviation in seconds for passing 4 different scenarios, where longer execution times mean more iterations were required to converge, failures happens due to weak gradient signals in configurations that are close a local minima. Table I shows expected result for DDP which can't complete trajectory on map with local minima, while our proposed method DDPEN is being able to complete trajectory with nonconvex obstacles.

Time of path execution in forward and backward directions are not equal, because trajectories in different directions have minor differences that affect the optimization process.

We can see time execution deviation in the same scenarios that shows how simulation errors affects trajectory optimisation process. A small error in the the simulation process leads to differences in the execution time of the trajectory, especially in the case of trajectories located near obstacles.

The sum of trajectory execution mean time have differences on cylinders and cubes maps - DDP $1102 \pm 68\text{s}$ comparing to DDPEN $1045 \pm 37\text{s}$, because DDPEN starts

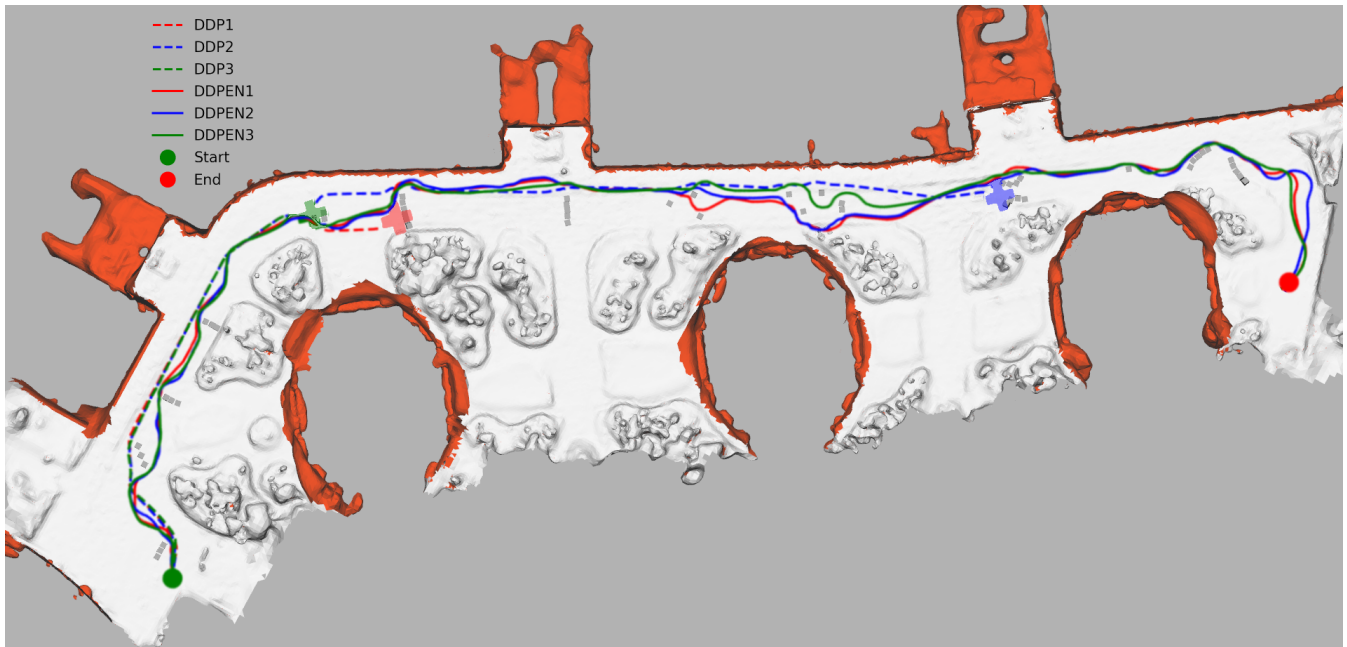


Fig. 5. Robot trajectories in a simulated environment with non-convex obstacles. The green circle is the beginning of the paths, the red circle is the end of the paths. The solid and dotted lines are paths created by DDPEN and DDP respectively. The cross marker is the end of the paths in case of hitting the local minimum.

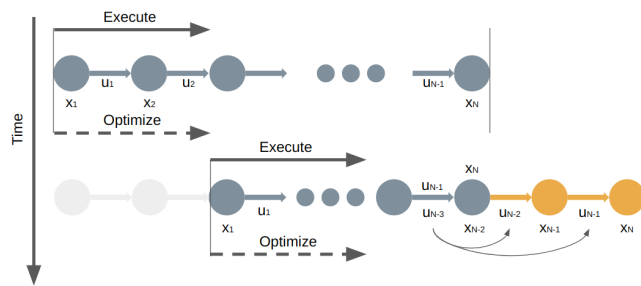


Fig. 6. Running cycle - execution trajectory while optimisation process in progress

avoiding obstacles based on additional sub goal provided by Sub goal generation model earlier than obstacle cost c_o starts to affect the vanilla DDP optimization process as shown in Figure 1.

IV. CONCLUSION

In this paper we have shown a novel approach DDPEN that allows trajectory optimization algorithms to escape from local minima. Our approach utilizes an additional optimization term representing the direction towards the robot should move in order to escape local minima. This additional optimization term based at deep sub goal generation model that predicts future directions that the robot should follow in order to avoid local minima. The model is trained with synthetic dataset and overall system is evaluated on a simulated scene at Gazebo simulator.

The evaluation shows that the proposed method DDPEN is more reliable against the local minima problem, while vanilla DDP fails.

REFERENCES

- [1] D. Mayne, "A second-order gradient method for determining optimal trajectories of non-linear discrete-time systems," *International Journal of Control*, vol. 3, no. 1, pp. 85–95, 1966.
- [2] D. H. Jacobson and D. Q. Mayne, *Differential dynamic programming*. Elsevier Publishing Company, 1970, no. 24.
- [3] Z. Xie, C. K. Liu, and K. Hauser, "Differential dynamic programming with nonlinear constraints," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 695–702.
- [4] I.-B. Jeong, S.-J. Lee, and J.-H. Kim, "Rrt*-quick: A motion planning algorithm with faster convergence rate," in *Robot Intelligence Technology and Applications 3*. Springer, 2015, pp. 67–76.
- [5] R. Yonetani, T. Taniyai, M. Berekatain, M. Nishimura, and A. Kanazaki, "Path planning using neural a* search," in *International conference on machine learning*. PMLR, 2021, pp. 12029–12039.
- [6] O. So, Z. Wang, and E. A. Theodorou, "Maximum entropy differential dynamic programming," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 3422–3428.
- [7] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 3. IEEE, 2004, pp. 2149–2154.
- [8] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*. PMLR, 2019, pp. 6105–6114.
- [9] S. Macenski, F. Martín, R. White, and J. G. Clavero, "The marathon 2: A navigation system," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 2718–2725.