RELAX: Reinforcement Learning Enabled 2D-LiDAR based Autonomous System for Parsimonious UAVs

Guanlin Wu[1], **Zhuokai Zhao**[2], **Huan Chen**[3], **Jinyi Zhao**[4], **Yangke Zhang**[5], **Yutao He**[6]
[1]The Chinese University of Hong Kong, Shenzhen, Shenzhen, China
[2]Department of Computer Science, University of Chicago, Chicago, IL, USA
[3]Department of Mathematics, Hong Kong University of Science and Technology, Hong Kong
[4]Eleanor Roosevelt college, University of California, San Diego, CA, USA
[5]Rady School of Management, University of California, San Diego, CA, USA
[6]Computer Science Department, UCLA, Los Angeles, CA, USA
guanlinwu@link.cuhk.edu.cn

Abstract

Unmanned Aerial Vehicles (UAVs) have become increasingly prominence in recent years, finding applications in surveillance, package delivery, among many others. Despite considerable efforts in developing algorithms that enable UAVs to navigate through complex unknown environments autonomously, they often require expensive hardware and sensors, such as RGB-D cameras and 3D-LiDAR, leading to a persistent trade-off between performance and cost. To this end, we propose RELAX, a novel end-to-end autonomous framework that is exceptionally cost-efficient, requiring only a single 2D-LiDAR to enable UAVs operating in unknown environments. Specifically, RELAX comprises three components: a pre-processing *map constructor*; an offline *mission planner*; and a reinforcement learning (RL)-based *online replanner*. Simulation experiments demonstrate that RELAX offers more robust dynamic navigation compared to existing algorithms, while only costing a fraction of the others. The code will be made public upon acceptance.

Unmanned Aerial Vehicles (UAVs), commonly known as drones, have gained immense importance and become a transformative technology across many application domains (Rovira-Sugranes et al. 2022). In addition to more commonly-known use cases such as military navigation (Patil et al. 2020), search-and-rescue (Mishra et al. 2020), and commercial package delivery (Alvarado 2021), UAVs are also widely used in metrology (Wieczorowski et al. 2021), agriculture (Ahirwar et al. 2019), and mining (Shahmoradi et al. 2020) thanks to their compact sizes and relatively high cost-efficiency, especially when compared to the piloted aircraft.

Despite tasks from different applications pose different, often specific challenges, one of the key challenges shared across all domains is being able to operate autonomously. Specifically, it covers many aspects of the UAV operations, including environment perception, path planning and real-time dynamic obstacle avoidance. It is especially important when UAVs are to operate under unknown or dynamically changing environments, where human control is unavailable.

UAV autonomous navigation algorithms require on-board sensors to understand the surrounding environments, and optimally navigates UAV to travel from one place to another (Barnhart, Marshall, and Shappee 2021). Optimal navigation can be defined in terms of the length of the traveled path (Noreen, Khan, and Habib 2016), traveled time (Kularatne, Bhattacharya, and Hsieh 2016) and trajectory smoothness (Xu, Song, and Cao 2021) while being collision-free (Shin and Chae 2020). Numerous efforts have been devoted to advance this field (Jones, Djahel, and Welsh 2023). However, existing solutions often require UAVs to equip with expensive sensor setups, including multiple RGB-D cameras (Cheng and Chen 2021; Xu et al. 2023; Cui, Chen, and Li 2022; Kim et al. 2022) or 3D LiDAR (Qin et al. 2019). While these sensors can help build real-time 3D maps for better environment representations, they significantly increase the cost of the autonomous system, as most RGB-D cameras are priced at a few hundreds dollars in average (Ulrich et al. 2020), and 3D LiDAR often costs well above a thousand US dollars (Van Nam and Gon-Woo 2021). Consequently, such high cost has become a primary factor preventing UAVs from wider adoptions (Aggarwal and Kumar 2020). Therefore, new solutions enabling UAVs to perform *successful and reliable autonomous navigation while using much simpler and cheaper sensor setups* are urgently needed.

Simpler sensor configuration, which changes the system perception of the surrounding world, poses many new challenges for all system components including surrounding environment construction, path planning, dynamic obstacle avoidance, and often calls for an entire new system design. Specifically, it introduces practical challenges in surrounding detection (Ocando et al. 2017; Murcia, Monroy, and Mora 2018) and RL training (Ding and Dong 2020). To this end, we introduce **R**einforcement Learning **E**nabled 2D-**L**iDAR **A**utonomous **S**ystem (**RELAX**), an end-to-end autonomous system presenting novel algorithms addressing these intricacies, so that parsimonious UAVs that carry only one 2D-LiDAR sensor can navigate autonomously in unknown environments. Specifically, RELAX comprises three components: a *map constructor*, which generates occupancy maps using 2D-LiDAR data; a *mission planner*, which cre-

ates obstacle-free paths using these maps; and an *online replanner*, which addresses the dynamic obstacle avoidance.

The main contribution of this paper is that we propose RELAX, *the first UAV autonomous navigation system that requires only a single 2D-LiDAR to support the entire UAV autonomous navigation pipeline*, which includes the initial environment mapping, offline planning and online replanning for dynamic obstacle avoidance. To address the unique challenges that come with the less feature-rich sensor inputs, we propose novel algorithms to enhance the capability and generalizability of our framework. Experiments shows that RELAX achieves comparable successful rates as more expensive UAVs navigation systems, at only a fraction of the cost. In addition, we advocate RELAX as a *successful proof-of-concept and a platform that boosts future research* by releasing a real-time training suite in ROS-Gazebo-PX4 simulator, which supports easy adaptation of RELAX algorithms into newly designed RL algorithms in the future. In other words, the idea of modularization behind the design of RELAX brings larger potential for further improvement of its performance.

Related Work

Existing end-to-end UAV autonomous navigation systems leverage sensor (e.g. RGB-D, 3D-LiDAR) inputs to perceive and understand surrounding environment, then conduct path planning and automatic dynamic obstacle avoidance (Elmokadem and Savkin 2021). Besides differences in the algorithmic aspects, sensor configurations also fundamentally affect the overall design of the system architecture, as well as the specific algorithms within each component. In this section, we briefly discuss different UAV navigation systems that equip with different sensor configurations.

Vision-based UAV navigation systems. Vision-based systems that employ RGB or RGB-D images to capture the environment are arguably the most prevalent configuration in autonomous UAVs (Lu et al. 2018). More specifically, RGB images are taken by monocular cameras, while RGB-D images refer to 3D representations of the world that is captured by either binocular cameras or monocular camera with additional depth sensor.

Numerous efforts have been devoted to vision-based UAV systems. For example, Engel et al. (Engel, Sturm, and Cremers 2014) developed a quadrator carrying a monocular camera that is capable of visual navigation in unstructured environments. Although being low in cost, the proposed system does not support obstacle avoidance, which is a major disadvantage for many modern tasks. As a result, many works choose to use binocular cameras (Mao et al. 2019; Jingjing, De, and Fei 2019). However, such systems are very prune to weather changes and are hard to operate at night, greatly limiting their working scenarios.

Because of the aforementioned disadvantages of monocular and binocular configurations, RGB-D which involves both RGB and infrared depth cameras quickly attracts many attentions, resulting in various UAV applications (Bachrach et al. 2012; Xu et al. 2023). While being effective, the use of RGB-D cameras inevitably increases both cost and on-board computational requirement, posing limitations and preventing designs of simple, low-cost and light-weight UAVs for wider adoptions.

LiDAR-based UAV navigation systems. Thanks to its robust performance under various weather and lighting conditions, LiDAR has quickly become the mainstream sensor in many modern UAV navigation systems (Jeong, Hwang, and Matson 2018; Qin et al. 2019). LiDAR sensors can be divided into two categories: single-line and multi-line, where single-line scans one plane of the obstacles to obtain a 2D map, while multi-line scans multiple surfaces to obtain a 3D point cloud of the environment. Based on the output types, single- and multi-line LiDAR are also called 2D and 3D LiDAR.

Attracted by the richer environment representations that 3D LiDAR produces, most existing UAV autonomous systems utilize 3D LiDAR as the sensor configurations (Qin et al. 2019; Aldao, González-de Santos, and González-Jorge 2022; Liang et al. 2023). However, despite existing work's favor into 3D LiDAR, the rich 3D environmental representations might not be all necessary to perform UAV path planning and robust obstacle avoidance, leaving room for better cost-effective designs. In other words, configurations that utilize 2D LiDAR, where we call a *parsimonious configuration*, may achieve a more balanced trade-off between performance and cost. For example, Gabriel et al. (Gabriel et al. 2023) leverage 2D LiDAR and propose an adaptive path-planning solution that combines Rapidly Exploring Random Trees (RRT) and deep RL for the autonomous trajectory generation of UAVs in agricultural environments. However, it does not depend on UAV-scanned data at all stages but rather leverages a comprehensive Python environment for its operations, failing to equip the system with efficient obstacle avoidance capabilities. Contrary to this, RELAX prioritizes enhancing obstacle avoidance by mainly utilizing LiDAR data. More specifically, we employ the ROS-Gazebo-PX4 simulator for developmental purposes, incorporating a variety of algorithms aimed at overcoming different obstacles and ensuring the training's applicability in a real-time simulation setting.

Methodology

RELAX is designed specifically for parsimonious UAVs, which are drones that lack odometers, RGB-D cameras, 3D-LiDAR, or gimbals systems, and only equip simple sensors, such as 2D-LiDAR and inertial measurement unit (IMU). More specifically, RELAX utilizes RPLiDAR[1], a low-cost 2D laser scanner that performs 360-degree scan within a certain range to produce 2D point clouds of the surrounding.

RELAX consists of five modules, as shown in Fig. 1. *Resources* module contains necessary sensor outputs including point clouds captured by 2D-LiDAR, velocity and pose of UAV obtained from IMU, and the map generated by a *map constructor*. Specifically, *map constructor* synthesizes an occupancy map of the environment using point clouds from 2D-LiDAR. *Mission planner* provides an obstacle-free path

[1]More details at https://www.slamtec.ai/product/slamtec-rplidar-a1/.

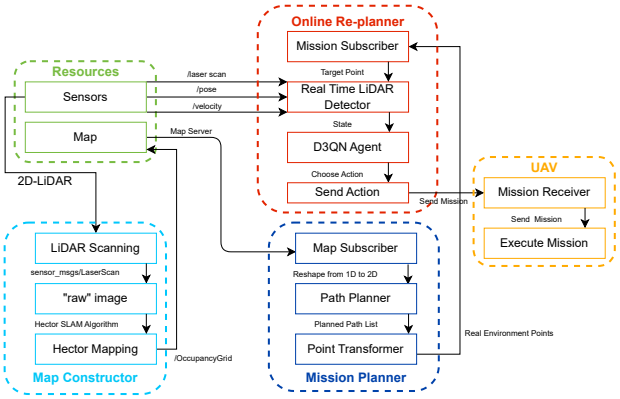

Figure 1: System overview: RELAX starts from checking whether the occupancy grid map exists. If there is no map, it will run *map constructor* to enter the map constructing mode. While we manually operate the drone to fly one complete circuit around the environment at a specific altitude, *map constructor* processes the data from 2D-LiDAR and integrates these data to create an occupancy grid map. This map is then sent back to *resources* and available to other modules. Next, *mission planner* subscribes this map and uses it to plan an obstacle-free path from start to target and sends to *online re-planner* for dynamic obstacle avoidance using real-time 2D-LiDAR inputs.

from the starting point to the target position based on the occupancy map. And *online re-planner* navigates the drone (illustrated as the *UAV* module) to move along this planned path and perform online re-planning to avoid dynamic obstacles. The "dynamic obstacles" in this paper refers to the static obstacles that are not included in the map produced by *map constructor*.

Following (Gabriel et al. 2023), we separate static path planning from online path re-planning, with the underlying intuition that the environment does not undergo significant changes in a short period. And separation in the different path planning stages significantly reduces the time cost. We illustrate *map constructor*, *mission planner* and *online re-planner* with details in the following sections.

Map Constructor

Map constructor leverages 2D-LiDAR in tandem with Hector-SLAM (Kohlbrecher et al. 2011) to construct a grid occupancy map of the environment.

LiDAR scanning. LiDAR scanning module generates raw images of the surrounding environment at a particular UAV position, as shown in the left of Fig. 2. In 2D-LiDAR system, the scanned images adopts a structure that aligns with x- and y-axis after a reshape operation performed on the one-dimensional LiDAR data array. Then the image is processed into an occupancy grid map, as shown in the right of Fig. 2, where the level of confidence regarding obstacle existence is represented through dark (low) to light (high). While flying through the environment, the drone constantly generates "raw" images, contributing to the ongoing construction of the environment.

Hector-SLAM. Map constructor employs Hector-SLAM (Kohlbrecher et al. 2011) to integrate all the LiDAR-scanned "raw" images into a single map that represents the entire environment. More specifically, Hector-SLAM operates across three primary phases, which are *map access*, *scan matching* and *multi-resolution map representation*. In *map access*, the initial occupancy grid map takes shape, driving from the first "raw" image. Next, *scan matching* matches the "raw" image taken at time t to the previous occupancy grid map from t 1 through points correspondence. To lower the risk of getting stuck in local optimal solution, Hector-SLAM applies *multi-resolution map representation* to simultaneously keep different maps and update them based on pose estimations. The resulting map in our case is shown in the left of Fig. 3.

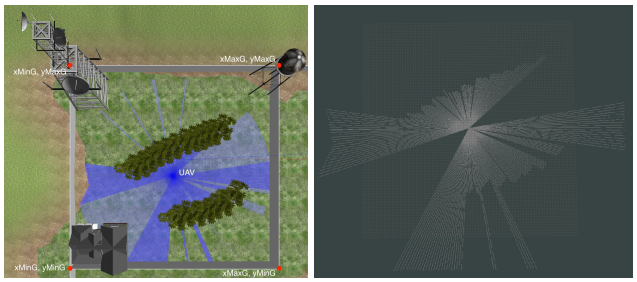

Figure 2: Left: environment of UAV at a particular position; Right: "raw" 2D-LiDAR scanning image of the left environment.

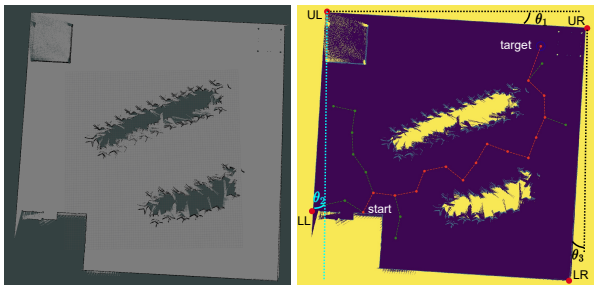

Figure 3: Left: occupancy map constructed by Hector-SLAM. Right: a path planning result based on the given occupancy map.

Mission Planner

Mission planner receives the occupancy map from map constructor, and plans an obstacle-free path from start to end. It includes two components, which are *path planner* and *point transformer*. The complete algorithm of mission planner is detailedly illustrated in Algorithm 1.

Path planner. Path planner is responsible for generating a collision-free path from start to end based on the static occupancy map. In our case, this map refers to the output of map constructor. Since the generation of such path does not depend on characteristics unique to parsimonious UAVs, any standard path planning algorithm should suffice. For wider adaptability, lower execution time, and relatively optimal path, we use Rapidly Exploring Random Tree (RRT) (LaValle and James J. Kuffner 2001) in this paper to

Algorithm 1: Mission Planner

Input: start, target, number of iterations, grid, step size, test range

Output: path between start and target in real environment

Initialization :

N_{start} treeNode(start); N_{target} treeNode(end)

R_{tree} RRTAlgorithm(N_{start}, N_{end}, numOfIterations, grid, stepSize, testRange)

upperLeftPoint, upperRightPoint, lowerRightPoint, lowerLeftPoint Scanned Occupancy Grid Map; xMinG, xMaxG, yMinG, yMaxG Gazebo World Environment

LOOP Process

for i = 0 to $numOfIterations$ **do**

  R_{tree}.resetNearestValues(); point R_{tree}.sampleAPoint()

  $N_{nearest}$ R_{tree}.findNearestPoint()

  N_{new} R_{tree}.steerToPoint($N_{nearest}$)

  flag check if there are obstacles between N_{new} and $N_{nearest}$

  if not reach target **then**

    Add N_{new} to R_{tree} and check whether N_{new} in the test range of target, if yes, then break

  end if

end for

R_{tree}.WayPoints R_{tree}.retraceRRTPath()

WaypointsTransformed Eq.1, Eq.2

return WaypointsTransformed

showcase the feasibility of our proposed framework. An example path is shown in the right of Fig. 3.

Point transformer. To navigate UAV through real-life environment, a transformation is needed to convert the path from path planner into real-life coordinates. To begin with, we initiate a rotation of the map, as shown in the right of Fig. 3. The rotation angle emerges from the cumulative summation of three lines' shifting angles (θ_1, θ_2, θ_3 in Fig. 3 right), where each bears a weight that minimizes potential errors. After rotation, every intermediate point along the trajectory undergoes calculation based on the ratio between distances in occupancy grid map and their counterparts in real environment. Four example points, `UL`, `LL`, `UR`, and `LR` are illustrated in Fig. 3, where their corresponding points in real-life environment are the `xMinG`, `xMaxG`, `yMinG` and `yMaxG` in Fig. 2.

Let θ denotes the weighted sum of individual-axis rotation angles, r denotes the distance between origin and point p, and (x_{pr}, y_{pr}) be the x and y of p after rotation, we have the real-life environment coordinates $(x_p^{\text{new}}, y_p^{\text{new}})$:

$$x_p^{\text{new}} = (r \cos(\theta) + x_{pr}) \frac{\texttt{xMaxG} - \texttt{xMinG}}{\sqrt{(x_{ur} - x_{ul})^2 + (y_{ur} - y_{ul})^2}} \tag{1}$$

$$y_p^{\text{new}} = (r \sin(\theta) + y_{pr}) \frac{\texttt{yMaxG} - \texttt{yMinG}}{\sqrt{(x_{ur} - x_{lr})^2 + (y_{ur} - y_{lr})^2}}. \tag{2}$$

Online Re-planner

As the multitude of dynamic obstacle scenarios makes it impractical to establish comprehensive avoidance rules, a learning-based planning algorithm is designed to perform autonomous obstacle avoidance in dynamic, unknown environment. More specifically, we propose a novel RL-based online re-planner combining Double Deep Q-networks (DDQN) (Van Hasselt, Guez, and Silver 2016) and dueling architecture (Wang et al. 2016).

Network structure. Dueling Double Deep Q-networks (D3QN) improved upon Deep Q-networks (DQN) (Mnih et al. 2013) and Double Deep Q-networks (DDQN) (Van Hasselt, Guez, and Silver 2016) by incorporating the dueling architecture (Wang et al. 2016). More specifically, it splits the Q-value estimations into two separate functions, namely a value function, $V(s)$, estimating the reward collected from state s; and an advantage function, $A(s, a)$, estimating if action a is better than other actions at state s. Both value and advantage functions are constructed with a set of dense layers and are later combined to output Q-value for each action, with the combination operator shown in Eq. (3).

$$Q(s, a) = V(s) + \left(A(s, a) \quad \frac{1}{|A|} \sum_{a'} A(s, a') \right) \tag{3}$$

State design. We integrate the orientation vector spanning from the current location to the target, along with real-time LiDAR data, into our state design. Specifically, real-time LiDAR data, which are 360-vectors representing each degree, is partitioned into 8 sectors through thresholded min-pooling, where each corresponds to a specific direction, such as "forward-left" or "forward-right", as shown in the left of Fig. 4. More precisely, we have:

$$d_i = \min\left(all\_dist \in region_i, det\_range\right), i \in [0, 7] \tag{4}$$

where det_range denotes the threshold value and all_dist represents the distances of all points (in our case is $\frac{360}{8} = 45$) in $region_i$. Let (x_c, y_c, z_c) and (x_t, y_t, z_t) denote the current and target position, we have the direction vector defined as:

$$(x_d, y_d, z_d) = (x_t, y_t, z_t) \quad (x_c, y_c, z_c) \tag{5}$$

Finally, the current state is defined as:

$$state = [x_d, y_d, z_d, dist_0, dist_1, \ldots, dist_6, dist_7] \tag{6}$$

One of the biggest challenges using 2D-LiDAR is that the data may be extremely noisy due to the disturbances from UAV maneuvers. To address this challenge, we propose a novel data filtering mechanism, as illustrated in Algorithm 2 to enhance the accuracy of the acquired data. The core idea for judging whether this data is noisy is that within all potential actions, the greatest conceivable variation in distance between two states should be $\sqrt{2} < 1.5$. The rule of $\sqrt{2}$ comes from our definition of action, which will be illustrated with more details later in this section. Let x_i, y_i denote the coordinate difference between step i 1 and step i in x and y-axis, respectively, we have $\sqrt{\max|x_i| + \max|y_i|}$ $\sqrt{2}$.

On the other hand, if the disparity is larger than 1.5, it is deemed to be spurious noisy and is more carefully handled as shown in Algorithm 2.

Algorithm 2: Real-time 2D-LiDAR Data Filtering

Input: LiDAR data from last episode (lidar_data_t) and this episode (lidar_data), a list used for recording function (index_list)
Output: state

```
for i = 0 to len(lidar_data)-1 do
  if lidar_data_t[i] - lidar_data[i]  1.5 then
    index_list[i] + = 1
    d_r = floor(0.5  det_range)
    if index_list[i]  d_r then
      lidar_data[i] = det_range  d_r + 1
      index_list[i]  = (det_range  d_r  1)
    else
      lidar_data[i] = lidar_data_t[i]  1
    end if
  else
    if index_list[i] > 0 then
      index_list[i]  = 1
    end if
  end if
end for
state = lidar_data
return state
```

The heuristic behind Algorithm 2 is that there is a very small likelihood of continuously obtaining noisy data more than $det\_range/2$ number of times within the same region. Thus, we maintain a $index\_list$ to record the number of times that noisy data occurred for each region and will dynamically decrease for getting data without noisy. In addition, after getting noisy data, we will set the distance to $(det\_range/2)$ + 2 or subtract 1 from it depending on the corresponding number at $index\_list$.

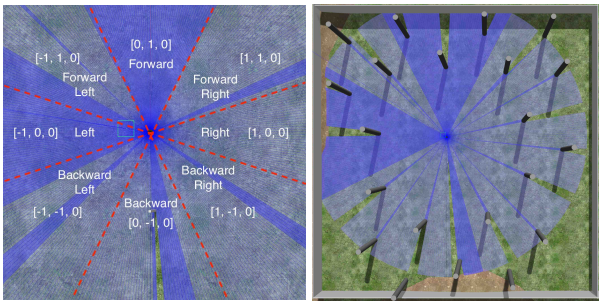

Figure 4: Left: state-action correspondence, where agent can choose or exclude action [1; 1; 0] based on the distance. Right: training environment of the model.

Action space. As we constrain UAV from moving vertically due to sensor limitations (RPLiDAR can only scan horizontally), the action space A, as illustrated in Eq. (7) and Fig. 4 left, only contains 8 actions. Specifically, we have

$$A = f[1; 1; 0]; [1; 0; 0]; [1; 1; 0]; [0; 1; 0]; [1; 1; 0]; [1; 0; 0]; [1; 1; 0]; [0; 1; 0]g \quad (7)$$

Reward function design. Temporal reward r_t that trains the model to learn optimal actions reaching the target point is illustrated in Eq. (8). More specifically, let $d_{current}$ denotes the distance between current and target position, and d_{last} denotes the distance between the previous and target position, we have:

$$r_t = \begin{cases} 3000; & d_{current} \quad 3 \\ 3000; & num\_steps\_taken \quad max\_num\_steps \\ 50; & d_{current} > d_{last} \\ 4000; & collision \end{cases} \quad (8)$$

And the final reward r for each chosen action is simply:

$$r = r_t \quad \frac{d^2_{current}}{100} \quad (9)$$

The underlying rationale for the configuration of the reward mechanism is to prioritize the drone's learning process in avoiding collisions as the paramount task, while also discouraging the behavior of continuously circling a point proximate to the target.

Check done. Ensuring timely notifications about the completion of an episode holds immense significance in RL training. Since we incorporate real-time LiDAR data into state representation, the training must proceed in Gazebo-ROS-PX4 simulator, which introduces complexity to the reset process after collision events. Upon drone's collision with an obstacle, automatic disarming occurs, and manual restarts of ROS, Gazebo, and PX4 are required for rearming, rendering continuous training infeasible. To expedite model convergence and streamline the intricate reset procedure, we devise the check done function as shown in Eq. (10), which defines the completion of an episode when the distance between the UAV and the obstacle is less than a predefined collision threshold $col\_threshold$.

$$done = \begin{cases} True; & d_{current} \quad 3 \\ True; & jx_cj > jlimit\_xj or jy_cj > jlimit\_yj \\ True; & counter \quad step\_threshold \\ True; & 9i \ 2 \ [0; 7] \ s.t. \ dist_i \quad col\_threshold \\ False; & else \end{cases} \quad (10)$$

Reset. Reset operation is the guarantee of a well-trained model. During training, at the end of each episode, the drone will be set to $(0, 0, 4.4)$ in Gazebo world. However, according to the settings of ROS, the position of drone will not immediately be set to $(0, 0, 4.4)$. It will either be directly set to $(0, 0, 4.4)$ after some time, which depends on the distance between the previous position of drone and $(0, 0, 4.4)$, or decreasing from the previous position to $(0, 0, 4.4)$ step by step. For example, if the drone was at position $(5, 6, 4.4)$, after setting to $(0, 0, 4.4)$ in Gazebo world, the position of drone in ROS topic might change as $(5, 6, 4.4)$! $(4, 6, 4.4)$! $(3, 6, 4.4)$! ... ! $(0, 0, 4.4)$.

Algorithm 3: Reset

Input: a_{thr}, b_{thr}, $offset_a$, $offset_b$
Output: None

while $True$ **do**
  if distance between (x_c, y_c, z_c) and $(0, 0, 4.4)$ 1 **then**
    break
  else
    if $(x_t$ 6 x_c **or** y_t 6 $y_c)$ **then**
      if jx_tj a_{thr} **then**
        if jx_tj b_{thr} **then**
          x_t jx_tj $offset_b$
        else
          x_t jx_tj $offset_a$
        end if
      else
        x_t x_c
      end if
      if jy_tj a_{thr} **then**
        if jy_tj b_{thr} **then**
          y_t jy_tj $offset_b$
        else
          y_t jy_tj $offset_a$
        end if
      else
        y_t y_c
      end if
      start $[x_t, y_t, 4.4]$
      Let drone move to start
    end if
  end if
end while

This setting raises a fatal problem: before the position of drone in ROS becomes $(0, 0, 4.4)$, the drone will move uncontrollably and has high risk of colliding with obstacles during this period. To solve this problem, we propose a novel reset algorithm as shown in Algorithm 3, in which (x_c, y_c, z_c) denotes the current drone position in ROS topic. The core idea of the Algorithm 3 is to control the movement of drone into a specific range of space, among which we can guarantee no collision will happen. The parameters such as a_{thr}, b_{thr}, $offset_a$, $offset_b$ are manually set to serve this purpose and should be modified when training environment is different.

Training details. To ensure the model learns a policy that is independent of absolute positions (specified as x, y, z), the target position for each episode will be generated randomly within a predetermined range for x and y, while maintaining z at a constant value. The complete training procedure, which summarizes the core algorithm of our proposed RELAX, is illustrated in Algorithm 4. Detailed hyperparameter values are shown in Table 2.

To emphasize the novelty of our proposed 2D-LiDAR UAV system RELAX, we summarize the differences between the most updated existing frameworks, to our best knowledge, and ours in Table 1. We see that RELAX presents several advantages than the others. Firstly, the mapping function, as we illustrated in §, enables the use of our system in a wide range of environments without the need of manually reconstructing the environments. Secondly, the ability to conduct real-time training in Gazebo-ROS-PX4 simulator ensures the learning of dynamic obstacle avoidance. More specifically, the dynamic obstacle avoidance utilizing real-time LiDAR data, as illustrated in §, improves significantly when compared to existing system (Gabriel et al. 2023) that utilizes only (x, y, z) positions.

Framework	Mapping	PP	Alg	DOA	TrInSim
Gabriel's	-	RRT	DQN	$\tilde{p}$	$\tilde{p}$
RELAX (Ours)	**H-S**	**RRT**	**D3QN**	✓	✓

Table 1: Comparison between RELAX and other 2D-LiDAR UAV Frameworks: PP means path planning algorithm. Alg means the RL algorithm. DOA means dynamic obstacle avoidance. TrInSim means real-time training in Gazebo-ROS-PX4 simulator and H-S means Hector-SLAM algorithm.

Algorithm 4: RELAX

Input: limits, start, max_vel, max_acc, max_jerk, det_range
Output: None

for Number of Episodes **do**
  (x_t, y_t, z_t) randomly generated target position
  score 0, counter 0, Drone take off and go to start, last_req $Time.now()$
  Initial state s_0 LiDAR Data after Alg.2 LiDAR Data Filtering
  while not done **do**
    if drone.armed **and** $Time.now()$ last_req $>$ 6 **then**
      Select an action a_t with ϵ-greedy algorithm
      Drone execute the action a_t, detect_flag $True$
      New state s_{t+1} LiDAR Data after Alg.2 LiDAR Data Filtering
      done Eq.10 Check Done, Reward r_t Eq.9 Reward Function, score + = r_t
      Push $(s_t, a_t, r_t, s_{t+1}$, done) in memory buffer
      if size(memory buffer) B **then**
        Sample B transitions from memory buffer
        Every f_u steps update the θ_t with θ_p, α_t with α_p, β_t with β_p
        Do forward operation as Eq.3 for Q_{policy} and Q_{target}, get q_pred, q_next and q_eval
        q_target = r_t + γ q_next
        Update the parameter $\theta_p, \alpha_p, \beta_p$ for Q_{policy} on loss (q_target, q_pred)
      end if
      s_t = s_{t+1}, counter + = 1
    else
      Maintain connection with drone
    end if
  end while
  Call Alg.3 Reset
end for

Experiment – A Case Study

In this section, we demonstrate RELAX on addressing a real-life challenge within the agricultural context, specifically catering to scenarios where farmers seek to do equipment checks during nocturnal hours or after extreme weathers such as storms.

Hardware and software setup. The experiment is conducted on a desktop with Intel Core-i5-13400 CPU, Nvidia GeForce RTX4070Ti GPU and 64GB of RAM. The operating system is Ubuntu 20.04 bionic. The simulator is executed on Gazebo 11, ROS Noetic and PX4 v1.12.3..

Experimental environment setup. We firstly train our RL model in environment shown in Fig. 4 right and fine-tune it in environment shown in Fig. 5 left.

Table 2: Parameters used in training of obstacle handler

Parameter	Value
State dimensions N_{dim}	11
Action dimensions A_{dim}	8
Training episodes N_{eps}	500
Maximum step for one episode N_{step}	50
Memory pool size M	1 10^6
Batch size B	96
Target network parameter update frequency f_u	1000
Discount factor γ	0.99
Learning rate α_l	5 10^{-4}
ϵ-greedy possibility max ϵ_{max}	1.0
ϵ-greedy possibility min ϵ_{min}	0.01
ϵ-greedy decay factor ϵ_{decay}	1 10^{-4}

To examine the feasibility of our proposed framework, we established a test environment illustrated in Fig. 5 right. The agricultural land is divided into smaller zones by movable iron bars and wires to facilitate diverse crop cultivation. However, the dynamic nature of these iron bars, subject to seasonal rearrangements by farmers to accommodate varying crop types, presents noteworthy challenges, in which the static path planning based on a pre-scanned map becomes impractical. The iron bars, which are not shown during the map construction stage, are considered as dynamic obstacles in our experiments.

Results. To comprehensively check the generalizability of our trained model, we conduct 50 run each with around 20 iron bars distributed randomly in the testing environment and report the average success rate. The results are shown in Table 3, where the better variant of RELAX (with D3QN) achieves an average success rate of 90%, which is 8 times higher than the other 2D-LiDAR based algorithm, Gabriel's alg (Gabriel et al. 2023), making RELAX a practically usable solution in such agricultural applications. Additionally, the performance of RELAX is on par with other state-of-the-art algorithms requiring 3D LiDAR and RGB-D cameras, such as Deep PANTHER (Tordesillas and How 2023) and FAST-LIO (Kong et al. 2021), showcasing RELAX's competitiveness while keeping the total cost much lower.

Figure 5: Left: training environment used for fine-tuning after training in environment shown in Fig. 4 right. Right: a typical farmland environment, where delineated regions are labeled as *a; b; c*, and etc., and are separated by the movable iron bars and wires (shown as red dotted lines). The objective of the case study is to navigate a parsimonious UAV from house to tower for nocturnal inspections.

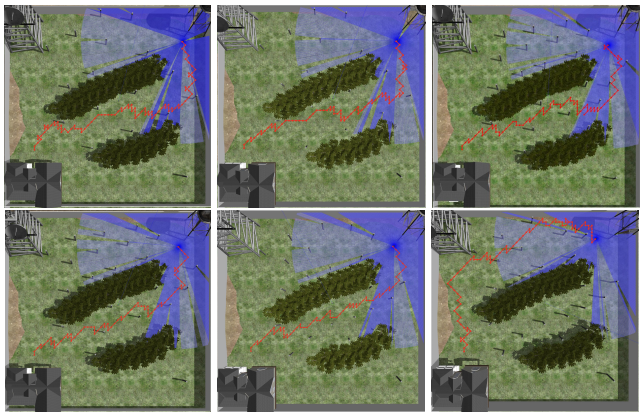

Figure 6: Trajectory paths (red lines) that UAV traverse in different experiments. Iron bars are randomly distributed in all experiments.

As an example, results from six sample experiments are shown in Fig. 6. The oscillatory patterns observed in the movements of our parsimonious UAV when in close proximity to obstacles, as depicted in the figures, distinctly illustrate its dynamic obstacle avoidance behavior. This behavior becomes particularly evident when the drone's distance from obstacles falls below the predefined threshold established during the training phase.

Moreover, we frequently find that certain algorithms excel in specific areas. For instance, as demonstrated in Table 3, Dijkstra's algorithm, despite achieving a lower success rate, boasts significant time efficiency. This variability in performance underscores the importance of allowing users to select algorithms tailored to their unique requirements, which highlights the value of the modular design of our framework, RELAX. This design facilitates easy integration and experimentation with emerging RL algorithms, offering a versatile platform for future research endeavors.

Conclusion And Future Works

In this paper, we introduce **RELAX**, a RL-based autonomous system for parsimonious UAVs that carry only one single 2D-LiDAR to successfully perform navigation in unknown environments. Rigorous feasibility tests confirm its effectiveness, showcasing a remarkable success rate

Algorithm	Time(SPP)	Time(ORP)	Success Rate
Genetic Alg	17.4s	-	12%
Dijkstra's Alg	**4.8s**	-	8%
Gabriel's Alg	13.5s	-	10%
Deep PANTHER	-	[0.01, 0.03]	**100%**
FAST-LIO	-	**[0.003, 0.013]**	98%
RELAX (DQN)	13.5s	[0.0003, 0.1]	82%
RELAX (D3QN)	13.5s	[0.0003, 0.05]	90%

Table 3: Performance Comparison between several algorithms and ours: For SPP (Static Path Planning), we mean planning a path on the map without iron bars, which mainly illustrates the performance difference between RRT and other algorithms. For ORP (Online Re-Planning), it denotes the time needed for online replanning to avoid the dynamic obstacles based on real-time 2D-LiDAR data, at which Genetic Algorithm (Tsai, Chou, and Liu 2006), Dijkstra's Algorithm (DIJKSTRA 1959), and Gabriel's Algorithm (Gabriel et al. 2023) are NOT able to handle. Deep PANTHER (Tordesillas and How 2023) is based on camera and FAST-LIO is based on 3D-LiDAR (Kong et al. 2021). Since the inference time of a trained RL model for each step differences a lot depending on state, we just record the range of time needed for one-step inference for illustration purpose. For success rate, it illustrates the percentage of dynamic obstacles (iron bars) the drone avoided during the path it took from starting point to the target. The average success rate of 50 tests for each algorithm are shown. The bold number in each column illustrates achieving best performance for this criterion among all algorithms.

of 90% in diverse scenarios, outperforming existing algorithms by a significant margin. In addition, we demonstrate RELAX's great potential as both RRT and D3QN can be replaced by more advanced algorithms and network structures to achieve more desirable performance.

Despite the success, ensuring precise 2D-LiDAR detection requires conservative speed settings, which limits our system's versatility. To mitigate this, we are investigating the integration of multiple 2D-LiDARs collecting data from different angles. By fusing these data, we aim to counteract the influence of the imperfect LiDAR readings, thus further broadening the application potential of RELAX.

References

Aggarwal, S.; and Kumar, N. 2020. Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges. *Computer Communications*, 149: 270–299.

Ahirwar, S.; Swarnkar, R.; Bhukya, S.; and Namwade, G. 2019. Application of drone in agriculture. *International Journal of Current Microbiology and Applied Sciences*, 8(01): 2500–2505.

Aldao, E.; González-de Santos, L. M.; and González-Jorge, H. 2022. Lidar based detect and avoid system for uav navigation in uam corridors. *Drones*, 6(8): 185.

Alvarado, E. 2021. 237 ways drone applications revolutionize business. *Drone Industry Insights*.

Bachrach, A.; Prentice, S.; He, R.; Henry, P.; Huang, A. S.; Krainin, M.; Maturana, D.; Fox, D.; and Roy, N. 2012. Estimation, planning, and mapping for autonomous flight using an RGB-D camera in GPS-denied environments. *The International Journal of Robotics Research*, 31(11): 1320–1343.

Barnhart, R. K.; Marshall, D. M.; and Shappee, E. 2021. *Introduction to unmanned aircraft systems*. Crc Press.

Cheng, C.; and Chen, Y. 2021. A neural network based mobile robot navigation approach using reinforcement learning parameter tuning mechanism. In *2021 China Automation Congress (CAC)*, 2600–2605. IEEE.

Cui, S.; Chen, Y.; and Li, X. 2022. A Robust and Efficient UAV Path Planning Approach for Tracking Agile Targets in Complex Environments. *Machines*, 10(10).

DIJKSTRA, E. 1959. A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik*, 1: 269–271.

Ding, Z.; and Dong, H. 2020. Challenges of reinforcement learning. *Deep Reinforcement Learning: Fundamentals, Research and Applications*, 249–272.

Elmokadem, T.; and Savkin, A. V. 2021. Towards fully autonomous UAVs: A survey. *Sensors*, 21(18): 6223.

Engel, J.; Sturm, J.; and Cremers, D. 2014. Scale-aware navigation of a low-cost quadrocopter with a monocular camera. *Robotics and Autonomous Systems*, 62(11): 1646–1656.

Gabriel, G.; Alvaro, M.; Jose, A.; and Milena. 2023. Adaptive Path Planning for Fusing Rapidly Exploring Random Trees and Deep Reinforcement Learning in an Agriculture Dynamic Environment UAVs. *Agriculture*, 13: 354–379.

Jeong, N.; Hwang, H.; and Matson, E. T. 2018. Evaluation of low-cost lidar sensor for application in indoor uav navigation. In *2018 IEEE Sensors Applications Symposium (SAS)*, 1–5. IEEE.

Jingjing, W.; De, G.; and Fei, L. 2019. Research on autonomous positioning method of UAV based on binocular vision. In *2019 Chinese automation congress (CAC)*, 3588–3593. IEEE.

Jones, M.; Djahel, S.; and Welsh, K. 2023. Path-planning for unmanned aerial vehicles with environment complexity considerations: A survey. *ACM Computing Surveys*, 55(11): 1–39.

Kim, H.; Kim, H.; Lee, S.; and Lee, H. 2022. Autonomous exploration in a cluttered environment for a mobile robot with 2d-map segmentation and object detection. *IEEE Robotics and Automation Letters*, 7(3): 6343–6350.

Kohlbrecher, S.; von Stryk, O.; Meyer, J.; and Klingauf, U. 2011. A flexible and scalable SLAM system with full 3D motion estimation. In *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*, 155–160.

Kong, F.; Xu, W.; Cai, Y.; and Zhang, F. 2021. Avoiding Dynamic Small Obstacles With Onboard Sensing and Computation on Aerial Robots. *IEEE Robotics and Automation Letters*, 6(4): 7869–7876.

Kularatne, D.; Bhattacharya, S.; and Hsieh, M. A. 2016. Time and Energy Optimal Path Planning in General Flows. In *Robotics: Science and Systems*, 1–10. Ann Arbor, MI.

LaValle, S. M.; and James J. Kuffner, J. 2001. Randomized Kinodynamic Planning. *The International Journal of Robotics Research*, 20(5): 378–400.

Liang, Q.; Wang, Z.; Yin, Y.; Xiong, W.; Zhang, J.; and Yang, Z. 2023. Autonomous aerial obstacle avoidance using LiDAR sensor fusion. *Plos one*, 18(6): e0287177.

Lu, Y.; Xue, Z.; Xia, G.-S.; and Zhang, L. 2018. A survey on vision-based UAV navigation. *Geo-spatial information science*, 21(1): 21–32.

Mao, T.; Huang, K.; Zeng, X.; Ren, L.; Wang, C.; Li, S.; Zhang, M.; and Chen, Y. 2019. Development of power transmission line defects diagnosis system for UAV inspection based on binocular depth imaging technology. In *2019 2nd International Conference on Electrical Materials and Power Equipment (ICEMPE)*, 478–481. IEEE.

Mishra, B.; Garg, D.; Narang, P.; and Mishra, V. 2020. Drone-surveillance for search and rescue in natural disaster. *Computer Communications*, 156: 1–10.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Murcia, H. F.; Monroy, M. F.; and Mora, L. F. 2018. 3D scene reconstruction based on a 2D moving LiDAR. In *Applied Informatics: First International Conference, ICAI 2018, Bogotá, Colombia, November 1-3, 2018, Proceedings 1*, 295–308. Springer.

Noreen, I.; Khan, A.; and Habib, Z. 2016. Optimal path planning using RRT* based approaches: a survey and future directions. *International Journal of Advanced Computer Science and Applications*, 7(11).

Ocando, M. G.; Certad, N.; Alvarado, S.; and Terrones, Á. 2017. Autonomous 2D SLAM and 3D mapping of an environment using a single 2D LIDAR and ROS. In *2017 Latin American robotics symposium (LARS) and 2017 Brazilian symposium on robotics (SBR)*, 1–6. IEEE.

Patil, D.; Ansari, M.; Tendulkar, D.; Bhatlekar, R.; Pawar, V. N.; and Aswale, S. 2020. A survey on autonomous military service robot. In *2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE)*, 1–7. IEEE.

Qin, H.; Meng, Z.; Meng, W.; Chen, X.; Sun, H.; Lin, F.; and Ang, M. H. 2019. Autonomous exploration and mapping system using heterogeneous UAVs and UGVs in GPS-denied environments. *IEEE Transactions on Vehicular Technology*, 68(2): 1339–1350.

Rovira-Sugranes, A.; Razi, A.; Afghah, F.; and Chakareski, J. 2022. A review of AI-enabled routing protocols for UAV networks: Trends, challenges, and future outlook. *Ad Hoc Networks*, 130: 102790.

Shahmoradi, J.; Talebi, E.; Roghanchi, P.; and Hassanalian, M. 2020. A comprehensive review of applications of drone technology in the mining industry. *Drones*, 4(3): 34.

Shin, H.; and Chae, J. 2020. A performance review of collision-free path planning algorithms. *Electronics*, 9(2): 316.

Tordesillas, J.; and How, J. P. 2023. Deep-PANTHER: Learning-Based Perception-Aware Trajectory Planner in Dynamic Environments. *IEEE Robotics and Automation Letters*, 8(3): 1399–1406.

Tsai, J.-T.; Chou, J.-H.; and Liu, T.-K. 2006. Tuning the structure and parameters of a neural network by using hybrid Taguchi-genetic algorithm. *IEEE Transactions on Neural Networks*, 17(1): 69–80.

Ulrich, L.; Vezzetti, E.; Moos, S.; and Marcolin, F. 2020. Analysis of RGB-D camera technologies for supporting different facial usage scenarios. *Multimedia Tools and Applications*, 79(39-40): 29375–29398.

Van Hasselt, H.; Guez, A.; and Silver, D. 2016. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30.

Van Nam, D.; and Gon-Woo, K. 2021. Solid-state LiDAR based-SLAM: A concise review and application. In *2021 IEEE International Conference on Big Data and Smart Computing (BigComp)*, 302–305. IEEE.

Wang, Z.; Schaul, T.; Hessel, M.; Hasselt, H.; Lanctot, M.; and Freitas, N. 2016. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, 1995–2003. PMLR.

Wieczorowski, M.; Swojak, N.; Pawlus, P.; and Pereira, A. 2021. The use of drones in modern length and angle metrology. In *Modern Technologies Enabling Safe and Secure UAV Operation in Urban Airspace*, 125–140. IOS Press.

Xu, L.; Song, B.; and Cao, M. 2021. A new approach to optimal smooth path planning of mobile robots with continuous-curvature constraint. *Systems Science & Control Engineering*, 9(1): 138–149.

Xu, Z.; Zhan, X.; Chen, B.; Xiu, Y.; Yang, C.; and Shimada, K. 2023. A real-time dynamic obstacle tracking and mapping system for UAV navigation and collision avoidance with an RGB-D camera. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 10645–10651. IEEE.