MEHUL DAMANI

(65) 93925912 | damanimehul24@gmail.com

EDUCATION

Nanyang Technological University (NTU) — **Singapore**
Bachelor of Mechanical Engineering, Minor in Mathematics — Class of 2022

- **CGPA - 4.94/5.00; ACT – 35/36**
- **Dean's List 18-19, 19-20**

Neerja Modi School (Central Board of Secondary Education) — **India (Jaipur)**

- 12th: A-Level Equivalent; Stream: Science; Percentage – **97.6%** — April 2017 – May 2018

WORK EXPERIENCE

New York University — **Singapore**
Research Intern (remote), *Mentor: Professor Lerrel Pinto* — Dec 2020 - Present

- Developing novel automatic curriculum generation methods for deep reinforcement learning agents.

Multi Agent Robotic Motion Laboratory (MARMot Lab), National University of Singapore — **Singapore**
Research Intern, *Mentor: Dr Guillaume Sartoretti* — April 2020 - Present

- Developing decentralized **reinforcement learning** solutions for multi agent pathfinding in complex grid-worlds.
- Implemented novel reinforcement learning algorithms to solve multi vehicle routing problems on dense railway networks for the **NeurIPS 2020 Flatland Challenge**.
- Integrated state-of-the-art reinforcement learning algorithms such as A3C and PPO into our distributed framework using Tensorflow.
- Constructed observation matrices, optimized loss functions (hyperparameter tuning) and utilized imitation learning to train and test more than 20 reinforcement learning models and achieved significant gains over previous benchmarks.
- Submitted paper as **first author** to **IEEE Robotics and Automation Letters and ICRA (under review).**
- **Reviewed a paper for ICRA 2021.**

Satellite Research Centre, NTU — **Singapore**
Research Assistant — Sept 2019 – April 2020

- Developed regression models to characterize drift and bias of sensors such as gyro and magnetometer.
- Implemented Arduino code to build continuous data acquisition framework for multiple sensors.

Temasek Laboratories, NTU — **Singapore**
Research Assistant — June 2019 – April 2020

- Developed, tested and integrated payload containing sensors such as Geiger counter, IMUs, and Chip scale Atomic Clock.
- Launched high-altitude latex balloon (HAB) in Malaysia to obtain data in near-space region at altitudes above 20 km.
- Analyzed balloon trajectories and performance by constructing various data visualization plots.

PUBLICATIONS

- **M. Damani**, Z. Luo, E. Wenzel, G. Sartoretti. PRIMAL2: Pathfinding via Reinforcement and Imitation Multi-Agent Learning – Lifelong. *Submitted to IEEE Robotics and Automation Letters and ICRA*.

ACADEMIC PROJECTS

Vertical Take-off & Landing Aircraft (VTOL) — **Singapore**
Mentor: Professor Ng Bing Feng and Professor James Wang — August 2019 – May 2020

- Developed control system for fixed wing novel Vertical Take Off and Landing Aircraft (VTOL) prototype using Pixhawk.
- Performed extensive on-ground testing to tune PID parameters controlling pitch, roll and yaw.

Vigilant Bot — **Singapore**
National University of Singapore Hack & Roll 2020 — January 2020

- Developed RNN-based embedded hardware device to detect distress calls conveyed through complex hand gestures.

Optimal Debris Deorbiting System (ODDS) — **Singapore**
Singapore Space Challenge — January 2020

- Devised mission concept report to deorbit space debris from low-earth orbit (LEO) using bidirectional ion thrusters.

AWARDS AND ACHIEVEMENTS

NeurIPS 2020 Flatland Challenge — **Singapore**

- Achieved 1st position worldwide in round 1 and 4th position worldwide in round 2 of the **NeurIPS Flatland Challenge**. — August 2020

Vicom Book Prize Winner **Singapore**

- Awarded book prize by NTU for being the top scorer in *MA2007: Thermodynamics* in a cohort of 500 students. July 2020

Kishore Vaigyanik Protsahan Yojana (KVPY) Scholar **India**

- Cleared two levels of screening to attain scholarship funded by the Department of Science and Technology, India. March 2017

National Talent Search Scholar **India**

- Awarded prestigious scholarship by National Council of Educational Research, India (Similar to National Merit in USA). May 2016

SKILLS

Skills: **Proficient** in Python, C, Machine Learning, Reinforcement Learning, Tensorflow, MATLAB, Arduino

Relevant Coursework: Deep Learning, Reinforcement Learning, Stochastic Processes, Control Theory, Probability & Statistics, Dynamics, Algorithms, Data Structures and Convolutional Neural Networks

CO-CURRICULAR & VOLUNTARY ACTIVITIES

TEDx Speaker **India**

- Delivered TEDx talk on **Black Holes and Time Travel** to describe intriguing relationship between time and gravity. June 2017

Publications Attached

1. **PRIMAL$_2$: Pathfinding via Reinforcement and Imitation Multi-Agent Learning - Lifelong**

   Currently being revised for resubmission at IEEE Robotics and Automation Letters.

   Currently under review for ICRA 2021 as well.

2. **Design and Development of VTOL**

   Published in the undergraduate proceedings of Nanyang Technological University, Singapore.

PRIMAL$_2$: Pathfinding via Reinforcement and Imitation Multi-Agent Learning - Lifelong

Mehul Damani[1,*], Zhiyao Luo[1,*], Emerson Wenzel[1], Guillaume Sartoretti[1,†]

***Abstract*— Multi-agent path finding (MAPF) is an indispensable component of large-scale robot deployments in numerous domains ranging from airport management to warehouse automation. In particular, this work addresses lifelong MAPF (LMAPF) – an online variant of the problem where agents are immediately assigned a new goal upon reaching their current one – in dense and highly structured environments, typical of real-world warehouse operations. Effectively solving LMAPF in such environments requires expensive coordination between agents as well as frequent replanning abilities, a daunting task for existing coupled and decoupled approaches alike. With the purpose of achieving considerable agent coordination without any compromise on reactivity and scalability, we introduce PRIMAL$_2$, a distributed reinforcement learning framework for LMAPF where agents learn fully decentralized policies to reactively plan paths online in a partially observable world. We extend our previous work, which was effective in low-density sparsely occupied worlds, to highly structured and constrained worlds by identifying behaviors and conventions which improve implicit agent coordination, and enabling their learning through the construction of a novel local agent observation and various training aids. We present extensive results of PRIMAL$_2$ in both MAPF and LMAPF environments with up to 1024 agents and compare its performance to complete state-of-the-art planners. We experimentally observe that agents successfully learn to follow ideal conventions and can exhibit selfless coordinated maneuvers that maximize joint rewards. We find that not only does PRIMAL$_2$ significantly surpass our previous work, it is also able to perform on par and even outperform state-of-the-art planners in terms of throughput.**

I. Introduction

Multi-agent pathfinding (MAPF) is a challenging NP-hard problem with numerous real-life applications such as surveillance, search and rescue, and warehouses [1], [2]. In particular, the goal of *one-shot* MAPF is to find collision-free paths for a team of agents from their start positions to their goal positions with the aim of minimizing a defined objective function, such as makespan (i.e., time until all robots are on target) or the sum of their path lengths. However, many real-world problems are dynamic and often require agents to tackle a series of targets instead of staying stationary after reaching the first one. *Lifelong* multi-agent pathfinding (LMAPF), is a variant of MAPF where agents are repeatedly assigned new goal locations and are required to reactively compute paths online [3], [4], [5], [6], [7], [8]. The performance of LMAPF is generally measured in terms of *throughput*, i.e., the average number of targets reached per unit time. In contrast to conventional one-shot MAPF, LMAPF poses additional challenges as it requires online algorithms capable of frequently replanning as goals change. LMAPF becomes even more challenging in densely populated, structured worlds typical of factory-like environments due to the high number of conflicts between individual agent paths. The main contribution of this paper is the introduction of PRIMAL$_2$, a distributed reinforcement learning framework that extends our previous work in one-shot MAPF, PRIMAL [9], to LMAPF for dense, structured warehouse-like environments. In this new framework, special emphasis is laid on achieving extensive implicit agent coordination during lifelong MAPF for arbitrarily large team sizes, while remaining fully decentralized and communication-free.

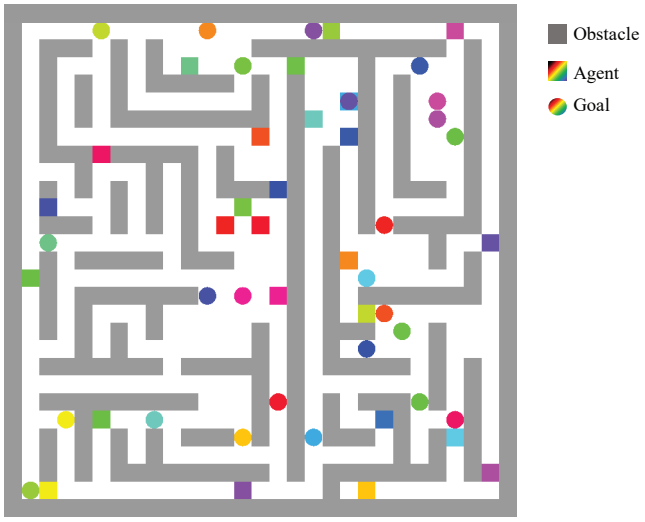

Fig. 1. Example of the type of highly structured environments we consider.

We focus on planning decentralized paths for a large population of agents in highly structured grid worlds, where obstacles compose narrow corridors that only allow one agent to pass at a time. To accomplish this, we rely on a threefold approach: first, we identify ideal behaviors and conventions that bring harmony to the movements of completely decentralized agents and enable the learning of such behavior through training aids and a supplemented observation. Second, we provide agents with an intuition about future states of their surroundings, by giving each of them access to their neighbors' predicted future movements (using single-agent A* and ignoring other agents in the system). Third, and drawing from the lessons of PRIMAL, we rely on imitation learning through a centralized planner to instill favourable

∗ These authors contributed equally to this work.

† Corresponding author, to whom correspondence should be addressed.

[1] Mehul Damani, Zhiyao Luo, Emerson Wenzel, and Guillaume Sartoretti are with the department of Mechanical Engineering at the National University of Singapore, 117575 Singapore. damanimehul24@gmail.com, e0452733@u.nus.edu, emersonwenzel@gmail.com, mpegas@nus.edu.sg

behaviors that are difficult to learn through vanilla RL. We also present a new distributed training code which relies on Ray [10] and shows significant speed gains over our previous works, by allowing us to train models in about 12 hours, compared to 10 days previously. Our full training code will be released publicly.

We present results of an extensive set of simulations containing up to 1024 agents for both one-shot MAPF and LMAPF, in dense, highly-structured environments. There, we experimentally demonstrate that PRIMAL$_2$ agents successfully learn to adhere to necessary conventions and execute coordinated manoeuvres which maximize joint performance without any explicit communication. Our results also show that PRIMAL$_2$ is able to significantly surpass the performance of our previous work and perform on par with state-of-the-art planners in many scenarios, which resemble real-life multi-robot deployments in structured environments.

The paper is structured as follows: Section II discusses the state-of-the-art in one-shot and lifelong MAPF. Section III formulates the specific LMAPF problem considered. Section IV proposes PRIMAL$_2$ and details the RL framework, while Section V describes how learning is carried out. Finally, Section VI presents and discusses the results from our simulations, and Section VII contains the closing remarks.

II. Prior Works

A. One-shot Multi-Agent Pathfinding

MAPF planners can be broadly divided into three categories: coupled, decoupled, and dynamically coupled. Coupled planners use the high-dimensional joint space to find complete and optimal paths but at a high computational cost. This scales exponentially with increasing number of agents [11], [12]. On the other hand, decoupled planners plan in the low dimensional space of each agent, and adjust paths to avoid collisions [13], [14], [15], [16], [17]. In particular, many recent works have started looking to machine learning methods to learn decentralized policies for MAPF [9], [18], [19]. Although significantly faster than coupled approaches, decoupled planners do not guarantee optimal solutions and are typically not complete. Dynamically coupled approaches lie between coupled and decoupled approaches, by dynamically increasing the search space when needed [20], [21], [22]. They are able to find (bounded-sub)optimal solutions without exploring the full joint configuration space.

In particular, our recent work, PRIMAL [9], proposed to address the trade-off between high-quality paths and scalability by relying on distributed reinforcement learning (RL) to teach agents fully decentralized reactive policies capable of computing individual paths online. Although PRIMAL scales well to arbitrarily large team sizes, it performs poorly in structured, densely occupied worlds that require substantial agent coordination to be solved effectively. To address this limitation of communication-free, decentralized MAPF planners, recent works have also proposed allowing agents to learn local communication and decision making policies in constrained environments using graph neural networks [19].

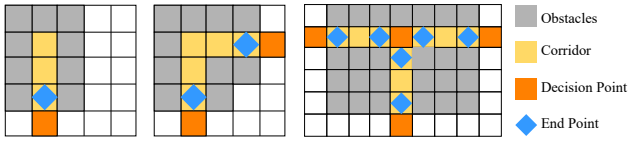

Fig. 2. Corridor examples: (a) dead-end, (b) usual corridor with two endpoints, and (c) combination of 3 corridors forming a T-junction.

However, these communication learning methods often suffer from poor scalability to larger teams.

B. Lifelong Multi-Agent Pathfinding

One of the common approaches to solve LMAPF involves stitching one-shot MAPF instances together by using a (usually complete, bounded-suboptimal) MAPF planner to recompute paths at each timestep at least one agent is assigned a new goal [4], [5], [23]. However, replanning time grows exponentially with the number of agents, and resources are wasted in the redundant computation of paths for agents whose goals are unaffected. Svancara et al. [7] propose adapted one-shot MAPF solvers for LMPAF, which reuse paths from previous planning iterations. However, dense, high-traffic worlds may contain many agents with conflicting paths, where significant replanning is still required. Some planners plan new paths for only the agents that have a new goal location, but have to resort to non-optimal techniques such as holding an agent's position or providing a dummy path [3]. Another very recent and promising approach is to plan paths within a finite window, which leads to better scalability and a more reactive algorithm [8]. However, this comes at the cost of *completeness*, which worsens when the window size is small in comparison to the average distance to goal, as agents cannot anticipate the situation outside the planning window and might plan greedy short-term paths that lead to unsolvable scenarios.

III. Problem Formulation

A. Environment Setup

In line with standard MAPF tasks, our environments are formulated as 2D discrete 4-connected grid worlds where agents, goals, and obstacles occupy one grid cell respectively. We consider highly structured worlds with moderate to high obstacle densities and long corridors. A typical example of the worlds we consider can be found in Figure 1. We create such worlds using a simple maze-generation algorithm: a randomized depth-first search, parameterized by the world size, the average obstacle density, and the typical corridor length. In this work, we further vary the average obstacle density and the typical corridor length to obtain a multitude of diverse worlds suited to train robust policies. Corridors impart structure to the world, but they are also potential bottlenecks as they can lead to deadlocks, and prudent planning is required to efficiently navigate them. Given their importance, particular emphasis is laid on corridors during the RL observation construction in Section IV-A.

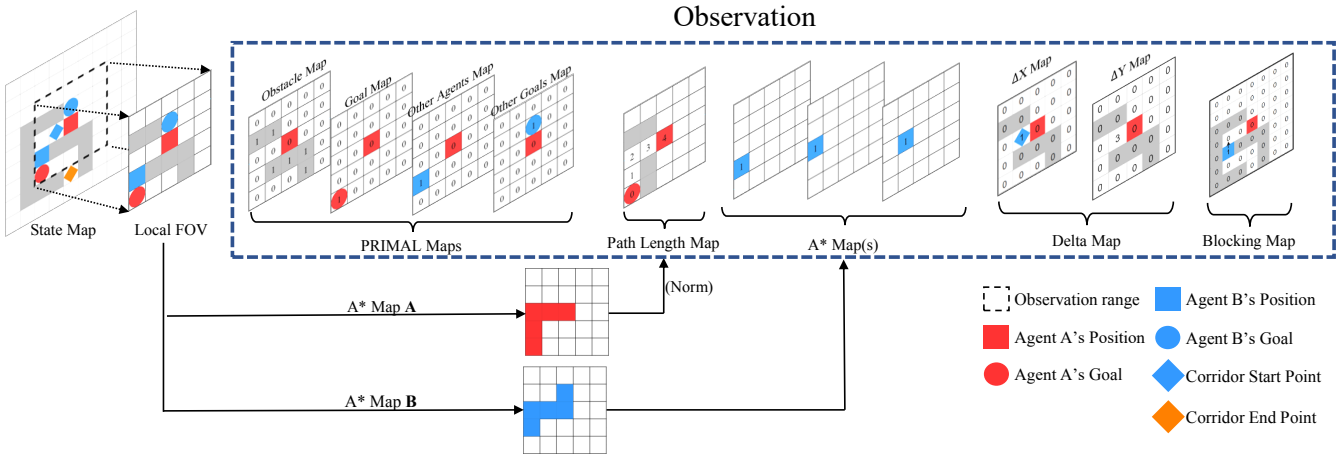

Fig. 3. Observation space of the agents (here for agent A, in red), as detailed in Section IV-A. The first four maps are identical to our previous work, providing information about obstacles, the agent's own goal, and nearby agents and their goals (e.g., agent B, in blue). The path length map displays the (normalized) shortest-path distance to the agent's own goal from all non-obstacle cells in the FOV. n_{pred} (here, 3) A* maps provide the future position of nearby agents, one per time step, predicted from single-agent A*. Finally, corridor information is encoded through the deltaX, deltaY, and blocking maps.

B. LMAPF Problem Formulation

As detailed in Section I, agents complete a series of tasks in LMAPF, and are required to recompute paths online as they are dynamically assigned a new goal. To achieve this, we construct a dynamic LMAPF environment where each agent is immediately assigned a new goal location upon reaching its current goal. The new goal's location is determined randomly and is constrained to be some minimum Euclidean distance away from the current goal. The agent has no prior information about its next goal's location. The LMAPF environment can be run indefinitely or terminated after a certain desired number of timesteps. The primary objective in our formulation of LMAPF is the maximization of agents throughput, i.e., the average number of targets reached per unit time per agent. Our constructed LMAPF environment effectively mimics real-world robot deployments in distribution centers where robots are dynamically assigned new tasks and are constantly in motion in an attempt to complete them.

IV. Policy Formulation

In this section, we describe our RL framework for LMAPF. In particular, we detail the observation and action spaces for each agent, the reward structure, and the end-to-end network structure used to represent the policy of each agent.

A. Observation Space

We consider a partially observable world where each agent can access the state of its surroundings within a limited square field-of-view (FOV) centered around itself (in practice, 11×11). We believe that, such a partially observable assumption is representative of real-world scenarios, where robots often only have access to incomplete information from their onboard sensors. Additionally, having a fixed, local FOV can help us learn a robust policy that can generalize to a wide range of world sizes while maintaining the same neural network structure.

In this limited FOV, information is separated into several channels to aid learning. Consistent with our previous works, four binary matrices provide information about obstacles, positions of other agents, goals of those observable agents, and the agent's own current goal position if within the FOV. [9]. We also provide each agent with a path length map that contains the shortest-path distance (normalized) to its goal from different positions within its FOV. This distance is calculated by running single-agent A*, ignoring all other agents in the environment. We believe that such a map resembles a gradient flow, enabling the agent to chart an effective (individual) trajectory even when its goal is far away and not observable in the agent's FOV.

We further introduce three smaller spatial maps (5×5 in practice, surrounded with zeros to reach 11×11) centered around the agents and encoding information about neighboring corridors. Because of their narrow structure, corridors are potential bottlenecks in the world, and hence need to be efficiently navigated.

Corridors are regions where agents have at most two possible actions, excluding staying stationary. Each corridor has two entry cells, barring corridors containing a dead-end that only have one. We refer to these entry cells as *Endpoints* and the cells outside the corridor connected to these endpoints as *Decision Points*. Decision points are named so, as agents occupying these cells have to take the critical decision of entering the corridor, which can potentially result in a future deadlock inside the corridor. Fig III contains some examples of corridors and illustrates the special points discussed above. All information about a specific corridor is encoded in the endpoint cells of that corridor. The first two maps, called delta X and delta Y, provide a corridor's orientation as a displacement between the two endpoints of that corridor. If a corridor is a dead-end, these maps are filled with zeros.

The third map, called the blocking map, contains information about other agents currently within a corridor. For an agent A which is outside a corridor, the blocking map contains a binary value which indicates whether there is any other agent currently inside that corridor moving in a direction which would cause it to exit the corridor from the

endpoint which is closer to agent A. Equivalently, it tells an agent whether it would be blocked on entering a corridor i.e., end up in a deadlock. This information is crucial for agent A to decide whether to enter a corridor.

In addition to specific corridor data, we believe that agents can benefit from having an idea about other agents' future movements. To this end, we let each agent construct a number, n_{pred}, of maps containing the predicted future position of other agents within its local FOV, one per map ($n_{pred} = 3$ in practice). These maps are generated using single-agent A*, under the assumption that each other agent is alone in the world.

Finally, an agent's goal is often outside its local FOV. To ensure that agents always have access to information about their goal and can plan longer-term actions, we provide each agent with a unit vector pointing towards its goal and the absolute magnitude of the distance to its goal at all times. Fig 3 details the complete structure of the observation space.

B. Action Space

We allow agents to take one out of five discrete actions in the grid world at every timestep: Moving one cell in any of the four cardinal directions or staying still. During training, actions are sampled from a list of valid actions, and agents are prevented from taking invalid actions, examples of which are moving into another obstacle or agent. Moreover, we also define some actions which fail to adhere to ideal predefined conventions about navigating corridors as invalid. These conventions and their corresponding invalid actions are detailed in Section V-A.1. A supervised loss function (i.e., valid loss) aids in learning valid actions. We experimentally observed that a supervised loss function works better in practice than rewards for learning the set of valid actions. This is possibly because learning valid actions does not depend on the preceding trajectory, and hence, bootstrapping with rewards can cause delayed and unfavourable convergence. Additionally, to prevent convergence to oscillating policies that prevent exploration and stall learning, agents are not allowed to return to the location they occupied at the last timestep. However, agents are allowed to stay still during multiple successive timesteps.

C. Reward Structure

To motivate agents to reach their goals quickly, we penalize them at every timestep they are not on goal ($r_t = -0.3$), as is common in most reward functions for grid worlds. Agents are also given a sizeable positive reward on reaching their goals ($r_t = +5$), which helps ensure that the immediate trajectories leading up to their goals are reinforced effectively. Finally, although agents are not allowed to take invalid actions as discussed in Section IV-B, it is still possible for them to collide with other agents in specific scenarios, such as two agents trying to move into the same empty cell. In such cases, agents are given a collision penalty ($r_t = -2$).

D. Network Structure

Our work relies on the asynchronous advantage actor-critic (A3C) algorithm [24], following the same network structure as our previous work [9]. We rely on a deep convolutional neural network to learn the approximate mapping between each agent's state and its policy output (discrete probability distribution over the 5 possible actions considered), parameterized by the set of weights θ. In this work, we reuse the same 11-layer deep convolutional network structure (CNN) from our previous work, which was itself based on a famous recent network architecture (VGGnet [25]).

The local observation channels are passed through two stages of three convolutional and one maxpooling layers, followed by one last convolutional layer to finally obtain a one-dimensional vector of features. In parallel, the goal unit vector and magnitude are first passed through one fully-connected (fc) layer. The concatenation of both of these pre-processed inputs is then passed through two fc layers, finally fed into a long-short-term memory (LSTM) cell. A residual shortcut [26] connects the output of the concatenation layer to the input layer of the LSTM. The output layers consist of the policy vector with softmax activation and the value.

During training, the policy and value outputs are updated in a batch when an episode finishes, as well as every time an agent reaches its goal (more details in Section V-B). The value is updated to match the total long-term cumulative discounted return $R_r = \sum_{i=0}^{k} \gamma^i r_{t+i}$ at every visited state during the most recent episode, using a standard L2 loss L_{value} (averaged over all states in the episode). The policy gradient loss (training the actor output of the network) reads

$$L_{actor} = \frac{1}{T}\sum_{t=1}^{T} \sigma_H \cdot H(\pi(o_t)) - log\Big(\overbrace{P(a_t|\pi,o;\theta)}^{\pi_t(a_t)} A(o_t,a_t;\theta))\Big), \tag{1}$$

where $H(\pi(o_t)) = -\pi_t(a_t) \cdot \sum_{i=1}^{5} log(\pi_t(a_i))$ is an entropy term to the policy loss, σ_H a small entropy weight (0.01 in practice), and $A(o_t,a_t;\theta)$ an estimate of the advantage function (see Eq.(2) below).

Recent works have shown that adding such an entropy term can help encourage exploration and discourage premature convergence [27]. As is standard in the advantage actor-critic algorithm, we use an approximation of the advantage function by bootstrapping using the value function (i.e., the output of the critic network):

$$A(o_t,a_t;\theta) = r_t + \gamma V(o_{t+1};\theta) - V(o_t;\theta)). \tag{2}$$

Besides the policy loss, we also rely on an additional loss to speed up the actor training, namely L_{valid}, which aims at reducing the log likelihood of selecting an invalid move to guarantee (thus training agents to reduce the activation of invalid actions):

$$L_{valid} = \frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{5} log(v_i(t)) \cdot \tilde{\pi}_t(a_i) + log(1 - v_i(t)) \cdot (1 - \tilde{\pi}(a_i)), \tag{3}$$

where $v_i(t)$ denotes the ground truth of action i's validity at time t (1 if valid, 0 otherwise), and $\tilde{\pi}$ is the result of a Sigmoid function being applied on π.

V. Learning

In this section, we detail our training framework for LMAPF . First, we describe the methods used to achieve

implicit agent coordination. Then, we state the parameters we use to set up the environment and the training. Finally, we present our new distributed training framework, which yields a significant speedup over our previous work.

A. Coordination Learning

In highly dense and constrained worlds with high traffic like those we consider, agents often find themselves in situations where coordination becomes necessary to find effective paths. While centralized planners can achieve this explicitly by planning in the high dimensional joint space, decentralized policies require agents to learn coordination implicitly with limited information about the environment and no direct control over other agents' actions. In the absence of any decentralized coordination learning, such as the techniques detailed in this section, we observed that agents distributively learn to act selfishly, merely trying to take the shortest A* paths to their goal and showing no coordinating behavior or regard for other agents' actions (even though coordination could lead to more optimal paths for themselves and a higher total reward for everyone).

In order to implicitly teach agents coordination, we use three techniques: identifying and forcing agents to learn certain conventions and exemplary behavior by using a supervised loss function (*Convention Learning*), using expert demonstrations from centralized planners during training (*Imitation Learning*), and sampling from a wide range of environments during training to enable learning of a robust, generalizable policy (*Episode Randomization*).

1) Convention Learning: In highly constrained worlds with a large number of long corridors, agents can drastically improve the quality of their paths if they learn a policy that adheres to a certain set of conventions and effectively breaks symmetries [28]. In this work, we have identified certain such conventions that are highly applicable to completely decentralized agents. For example, agents should never enter a narrow corridor if another agent is currently moving inside that corridor in a direction opposite to that of the agent, as this will lead to a deadlock. Similarly, agents moving inside a corridor should never reverse their movements abruptly and retrace their paths unless there is a deadlock (i.e., most other scenarios where agents follow such behavior are bound to be non-optimal). If agents learn to follow the conventions above, they can navigate corridors much more efficiently and find higher quality paths with more coordinated movements. However, these conventions are not evident to agents and are very difficult to learn using pure policy gradient methods, mainly because agents learn selfish policies, and rewards cannot effectively capture and reinforce such conventions.

To enable learning such conventions, we rely on our supervised valid loss function Eq.(3), which teaches agents to avoid taking actions that go against the above conventions. A metric called *valid rate* keeps track of the fraction of actions chosen by agents which are valid, i.e., the success rate in selecting a valid action. While the valid rate starts out low during training, agents are eventually able to learn to adhere to conventions and achieve a near perfect valid rate ($> 99.5\%$). Interestingly, following from the high valid rate, agents can also learn when they are inside corridors even though this information is not provided to them explicitly through the observation (i.e., only the endpoints of a corridor are evident in the observation). We believe that this is made possible by the LSTM cell in the network architecture, and future work will explore the integration of more powerful recurrent networks with our current architecture.

2) Imitation Learning: The combination of RL and Imitation Learning (IL) has been shown to lead to faster convergence and higher quality solutions in robotic applications [29], [30]. In LMAPF, IL from centralized near-optimal planners which plan in the joint space can instill good quality coordination behavior in agents, which is challenging to accomplish by decentralized RL. In line with our previous work, we experimentally observe that the addition of IL leads to faster convergence, more stable training and improved implicit agent coordination.

In this work, we rely on our new training framework to distribute RL and IL episodes to independent meta-agents, each containing one copy of the LMAPF environment in which a number of agents (8 in practice) learn in a synchronized manner (i.e., one timestep passes once all agents have selected and enacted a single action each). Meta-agents train independently, and update the global neural network weight asynchronously. The ratio of RL to IL episodes is maintained close to 50%, as in our previous work.

Expert demonstrations in IL episodes are generated by the bounded-suboptimal centralized planner ODrM* (with inflation 2) [20], and a trajectory of observations and actions is attained. We use these trajectories and the corresponding observations to minimize a standard behavior cloning loss:

$$L_{bc} = -\frac{1}{T}\sum_{t=0}^{T} log\left(P(a_t|\pi, o_t; \theta)\right). \tag{4}$$

Since ODrM* is a one-shot MAPF planner, several one-shot MAPF instances need to be combined for a single LMAPF environment as is common when adapting one-shot planners to LMAPF [4], [5]. As a result, during training in the LMAPF environment, ODrM* is called at all timesteps where path replanning is required, i.e., all timesteps where at least one agent reaches its goal location.

3) Environment Randomization: To ensure that agents encounter diverse environments during training, we randomize the world size, density, and typical corridor length at the beginning of each episode. Specifically, the size of our square worlds is varied between 10 and 70, the average obstacle density is set between 20%-70%, and the typical corridor length is an integer value between 3 and 21. We find that uniformly sampling these parameters works well in practice. Curriculum learning with increasing difficulty of environments has also been shown to be effective in practice [31]. However, our experiments with implementations of curriculum learning did not yield significant performance improvements. Future works might investigate this technique.

In our environment randomization process, we believe larger-sized worlds are necessary for agents to learn to navigate to their goal even if it is a significant distance away,

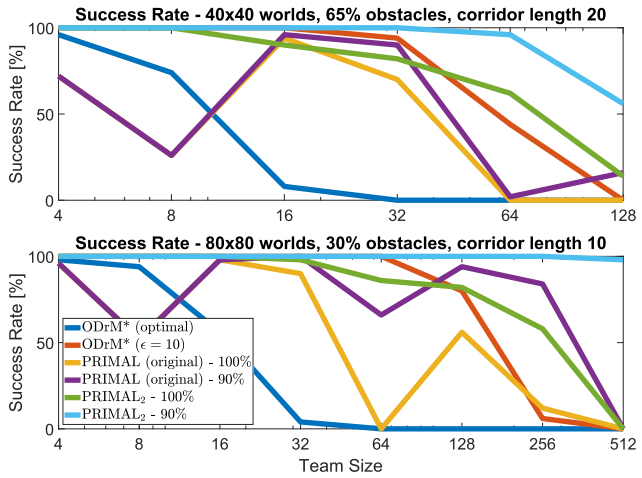

Fig. 4. Success rates of the considered planners in one-shot MAPF. As expected, PRIMAL$_2$ is slightly outperformed by centralized planners in very dense worlds with smaller teams, but outperforms all other planners with larger teams ($\geq$ 128 agents), especially with the considered 90% metric.

while the smaller sized worlds expose agents to cluttered scenarios which require coordination and collision avoidance to be solved effectively. The positions of agents, obstacles, and goals are set randomly across the world, with the constraints that each agent has at least one path to its goal, and the goal is some minimum Euclidean distance away from the agent (2 cells in practice). In addition to this, the agents' initial positions are constrained, such that there is no more than one agent inside any narrow corridor. This is primarily done to ensures that the conventions discussed in Section V-A.1 are adhered to since the beginning of the episode.

B. Training

1) General Training Parameters: In line with standard RL parameter choices, we use a discount factor (γ) of 0.95 and an episode length of 256. However, IL episodes have a length of 64 because of the high cost associated with repeatedly calling ODrM*. In addition to performing a gradient update at the end of the episode, we also perform one immediately after an agent reaches its goal. As a result, an agent may be trained more than once per episode, depending on the number of targets it reaches. Such training strategy decomposes LMAPF into a series of independent one-shot trajectories and has experimentally been shown to lead to faster convergence. We use the NAdam Optimizer with an initial learning rate of 2.10^{-5} and decay the learning rate proportionally to the inverse square root of the episode count.

2) Distributed Training Framework: We train our model utilizing Ray, a distributed framework for machine learning [10]. Ray allows us to bypass Python's Global Interpreter Lock and easily scale to a cluster using multiple GPUs. In practice, the final policy was trained on a single workstation equipped with a i9-10980XE CPU (18 cores, 36 threads) and one NVIDIA Titan RTX GPU. The code employs 9 remote training nodes, 4 of them calculating gradients via imitation learning with ODrM*, while the remaining 5 run pure RL episodes using the most up-to-date policy. The choice of these numbers has been made experimentally to keep the RL to IL episodes ratio close to 50%.

Each node is equivalent to a single meta-agent of the overall A3C architecture and contains a copy of the LMAPF environment in which 8 agents are learning to plan paths. All 9 nodes run in parallel and pass gradients to the master node to be applied to the global network asynchronously. Training lasts around 10 hours and converges within $35k$ episodes (nearly 10x fewer episodes and 24x shorter training time than our previous work). A non-Ray-based version of our code needs close to 4 days to converge on the same workstation, yielding a training speedup of nearly 10x for our new distributed learning framework. The full training code for PRIMAL$_2$ is available at `https://bit.ly/PRIMAL2`, and can easily be adapted to other MARL tasks.

VI. Results

This section presents and discusses our simulation results comparing PRIMAL$_2$ to state-of-the-art MAPF planners in a large number of diverse worlds. While the results of LMAPF are the main focus of this section, we also first provide results of one-shot MAPF for ease of comparison with various one-shot planners and our previous work.

A. One Shot MAPF Results

For all of our experiments (one-shot and LMAPF), we systematically tested team sizes in {4,8,16,....1024}, world sizes in {20,40,80,120}, densities in {0.3, 0.65}, and typical corridor lengths in {1,10,20}. We run 50 tests for each possible combination of the above parameters, barring a few infeasible combinations, and average the results in our plots. Specifically, we do not run tests containing 64 agents or more in 20-sized worlds, 256 agents or more in 40-sized worlds, and 1024 agents in 80-sized worlds. To eliminate any bias or noise in our results, all planners encounter the same test scenarios. Finally, invalid actions selected by agents during testing, i.e., crashing into a wall or another agent, are ignored, leaving the agent unmoved for that particular time step.

In our formulation of one-shot MAPF, agents vanish immediately upon reaching their goal, i.e., they are no longer considered as obstacles for other agents. This MAPF formulation is interesting for scenarios where the agent's goal location is a private space where the agent can reside without interfering with others, e.g., a parking spot in traffic applications [7], or a train station for railway operations [32]. We use makespan (i.e., time until all robots are on target) and success rate as our primary evaluation metrics.

We select ODrM* as both our optimal (inflation 1.0) and bounded-suboptimal centralized planner (with inflation factors $\varepsilon = 3$ and $\varepsilon = 10$) [20], with a timeout of 300s. We modified the ODrM* code to handle vanishing agents once on goal. We also use PRIMAL as a baseline MARL-based decentralized planner [9]. For PRIMAL and PRIMAL$_2$, we allow agents a maximum of 320 timesteps for 20 and 40-sized worlds, 480 timesteps for 80-sized worlds, and 640 timesteps for 160-sized worlds. For these two RL-based

planners, note that we trained separate dedicated models for both one-shot MAPF and LMAPF.

One of our recent studies into PRIMAL showed that unsuccessful episodes still often drive an overwhelming majority of agents to their goal [33]. Therefore, in our one-shot MAPF testings, we further provide PRIMAL/PRIMAL$_2$ results where we consider an episode to be successful when 100%, 95%, and 90% of agents reach their goals successfully, but only report the 100% and 90% values in our plots here for readability (same with ODrM* results with $\varepsilon = 3$). However, complete result plots are available at `https://bit.ly/PRIMAL2`, including LMAPF results. Figure 4 compares success rates in two interesting cases.

Based on our results, we first notice that all planners have very high success rates in worlds with low densities and short corridors. In moderate team sizes, we observe that PRIMAL$_2$ (with success metric 100%) is slightly outperformed by ODrM* but outperforms PRIMAL. In large team sizes (256 agents and above) with long corridors, the performance of both PRIMAL and PRIMAL$_2$ drops off gradually. However, the performance of ODrM* drops sharply, a common problem faced by many centralized planners, due to the exponential increase in the dimension of the joint configuration space to be searched. Interestingly, we note that while the performance of PRIMAL$_2$ with 100% success metric drops off gradually as the number of agents are increased, the performances of PRIMAL$_2$ with 90% and 95% success metrics stay nearly constant. Upon careful inspection of the results, we often find that a few agents get stuck in undesirable looping behaviour which can be corrected with the introduction of a centralized planner at this stage. Although PRIMAL$_2$ is able to find paths with a high success rate and scales well, we find that paths yielded by PRIMAL$_2$ tend to be considerably longer than ODrM* on average. Even though we observe a significant increase in implicit agent coordination compared to our previous work, the decentralized nature of PRIMAL$_2$ makes it very hard to achieve perfect joint coordination with the same quality of paths as other centralized planners.

B. LMAPF Results

In LMAPF, agents aim at continually planning paths online and maximizing the throughput (i.e., the average number of targets reached per timestep). To implement conventional baselines for LMAPF, we decompose the problem into a series of one-shot MAPF instances as is commonly done [23]. We select ODrM* (with inflation factors $\varepsilon = 10$), CBS, and iECBS as our bounded suboptimal planners [21], [20], [34] and use a timeout of 60s when using C++ implementations and 300s for python codes to remain consistent with other works in the field. For these conventional baselines, we count the number of goals that can be reached by agents until the end of the experiment (maximum number of timesteps), or until one sub-planning instance times out, whichever happens first. For all planners, our experiments last 128 timesteps in 20- and 40-sized worlds, 192 timesteps in 80-sized worlds, and 256 timesteps in 160-sized worlds. Figure 5 presents

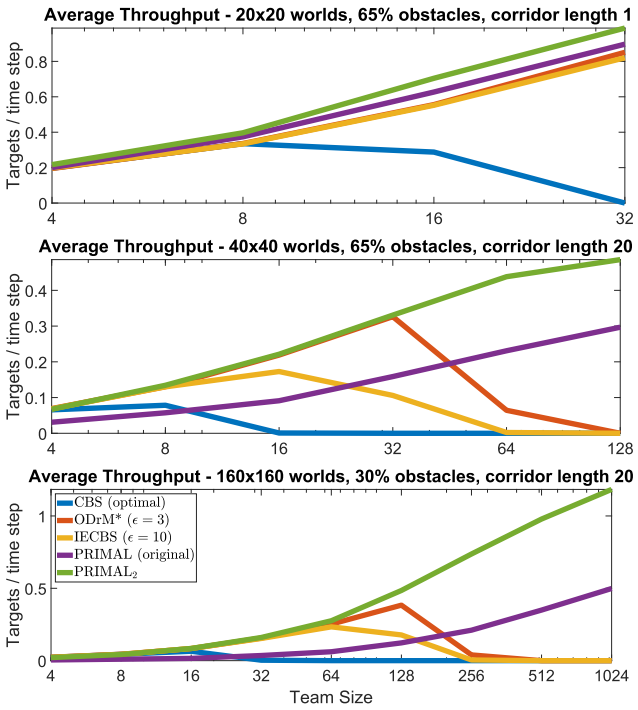

Fig. 5. Average throughput of planners for LMAPF. PRIMAL$_2$ performs on par with centralized planners for smaller teams, and outperforms them for larger teams due to its decentralized nature. The performance of all planners drop as environments become denser and require tighter coordination. There, PRIMAL$_2$ can be seen to significantly surpass PRIMAL, most likely due to the new coordination techniques introduced in this work.

the average throughput of our approach and the baselines considered in three representative scenarios.

Based on our results, we first observe that both centralized and decentralized planners have high throughput in worlds with low density, small team sizes and short corridors, since these worlds do not require significant joint coordination and are not very constrained. PRIMAL and PRIMAL$_2$ scale remarkably well to larger teams, while the performance of centralized planners drops sharply. However, as the typical corridor length and the average obstacle density is increased to make the worlds more constrained and challenging, we see drops in performance for all planners. We highlight that while PRIMAL and PRIMAL$_2$ have equivalent performance in worlds with low densities and short corridors, PRIMAL$_2$ easily outperforms PRIMAL in constrained worlds with long corridors. We believe that the additional coordination learning techniques introduced in this paper are the reason for these differences, and interesting learned maneuvers can be observed upon closer inspection of our PRIMAL$_2$ results. Video results of our approach on various environments and team sizes can be found at `http://bit.ly/PRIMAL2videos`.

Moreover, we also present averaged results over all worlds, which compare the performance of PRIMAL$_2$ to two other PRIMAL$_2$ variants: the first variant has no convention learning. In contrast, the second variant has no convention learning and a reduced observation with no corridor information channels in the agents' observation (Figure 6). We observe that, while these three planners perform near-identically up

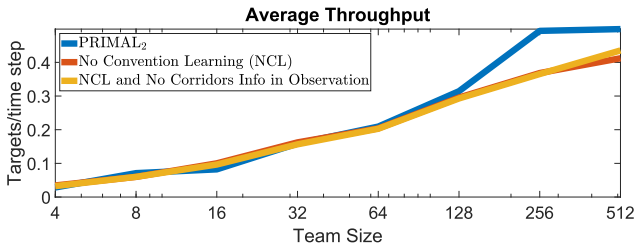

Fig. 6. Performance of PRIMAL$_2$, compared to two PRIMAL$_2$ variants: 1) no convention learning, and 2) no convention learning, and no corridor information in the observation. Note how, in larger teams ($\geq$ 128 agents), PRIMAL$_2$ outperforms these variants by more than 10% in terms of throughput, showcasing the importance of convention learning.

to 64 agents, PRIMAL$_2$ surpasses them in larger team sizes (by around 10%). We believe that these results show that for smaller teams, the gains from following conventions are neutralized by the higher freedom of movement enjoyed by not following any. However, as team sizes increase, these conventions become integral to plan effectively and bring order to agents' movements.

VII. Conclusion

This work introduced PRIMAL$_2$, a new distributed reinforcement learning framework for lifelong multi-agent path finding in highly constrained worlds. In this framework, agents plan individual paths online in a wholly decentralized way, based on local information but without communication for explicit joint coordination. We focused on achieving implicit agent coordination by helping agents learn ideal behaviour through conventions, which effectively break symmetries and bring harmony to their movement. Through our results, we highlighted the importance of these conventions in larger teams and experimentally showed that PRIMAL$_2$ agents are successful at learning them. We also showed that PRIMAL$_2$ scales to arbitrarily large teams and is able to plan effective paths online while producing throughputs at least on par with those of centralized planners. Finally, we also introduced our new Ray-based training code, which can train an order of magnitude faster than our previous works. Future work will try to further improve implicit agent coordination via a variety of techniques such as more powerful recurrent network architectures, the systematic investigation of RL-to-IL ratios, or the use of recent off-policy learning methods.

References

[1] K. Nagorny, A. W. Colombo, and U. Schmidtmann, "A service-and multi-agent-oriented manufacturing automation architecture: An IEC 62264 level 2 compliant implementation," *Computers in Industry*, vol. 63, no. 8, pp. 813–823, 2012.

[2] J. Berger and N. Lo, "An innovative multi-agent search-and-rescue path planning approach," *Computers & Operations Research*, vol. 53, pp. 24–31, 2015.

[3] H. Ma, J. Li, T. Kumar, and S. Koenig, "Lifelong multi-agent path finding for online pickup and delivery tasks," *arXiv:1705.10868*, 2017.

[4] M. Čáp, J. Vokřínek, and A. Kleiner, "Complete decentralized method for on-line multi-robot trajectory planning in well-formed infrastructures," in *Proceedings of ICAPS*, 2015, pp. 324–332.

[5] M. Liu, H. Ma, J. Li, and S. Koenig, "Task and path planning for multi-agent pickup and delivery." in *AAMAS*, 2019, pp. 1152–1160.

[6] V. Nguyen, P. Obermeier, T. C. Son, T. Schaub, and W. Yeoh, "Generalized target assignment and path finding using answer set programming," in *Proceedings of SoCS*, 2019.

[7] J. Švancara, M. Vlk, R. Stern, D. Atzmon, and R. Barták, "Online multi-agent pathfinding," in *AAAI*, vol. 33, 2019, pp. 7732–7739.

[8] J. Li, A. Tinka, S. Kiesel, J. W. Durham, T. Kumar, and S. Koenig, "Lifelong multi-agent path finding in large-scale warehouses," *preprint arXiv:2005.07371*, 2020.

[9] G. Sartoretti, J. Kerr, Y. Shi, G. Wagner, T. S. Kumar, S. Koenig, and H. Choset, "Primal: Pathfinding via reinforcement and imitation multi-agent learning," *IEEE RA-L*, vol. 4, no. 3, pp. 2378–2385, 2019.

[10] P. Moritz, R. Nishihara, S. Wang, A. Tumanov, R. Liaw, E. Liang, M. Elibol, Z. Yang, W. Paul, M. I. Jordan, and I. Stoica, "Ray: A distributed framework for emerging AI applications," 2018.

[11] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[12] T. S. Standley, "Finding optimal solutions to cooperative pathfinding problems," in *Proceedings of AAAI*, 2010.

[13] M. Erdmann and T. Lozano-Perez, "On multiple moving objects," *Algorithmica*, vol. 2, no. 1-4, p. 477, 1987.

[14] K. Kant and S. W. Zucker, "Toward efficient trajectory planning: The path-velocity decomposition," *IJRR*, vol. 5, no. 3, pp. 72–89, 1986.

[15] S. Leroy, J.-P. Laumond, and T. Siméon, "Multiple path coordination for mobile robots: A geometric algorithm," in *Proceedings of IJCAI*, vol. 99, 1999, pp. 1118–1123.

[16] M. Saha and P. Isto, "Multi-robot motion planning by incremental coordination," in *Proceedings of IROS*. IEEE, 2006, pp. 5960–5963.

[17] D. Silver, "Cooperative pathfinding," *AIIDE*, vol. 1, pp. 117–122, 2005.

[18] Y. Zhang, Y. Qian, Y. Yao, H. Hu, and Y. Xu, "Learning to cooperate: Application of deep reinforcement learning for online agv path finding," in *Proceedings of AAMAS*, 2020, pp. 2077–2079.

[19] Q. Li, F. Gama, A. Ribeiro, and A. Prorok, "Graph neural networks for decentralized path planning," in *AAMAS*, 2020, pp. 1901–1903.

[20] G. Wagner and H. Choset, "Subdimensional expansion for multirobot path planning," *Artificial Intelligence*, vol. 219, pp. 1–24, 2015.

[21] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artificial Intelligence*, vol. 219, pp. 40–66, 2015.

[22] M. Everett, Y. F. Chen, and J. P. How, "Motion planning among dynamic, decision-making agents with deep reinforcement learning," in *Proceedings of IROS*. IEEE, 2018, pp. 3052–3059.

[23] Q. Wan, C. Gu, S. Sun, M. Chen, H. Huang, and X. Jia, "Lifelong multi-agent path finding in a dynamic environment," in *Proceedings of ICARCV*. IEEE, 2018, pp. 875–882.

[24] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *ICML*, 2016, pp. 1928–1937.

[25] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *preprint arXiv:1409.1556*, 2014.

[26] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.

[27] M. Babaeizadeh, I. Frosio, S. Tyree, J. Clemons, and J. Kautz, "Reinforcement learning through asynchronous advantage actor-critic on a GPU," *preprint arXiv:1611.06256*, 2016.

[28] J. Li, G. Gange, D. Harabor, P. J. Stuckey, H. Ma, and S. Koenig, "New techniques for pairwise symmetry breaking in multi-agent path finding," in *Proceedings of ICAPS*, vol. 30, 2020, pp. 193–201.

[29] Y. Zhu, Z. Wang, J. Merel, A. Rusu, T. Erez, S. Cabi, S. Tunyasuvunakool, J. Kramár, R. Hadsell, N. de Freitas, *et al.*, "Reinforcement and imitation learning for diverse visuomotor skills," *preprint arXiv:1802.09564*, 2018.

[30] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, G. Dulac-Arnold, *et al.*, "Deep q-learning from demonstrations," *preprint arXiv:1704.03732*, 2017.

[31] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of ICML*, 2009, pp. 41–48.

[32] D. Roost, R. Meier, S. Huschauer, E. Nygren, A. Egli, A. Weiler, and T. Stadelmann, "Improving sample efficiency and multi-agent communication in RL-based train rescheduling," *arXiv:2004.13439*, 2020.

[33] G. Sartoretti, S. Koenig, and H. Choset, "A combined learning- and search-based approach to complete multi-agent path finding," in *Proceedings of the IJCAI workshop on MAPF*, 2019.

[34] L. Cohen, T. Uras, T. S. Kumar, H. Xu, N. Ayanian, and S. Koenig, "Improved solvers for bounded-suboptimal multi-agent path finding." in *Proceedings of IJCAI*, 2016, pp. 3067–3074.

Design and Development of VTOL

Mehul Damani
School of Mechanical and Aerospace Engineering

Ast Prof Ng Bing Feng
School of Mechanical and Aerospace Engineering

Abstract **–** Electric vertical take-off and landing aircrafts (eVTOL), which are powered electrically and have the capability to hover, take-off and land vertically, have been receiving increasing attention in the quest for fully autonomous passenger air vehicles (PAV) and aviation giants such as Boeing and Airbus have already developed demonstrator aircrafts. The primary objective of our research was to design and develop a scaled down novel eVTOL prototype. Multiple test flights on the prototype were used to validate our design and choice of parameters and also to validate electrical propulsion as a viable means for passenger air vehicles. Structural design of the prototype was first completed on SolidWorks and carbon fibre rods were used to construct the frame. Care was taken to prevent the localisation of stresses and custom 3D-printed joints were designed to minimise take-off weight. Due to its versatility, Pixhawk was chosen as the default controller. Altitude inputs were provided by a barometer and LIDAR, orientation inputs by an IMU and localisation was done through GPS. A 6S Lithium Polymer (LIPO) battery was selected as the power source and was projected to provide a 10-minute hover endurance. Initial test flights were highly unstable and this was attributed to a badly tuned flight controller. Subsequent flights involved progressive tuning of Pixhawk parameters and minor adjustments in design until level flight was achieved in Altitude-Hold mode. The prototype was finally put through endurance tests and was successfully able to achieve the 10-minute projected milestone.

Keywords – eVTOL, control systems, structural design

1 INTRODUCTION

1.1 BACKGROUND

Recent decades have witnessed rapid urbanization and increase in traffic congestion in major metropolitan cities of the world. Consequently, both private and public ground transportation systems around the world have come under heavy pressure [1]. There has been a lot of recent work on smart ground traffic optimization systems. However, the gains from optimization alone will not be enough to tackle the fast-rising congestion and as a result, there has been growing interest in unconventional means of transportation such as underground and in the air [2]. This has resulted in the development of aerial vehicle concepts for intra-city passenger transportation, also known as “Urban Air Mobility” (UAM). Among the different UAM vehicle concepts, electric Vertical Take-off and Landing Aircrafts (eVTOL) have received widespread attention, primarily due to their ability to take-off and land vertically in congested areas and due to the recent advancements in battery technology and electric propulsion systems [3].

Recent work has involved the development of a wide variety of VTOL demonstrator aircrafts. CityAirbus, a demonstrator aircraft by Airbus, began ground testing in 2018. Bell Nexus, a hybrid eVTOL, was unveiled by Bell Labs in 2019. Lilium Jet, a 5-seater eVTOL with 36 engines completed its maiden flight in 2019 [4].

1.2 OBJECTIVE

The objective of this paper is to describe the development, validation and ground testing of a scaled eVTOL prototype. First, this paper describes the structural design of the prototype and justifies the choice of design parameters. Then, this paper elaborates on the architecture of the control system and describes both the software and hardware used. Last, this paper describes the ground tests conducted on the prototype, the progressive modification of control parameters until level flight was achieved and the endurance milestones achieved.

2 STRUCTURAL DESIGN

A 1:17.5 scaled prototype was developed to validate the feasibility of eVTOL in UAM. Initial numerical analysis was used to obtain target parameters for the scaled prototype. For a wingspan of 1.2m, a Maximum Take-off Weight (MTOW) of approximately 4.7kg was arrived at. The VTOL consists of 8 upward facing propellers for lift and 2 forward facing propellers for thrust. A full list of the scaled prototype specifications is shown in Table 1.

Scaled Prototype Specifications		
Parameters	**Estimated**	**Actual**
MTOW (kg)	4.8	4.75
Payload(kg)	2.6	2.7
Cruise Altitude (ft)	150	-

Cruise Speed (km/h)	75	-
Range (m)	40	-
Wingspan (m)	1.19	1.2
Empty Weight (kg)	2.1	3.4

Table 1. Scaled Prototype Specifications

The initial structural design of the prototype was conducted on SolidWorks with the aim of minimizing weight. Endurance has a highly negative correlation with weight and minimizing weight was key to achieving the targeted endurance milestone. Custom connectors were designed with multiple joints to combine the functionality of different connectors. The main materials used in construction of the prototype were PLA, CFRP and ABS. CFRP was chosen as the primary load-bearing material due to its high strength-to-weight ratio which gave it an advantage of conventional aerospace materials like Balsa wood. The selection of CFRP was in line with the objective to minimize the prototype's weight.

Initially, hollow circular CFRP rods were chosen to construct the wing and body frames of the aircraft. However, ground testing with circular rods led to unstable flight performance, owing to the freedom in rotation of these rods about the longitudinal axis. This freedom in rotation caused misalignment in propellers' lines of thrust which introduced biases in pitch, roll and yaw manoeuvres. Thus, a switch was made from circular to square rods which resulted in much more stability and limited, correctable biases.

The conceptual design of the full-sized eVTOL vehicle is shown in Figure 1 and the frame structure of the scaled prototype is shown in Figure 2. As shown in Figure 2, the wing group consists of 2 longitudinal spars and 4 transverse booms to support the motors.

Figure 1. eVTOL conceptual design

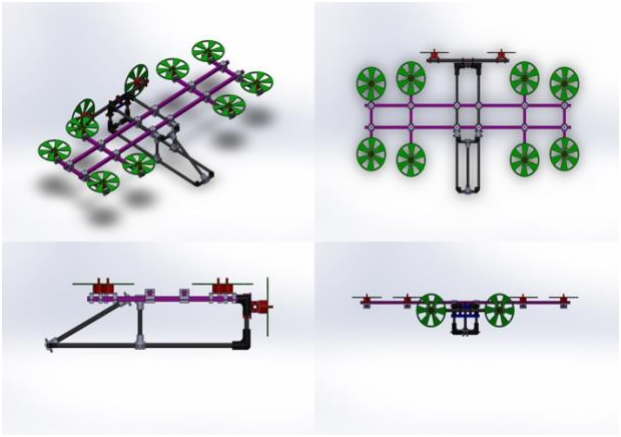

Figure 2. CAD drawings of frame structure

Although CFRP was used in construction of the frame, 3D printed parts are used for custom made designs such as mounting plates, connectors and the prototype body. The flexibility in design for 3D printed parts more than compensates for the lower strength-to-weight ratio. ABS is used as the printing material for critical components which are expected to be exposed to high stressed, such as the connectors connecting the wing and body groups. PLA is vulnerable to UV degradation, which makes it an unsuitable choice for these critical connections, particularly in outdoor flight conditions. However, PLA was used as the printing material for non-critical components housed in the interior of the prototype, such as the battery bay and the avionics components. The design of unconventional 3D-printed multifunctional connectors helped in significantly reducing the weight of the prototype. The entire outer body of the prototype was also 3D-printed, primarily due to its unconventional design and the requirement to minimize weight. Figure 3 shows the CAD drawings of the prototype with the outer body superimposed on it. Figure 4 shows the custom designed 3-D printed parts.

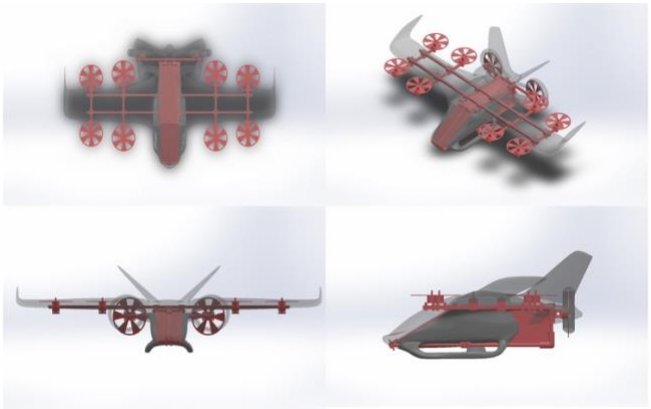

Figure 3. CAD drawings with outer body

Figure 4. 3D printed prototype parts

3 CONTROL SYSTEMS

3.1 SYSTEM ARCHITECTURE

The prototype's control system was built around Pixhawk, which is a general purpose open source flight controller [5]. The reasons for selecting Pixhawk are multifold. Pixhawk is highly versatile, and offers control architectures for a wide variety of airframes. It also offers support for numerous VTOL configurations, which are difficult to build from scratch owing to the high complexity of the VTOL transition phase. Additionally, the open-source nature of Pixhawk makes it highly customizable and the user has control of more than 200 parameters. The airframe of our prototype was a distorted version of the default OctoX architecture provided by Pixhawk as shown in Figure 5 [6]. As a result, significant control tuning was required to manually alter the default PID values until pitch, roll and yaw were stabilized.

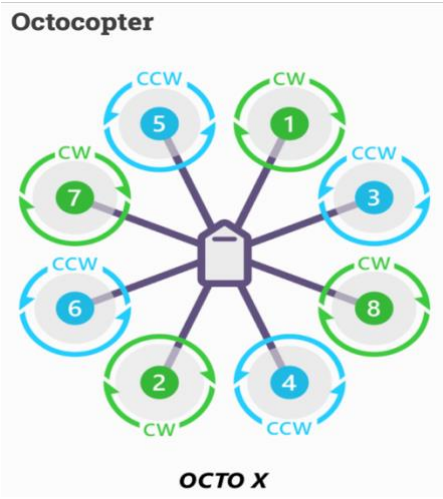

Figure 5. Pixhawk OCTOX Airframe

The pixhawk controller consists of many inbuilt sensors and multiple input ports to provide sensor data. Two inbuilt Inertial Measurement Units (IMU) provided orientation data to the flight controller. An externally connected GPS and compass were used for localization of the VTOL. A downward facing LiDAR was used to provide altitude data when close to the ground. This was necessary since the inbuilt barometer was unreliable and noisy when flying close to the ground. An Extended Kalman Filter (EKF) was enabled on the flight controller to fuse IMU and LiDAR data. A telemetry unit was connected externally to the Pixhawk to provide essential flight and battery status information in real time to the workstation. Finally, externally connected transmitters and receivers were used to enable manual line-of-sight control of the VTOL.

The input data to the Pixhawk controller was processed by the control algorithm which output 8 Pulse Width Modulation (PWM) signals to control the 8 brushless motors. The 8 PWM signals were first passed to 8 Electronic Speed Control (ESC) units. All ESC units were connected to a common 8-output power distribution board which was powered through a 6S Lithium Polymer (LiPo) Battery. A separate 3S LiPo battery was used to provide power to the Pixhawk board. The primary motive of doing so was to protect the Pixhawk board in case of battery failure or a power surge in the 6S LiPo battery which was being drained vigorously by the 8 motor system.

Table 2 provides a summary of the components used in the avionics system and Figure 6 provides the prototype circuit diagram.

Item	Specifications
Flight Controller	Pixhawk 1 Autopilot
GPS Module	Ublox 8N High Precision GPS Built-in Compass
LiDAR	Benewake TF-mini S
Telemetry Set	100mV 433 MHz FPV set
Receiver	Futaba R7008SB 2.4GHz
Transmitter	Futaba 14SGA 14 Channel 2.4 GHz
PDB	200A Power Distribution Board
ESC	FlyColor 45A BLHeli_32
Motor	T-motor F80 Pro KV1900
Battery	Turnigy 6S 10000mAh 12C LiPo Pack
	Turnigy 3S 2200mAh 30C LiPo Pack

Table 2. Avionics Components List

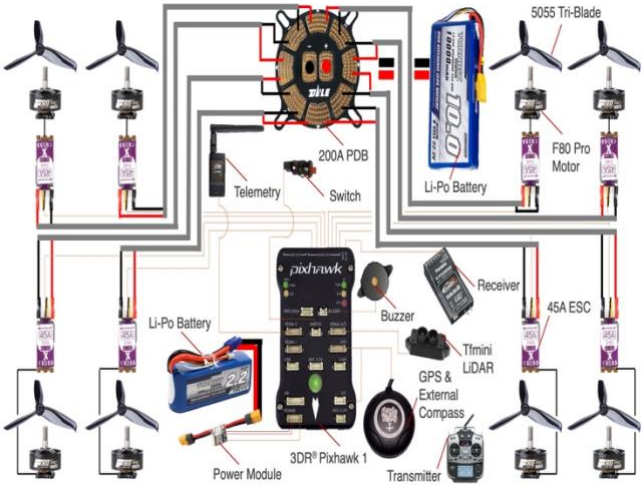

Figure 6. Prototype Circuit Diagram

3.2 CONTROLLER DESIGN

Our prototype had functionalities of both a multi-copter and a fixed wing aircraft. Thus, the control algorithm for the eVTOL was not constant but dependent on its flight mode – Hover, Transition or Forward Flight. PID controllers were used to drive the parameters to the desired set-points [7].

The complete pipeline for the controller is shown in Figure 7. An estimator uses multiple sensor inputs to compute the vehicle state. A controller takes the desired set-point and the current estimated state of the vehicle and outputs a corrective action to drive the current state towards the desired set-point. A mixer takes the controller outputs and maps them to individual motor commands. This mapping is specific for a vehicle type, and depends on various factors such as the vehicle's rotational inertia and the arrangement of motors with respect to the center of gravity.

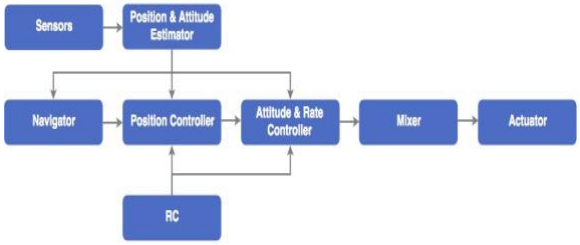

Figure 7. Control pipeline

During hover, the multi-copter controller will be in operation while during forward flight, the fixed wing controller will be in operation. During the highly complex transition manoeuvre, both controllers will work in conjunction to pass control gradually from one mode to the other.

4 TESTING

Initial testing on the prototype was conducted without the outer body attached, in an indoor environment with a take-off weight 500-700 grams below the MTOW. Since testing was conducted in an indoor environment, GPS was not accessible and the prototype didn't have the capability of very accurate localization. Initial flights showed heavy imbalance, with constant manual inputs needed to stabilize the VTOL. The heavy imbalance in initial flights was due to a variety of reasons. The Center of Gravity (CG) wasn't placed properly, primarily due to failure in accounting for the impact of the 6S LiPo battery on the CG position. More importantly, the initial flights were conducted with a badly tuned controller, and iterative improvement of PID parameters based on observations was necessary.

Subsequent test flights involved progressive tuning fixes to the controller [8]. Roll and pitch sensitivity parameters were decreased due to the aggressive nature of the VTOL. Initial flights had a tether attached to the prototype to constrain it and prevent major damage in case of a loss in control. As the VTOL became better tuned, the tether was removed to allow free climbing to a level altitude. The VTOL also faced significant imbalance close to the ground due to ground effects caused by the high speed propellers. PID tuning of the aircraft helped significantly and the VTOL gained stability and required fewer manual stick manoeuvres to stabilize. Figure 8 shows the prototype in stable flight.

Figure 8. Prototype in stable flight

The final phase of testing aimed to stabilize the prototype's altitude using 'Altitude Hold Mode' and to conduct endurance tests to verify the numerically calculated values. However, it was observed that the prototype dropped altitude rapidly whenever Altitude Hold Mode was activated [9]. This sharp drop in altitude is shown in Figure 9. This was initially attributed to a faulty altitude sensor, but the problem remained even after trying LiDAR, SONAR and Pressure Sensor as primary Altitude sensors.

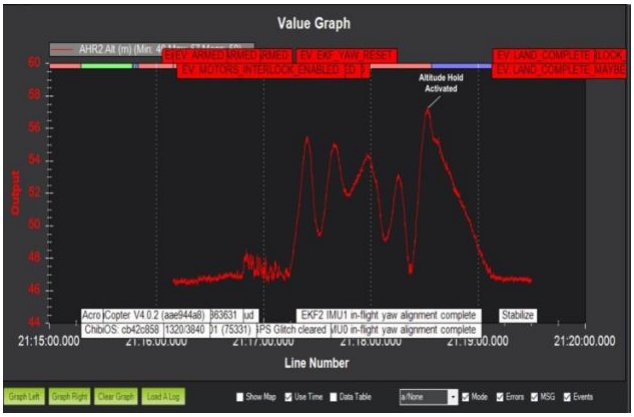

Figure 9. Initial Altitude vs Time graph

After analysis of various flight logs, it was found that the throttle required to hover was around 35%. The minimum throttle level required for the default Altitude Hold mode was 40%. Thus, when the throttle was at 35%, Altitude Hold mode executed a controlled descent which was the reason for the sharp drop in altitude. The minimum throttle level required was then lowered to 30% using the "THR_DZ" parameter, also known as the deadband parameter. Subsequent flights showed that our analysis had been correct, and our prototype was successfully able to maintain altitude as shown in Figure 10. The aircraft's PID parameters were then further fine-tuned increase roll and pitch stability. Finally, endurance tests were conducted on the prototype with a fully charged 6S LiPo battery. It was observed that the aircraft was successfully able to hover beyond the projected 7 minutes.

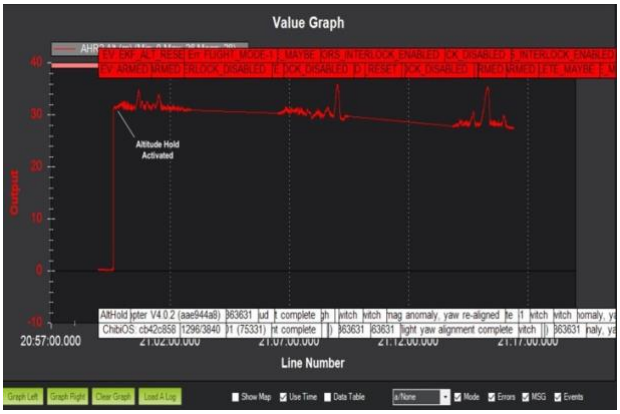

Figure 10. Altitude vs Time graph after Deadband correction

6 CONCLUSION

This paper described the design, assembly and control of a 1:17.5 scaled VTOL prototype. With rising populations and traffic congestion, the interest and demand in unconventional means of transport is only expected to grow and VTOL's are leading the way in Urban Air Mobility (UAM). Our prototype was able to achieve and exceed the projected endurance using electric propulsion. The increasing advancements in electric propulsion is another promising sign for the future of VTOL's. Extensive work was done on developing and tuning the control system and this was validated by the stability of the prototype in the final phases of testing. In the final phase of testing, the prototype is able to execute stable hover flight and corresponding manoeuvres. Future work will try to integrate the fixed wing controller with the multi-copter controller and successfully execute transition from one to the other.

ACKNOWLEDGMENT

I would like to express gratitude to Nanyang Technological University (NTU) for providing me with the resources and support to undertake research in VTOL development. I would also like to thank my supervisor Ast/P Ng Bing Feng from the school of Mechanical and Aerospace Engineering at NTU. His experience and insights gave direction and invaluable support to this project. Lastly, I would like to thank my team members, Mr Orion Dai Yuhui, Mr Jonas Erlmann and Mr Wong Zheng Xiong without whom this project would not have been possible.

"I would like to acknowledge the funding support from Nanyang Technological University – URECA Undergraduate Research Programme for this research project."

REFERENCES

[1] Organisation for Economic Co-operation and Development and European Con-ference of Ministers of Transport (ECMT), *Defining and characterising congestion*. Managing Urban Trac Congestion, Paris: Organisation for Economic Co-operation and Development : European Conference of Ministers of Trans-port (ECMT), 2007. Available: https://www.itf-oecd.org/sites/default/files/docs/07congestion.pdf.

[2] W. Broere, "Urban underground space: Solving the problems of today's cities," *Tunnelling and Underground Space Technology*, vol. 55, pp. 245–248, 2016.

[3] Y. Liu, M. Kreimeier, E. Stumpf, Y. Zhou, and H. Liu, "Overview of recent endeavors on personal aerial vehicles: A focus on the us and europe led research activities," *Progress in Aerospace Sciences*, vol. 91, pp. 53–66, 2017.

[4] EVTOL Aircraft Directory. (2020, June 22). Retrieved June 28, 2020, from https://evtol.news/aircraft/

[5] Dronecode. (2020, Mar. 30). Pixhawk 1 flight controller [Online]. Available: https://docs.px4.io/v1.9.0/en/flight_controller/pixhawk.html.

[6] Ardupilot. (2019) Motor order diagrams [Online]. Available: https://ardupilot.org/copter/docs/connect-escs-and-motors.html.

[7] Controller Diagrams. (n.d.). Retrieved June 28, 2020, from https://dev.px4.io/master/en/flight_stack/controller_diagrams.html

[8] Ardupilot. (2019). Basic tuning [Online]. Available: https://ardupilot.org/copter/docs/basic-tuning.html.

[9] Ardupilot. Altitude hold mode [Online]. Available: https://ardupilot.org/copter/docs/altholdmode.html.