Associative-Memory-Recall-based Control System for Learning Hovering Manoeuvres

Huang Pei-Hua
Tokyo Institute of Technology
Email: huang.p.aa@m.titech.ac.jp

Hasegawa Osamu
Tokyo Institute of Technology
Email: oh@soinn.com

Abstract—**This study presents a robotic application of neural associative memory-based control system that imparts online learning and predictive control strategies to a cost-effective quadrotor helicopter, the Parrot AR.Drone 2.0. The control system is extended with to tackle a fundamental and challenging problem for the quadcoptor: hovering control. The proposed system is based on self-organizing incremental neural network that includes an associative memory algorithm. The algorithm can cope with a hierarchical data space and complex time-transition dynamics; it enables online incremental learning from manual control, thereby gradually improve the stability against interference such as drift caused by either mechanical impairment or external excitation. In particular, after continuously learning the associative state-command pair of hovering manoeuvre, the system can execute the command associated with current state. The proposed system is evaluated on a realistic AR.Drone quadcoptor to test its capacity to tackle the hovering control problem. The results demonstrate that for the first time, the proposed system effectively offers a novel approach to quadcoptor application of an associative memory-based neural network by successfully tackling a hover task through iterative on-line learning.**

I. INTRODUCTION

Recent progress in autonomous control systems has led to a growing interest in quadcoptor applications and research. This is because of their ability to explore real world space through rapid manoeuvres and mobility. Consequently, companies recently intensified their interest in achieving greater manoeuvrability and control stability by offering small scale, commonly available, commercialized, and low-cost developmental platform for such quadcotor system, which have provided an ideal testing bench for not only the researches purposes on sophisticated control technologies but also for entertainment applications. However, despite the fact that a light-weight sensor solution for self-localization is still a critical and popular problems in robotic system, in terms of systematicity, the main challenge of the platform that we are focusing on is the absence of an autonomous control solution. Historically, the control of robots is pervasively preprogrammed to safely meet the expectations, such as navigating through space and interact with people in everyday life. We assume that the program is based on code and models that make interaction with them challenging.

In this study, we consider the concept of an associative memory (AM)-based approach, which decreases the complexity of interaction between robots and human by enabling the program to learn from environments, and recalling what has been learned in the past to impart a generalized estimation about the current environments. The AM is inspired by the brain mechanism using which it acquires new knowledge, with memorization being a critical mechanism of human intelligence. We assume that the AM is also crucial for applying human intelligence to permit mechanical agents and robots to acquire new motion or knowledge, which allows a robot to complete complex tasks with sufficient accuracy and experience through practice that previously only humans could perform. Few studies have adopted the AM approach for robotic applications in intelligent robots [1] [2]; the AM is expected to efficiently process with large data storage capacity through compression, and it can perform well on online incremental robot learning. This study provides an AM-based neural network to synthesize autonomous control of a quadcopter; the AM-based neural network demonstrated guaranteed performance with generalized estimation of control, when dealing with a noisy environment in real practice.

We chose quadcopter for our study because it allows more freedom for analysing our proposed system. A quadcopter is a helicopter that is mounted with four independent, fixed propellers and motors; varying the speeds of the four corresponding motors interactively provides it with six degrees of freedom to navigate around space. Generally, a commercial quadcopter is also equipped with adequate inexpensive on-board features, such as monocular camera, inertial navigation system (INS), ultrasonic sensor (used for altitude less than around 1 m), and controller processor. Siegwart et al. [3] has analysed and described the abilities of quadcopters in detail. In this study, we address the problem of hovering control by reducing the quadcopter drift error in GPS-denied unknown environments using a learning based approach. The preliminarily described aspects of the proposed algorithm are based on a neural AM network that adapts the drift reduction to autonomous hovering control in disruptive conditions; the overall goal of the system is to offer an alternative method to achieve stable and precise hovering manoeuvres by learning control commands supervised by a human operator. In the present demonstration of our system, it is using only an on-board camera, essentially with a monocular Simultaneous localization and mapping system (MonoSLAM) [4], which has been demonstrated to work well enough with a single inexpensive camera [5]; while the quadrocopter is flying blindly in space, MonoSLAM provides posture estimation to the control system. A survey revealed that the fully functional MonoSLAM system working in real-time visual tracking is the Parallel Tracking and Mapping (PTAM). PTAM has been used to conduct similar researches using an on-board camera [6] to stabilize a quadcoptor with modelling control. The results showed that PTAM provides a robust framework for visual tracking that is adaptable to the testing platform of our proposed algorithm.

The motivation behind our work is to showcase a scale-aware and simple topological AM-based control synthesis that is feasible for the hovering of a low-cost quadcoptor. As the hardware platform is limited, we use the Parrot AR.Drone that is available commonly in toy store, with a light weight of only 420g and a protective hull, safe to use in public. As the on-board computational resources are very limited, all computations including the proposed algorithm are performed externally on a modest notebook processor through WiFi port, the AR.Drone transmits visual information to PTAM for posture estimation and receives estimated manoeuvring commands with the given posture from the system. During the experimental procedure, the AR.Drone is initially controlled by human operator, who "instructs" the AR.Drone to perform hovering manoeuvres through a game-pad controller. Sets of associative state-command pair are created by collecting data from PTAM and controller. The system learns the manoeuvre from these associative pairs in real-time, once sufficient association are learned, the neural AM-based control system can recall the associative command response by providing the current posture to perform hovering control. The system also allows real-time updating to improve the setting of tasks without an off-line retrain. The contribution of this study is two-fold: first, our approach leverage the use of model-free algorithm for robotic exploration, and second, it demonstrates a neural AM-based control algorithm is feasible for enabling a highly dynamic robot to learn from the unknown environment in real-time, without any advance knowledge from static or dynamic modelling.

II. RELEVANT WORK

In the context of developing a flight control system, examples include the dynamic control approach on large-scale quadcoptor that is designed by Pounds et al. [7], and the model-based algorithm presented by Hoffmann et al. [8] for autonomous fly. However, the conventional model-based approaches can not satisfy our requirements for autonomous navigation throughout space, because most of these control system are task-specified, which commonly rely on discovered or pre-defined dynamic flying models of the robots and their interactive environments, these control approaches are impractical for autonomous navigation in complex indoor or outdoor environments, which are subject to changes. Additionally, The requirements for prior knowledge of interpretable and meaningful dynamic models cause the target robots to perform aggressive movements with no generalization, and significant issues of manoeuvres replication are persistent in other robots.

In contrast to learning based methods, which often proceed in reverse, by relying on model identification or system synthesis, constructing system model from data-driven observations. This process requires data collection to learn effective control strategies. The learning based approach benefits from having no need for pre-defined complex models to develop a system. Research has shown good performance using learning based control approach in largely two categories, open- and close-looped control, for close-loop, e.g., Calise [9] and Chowdhary et al. [10] outline the framework for concurrent learning

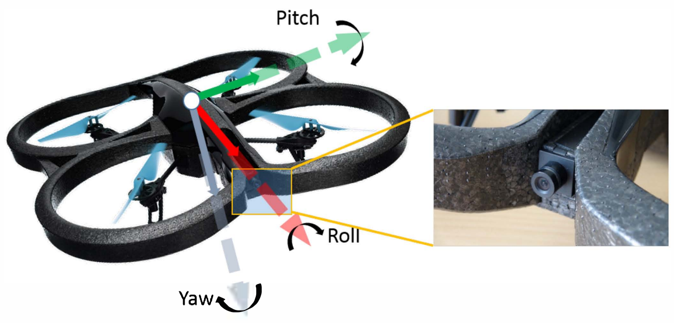


Fig. 1: Parrot AR.Drone 2.0, the experimental platform with its frontal camera and body reference frame

adaptive control, which guarantee improved learning and stability with selected and recorded data, David Nodland et al. [11] have developed neural network-based optimal control with output feedback, Efe [12] have introduced a neural network-based controller that learned Proportional, Integral and Derivative(PID) control. These previously proposed active control algorithms are mostly based on the strategy of closed-loop controller, which by optimizing the feedback of control output from robot, this results in the risk of oscillatory output response, also makes the system relatively more complex and costlier to construct than open-loop controller; Open-looped control system is not engaged in learning based method, however it does not encounter the problem of oscillation, hence can be embedded to improve the performance of control system, e.g., Abbeel [13] have shown an example of applying open-looped control through mimicking an expert performance in the learning process for ensuring effective performance learning. The proposed AM-based system can be seen as dual controller architecture through the combination of both open- and close-loop approaches, as the system self-organizes and incrementally learns from the closed-loop control fashion for a period of time. Then, it controls the robot in an open-loop fashion. This maintains the stability and simplicity in the open-loop part, as well as the accuracy and adaptation to dynamic changes in close-loop part. In addition throughout the experimental observations, we found that the proposed AM-based controller generates similar control outputs to the modern optimal control strategy, e.g., Linear Quadratic Regulator(LQR) [14] and the $H\infty$ [15] techniques, hence the learned dynamics of the proposed system can be evaluated by comparing with tuned PID control (assumed a simplified dynamics) in terms of precision and stabilization.

III. APPROACH

Given that substantial navigation information and manual control commands are collectable from the AR.Drone platform (shown in Fig. 1), we assume that there are unexplored associations among these data, which largely related to the dynamic control model of the AR.Drone. The proposed system uses the Self-Organized Incremental Neural Network, SOINN, combed with AM-based predictive control method, which is capable of capturing the associations among the data. This turns AR.Drone behaviour into a regular pattern, and uses the pattern to estimate control command from the current state of drift error.

The proposed AM-based neural network control algorithm has two phases: learning(training) and predicted control(testing). The proposed system can switch back and forth between these two phases at any time, allowing it to achieve consistent learning from the complex environments in real time.

A. Quadcopter Platform

The main hardware required is a Quadcopter. During last few couple of years, the quadcoptor platform has become widely available to the public, regardless of whether it is commercialized or open-sourced. This study uses the quadcoptor that is available from Parrot Corporation, named AR.Drone 2.0 (see Fig. 1), a low-cost, light-weight quadcopter, compared with other research purposed quadcopters, such as Ascending Technologys Pelican or Hummingbird quadcopters. AR.Drone was released to the market in 2010 as a high-technology toy, this type of quadcopter has been used in several researches. Its robustness against accidental crashes is particularly suitable for this study because it requires practice during training, and it can safely be used in public for demonstration. However a trade-off exists between its price and flexibility of customization, the on-board hardware or firmware cannot easily be modified, and that communication with the quadrocopter is only through 802.11g Wi-Fi connection, this limits the range of flying. The AR.Drone weights roughly 420g, when fully geared with the 1500 mAh battery and protective hull, measures $53cm \times 52cm$, it has an endurance of operation with fully charge batter of 15 minutes when carrying no payload.

The AR.Drone is mounted with two cameras (forward and downward camera). The forward cameras covers a field of view with a resolution of 640×480 pixels, and has rolling shutter with a delay of 40 ms between the first and the last line captured. The frame rate is streamed at 15 fps, using lossy compression, whereas the downward camera has a resolution of 167×144 pixels, at 60 fps. The controller board is based on an ARM cortex A8 processor running at 1 GHz, with separated video digital signal processing at 800 MHz used to process the video image data. The AR.Drone initially sends drone's status to the end-user connected to it. After AR.Drone acknowledges to the end-user's signal by continuously sending the navigation data to the end-user every 10 ms through 802.11g Wi-Fi connection.

B. ASSOCIATIVE PAIR

The concept of associative pair represents as a fuzzy rule, which in a form of "If *input* THEN *output*"; this study defines the input as an associative key (the posture state) and the output is a corresponding value(responded command). For every given input associative pair a with respect to time t, such that $a_t \in \mathbb{R}^{\omega+\omega}$ for a given length of interval of time series input (ω), which can be derived from

$$a_t = \begin{bmatrix} f_t \\ r_t \end{bmatrix} \quad (1)$$

where the $f \in \mathbb{R}^{\omega}$ is the feature vector of that associative key that comprises $(f_t^1, f_t^2, ..., f_t^{\omega})$, and $r \in \mathbb{R}^{\omega}$ is the corresponding value, which comes with $(f_t^1, f_t^2, ..., f_t^{\omega})$. The associative pair f and r rise the concern regard to different measurement unit (i.e. posture state(m) $\leftrightarrow$ control command) during clustering, which may cause error when the euclidean distance is calculated due difference in scale size. Therefore, the associative pair with same unit is grouped when performing training and testing, i.e., posture state with posture state and command with command.

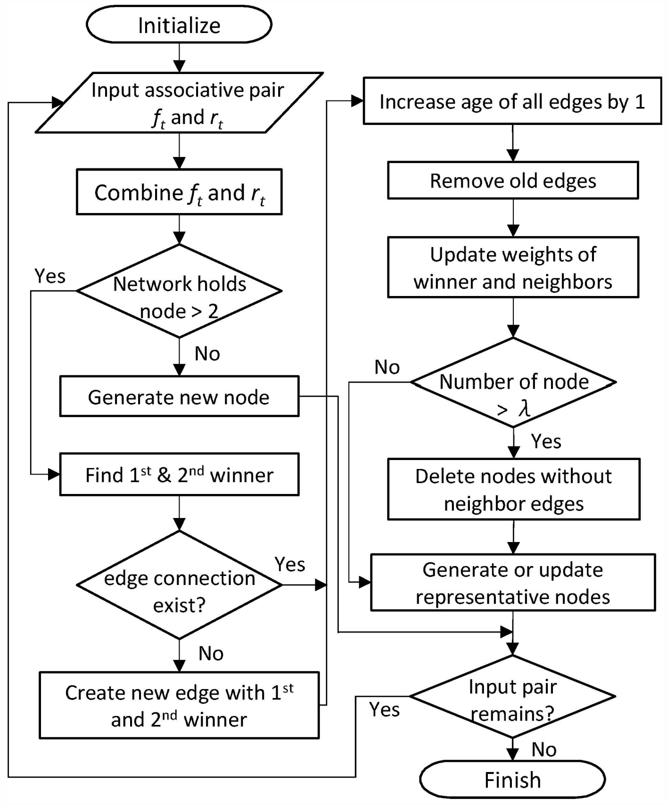

Initialize
Input associative pair f_t and r_t
Combine f_t and r_t
Yes
Network holds node > 2
No
Generate new node
Find 1st & 2nd winner
edge connection exist?
Yes
No
Create new edge with 1st and 2nd winner
Increase age of all edges by 1
Remove old edges
Update weights of winner and neighbors
No
Number of node > λ
Yes
Delete nodes without neighbor edges
Generate or update representative nodes
Yes
Input pair remains?
No
Finish

Fig. 2: The flowchart for SOIAM algorithm training an associative pair of input data ft and rt.

C. SOINN-AM

The Self-Organizing and Incremental neural Associative Memory (SOIAM) is an extension of SOINN that incorporates the AM, which is an incremental, on-line algorithm that is developed to deal efficiently with associative pair data, Fig. 2 shows the training framework. In previous works, its performance has been demonstrated to be better than comparable methods [16]. SOIAM generally begins training from an empty set that considers the first two input associative data as the starting nodes, and then, when SOIAM holds more than two nodes, it finds the first- and second-winner nodes (I) by calculating the minimum distance between the given associative a_t and the dimensional weight vector W_i of the ith node in all SOIAM nodes (S), i.e.

$$I_{1st} = \underset{i \in S}{\text{argmin}} \|a_t - W_i\| \quad (2)$$

$$I_{2nd} = \underset{i \in S - W_{1st}}{\text{argmin}} \|a_t - W_i\| \quad (3)$$

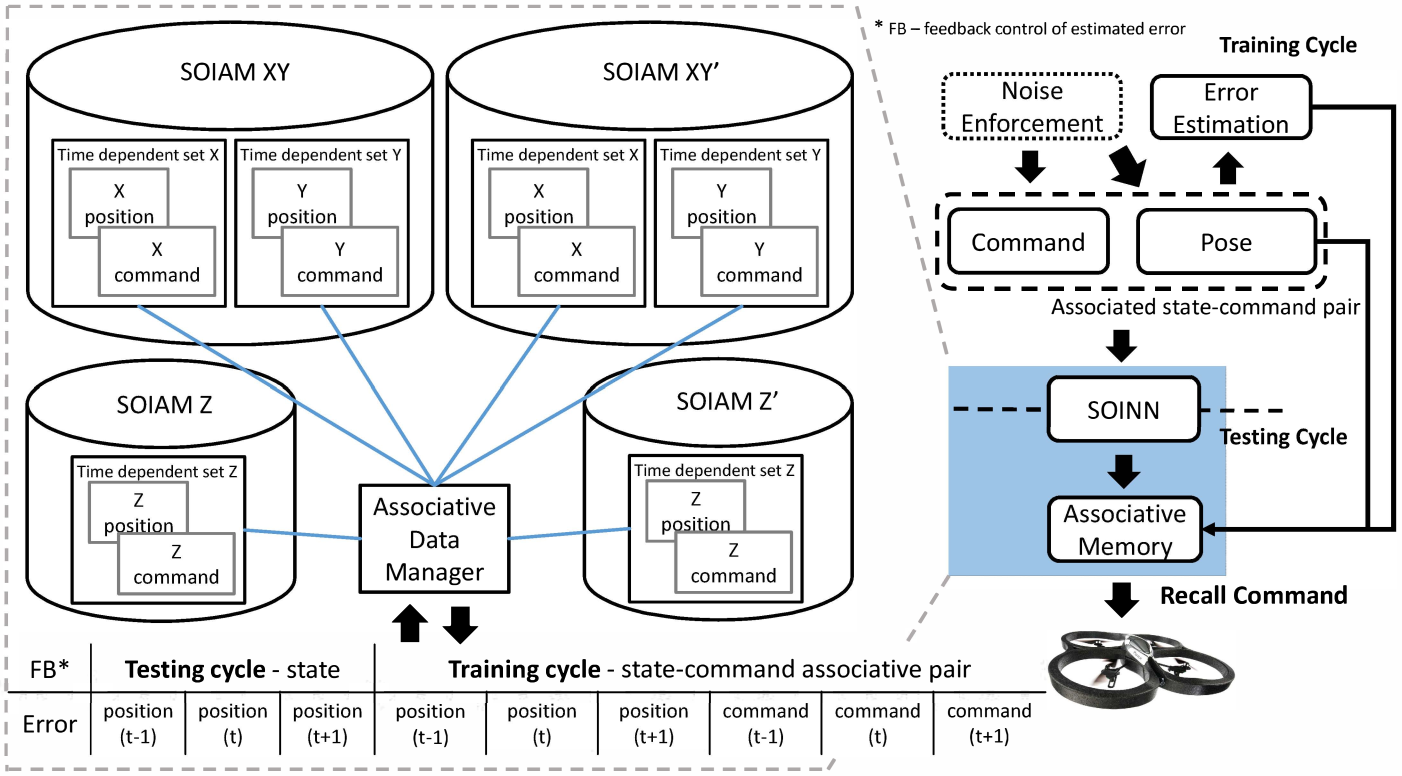


Fig. 3: Designed database structure for learning and recalling of the system organized with an associative data manager that deals with incoming and outgoing associative pair data.

If the distances between new associative input and the first- or second-winner are less than the similarity threshold that defined in [17], the input pattern becomes the first-winner node. Otherwise, SOIAM creates a new node for the new associative input in current network. In the case in which a new input associative pattern is assigned to the nearest node in the SOIAM, the weight vector is updated by the value of new input pattern and an edge between the first and the second winners is created (if edge does not exist). The SOIAM network clustering behaves similarly to SOINN [17], the cluster is formed by connecting existing nodes in SOIAM, this is largely different from other clustering techniques, such as k-means, where a new input is directly added to form the cluster. Therefore significantly saves computational memory when running for an extended time. Within the SOIAM network, each edge is assigned an age value, therefore every time a node is selected as the first winner, the age increase by 1, the entity permits each node to be removable by setting an age threshold Λ_{edge} and removal threshold λ (the Λ_{edge}=750 and λ=4000 are examined to be practical enough to carry out the experiment), two steps of activities can kill the node: (1) When node exists for a long time without winning any new input pattern, once Λ_{edge} is reached, all connected edges die and are eliminated. (2) As the accumulated nodes increase to reach threshold λ, the node without bonded edge is eliminated from the network.

D. Incremental Learning (Training) and Recall (Testing)

The system contains a database with prior knowledge, which can be incrementally trained in real-time while the AR.Drone is in the air. A six-dimensional vector of training data is received at each time-step as a state-command paired set,$\{p_x, cmd_x\}$, $\{p_y, cmd_y\}$ and $\{p_z, cmd_z\}$, obtained from the estimated position of PTAM, with an associated AR.Drone control command that executed from a USB game-pad controller manually operated by an operator. The training and testing phase of the proposed SOIAM-based system is detailed in Fig. 3.

1) Data-Pool Structure: The trained SOIAM network maintains a data pool, that stores the associative pairs of the state-command sets in a time series format. Left-hand side of Fig. 3 shows the designed structure of the data pool, where SOIAM XY and SOIAM Z are the coarse learning sets, which are trained SOIAM network with untuned control command, that gives a full range of command values (ranging from 0 to 1, higher value provides more aggressive fly experience to the AR.Drone), this provides generalized and dynamical manoeuvre of piloted control, whereas SOIAM XY' and SOIAM Z' hold an identical data structure; however these are fine learning set only trained with finely tuned command values(from 0 to 0.34). They are not involved in the cycle of incremental training. The advance of having this fine learning set is to improve the accuracy of control, while maintain the generalization of task performance. In particular, an associative data manager (ADM) is embedded to select which SOIAM network to be incrementally trained or recalled, based on the feedback error estimation given by system, as the error is less than 0.3m, fine learning set is selected, otherwise coarse learning set is selected.

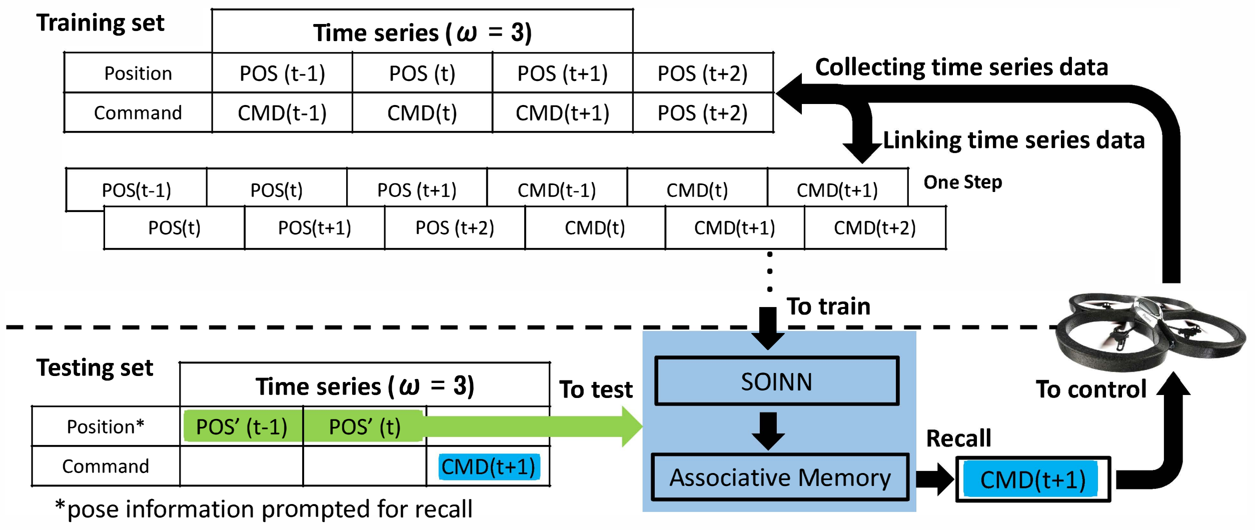


Fig. 4: Process of associative state-command pair dataset from the proposed self-organized incremental neural network associative memory, in both training and testing phase.

2) Time Dependent Learning and Recall: This study defines the time series data as successive postures measured over time, and assumes that each measured posture links with a corresponding control command. Fig. 4 details the composition of time series training and test data, where the time series is represented by a sequence of $t-1, t, t+1$ and $t+2...etc$, and so on. To efficiently predict from the previously learned time-series data, SOIAM is trained through employing a sliding window technique with shifting steps. This process is performed by the ADM, shown in Fig. 3. It helps to collect and link the time series training and testing data. The number of training steps at time t is given by $\frac{(s \times t)}{\omega}+1$, where s is the data sampling rate and ω indicates the time series length of associative pair. This study uses ω=3, a value fast enough to allow the system to respond to the motion of AR.Drone. To perform recall after sufficient steps of the learning cycle, during which the estimated position of AR.Drone continuously provided, the trained system is prompted to estimate the associated commands from the data pool. The performance of the system can be evaluated by logging the series of position changes.

E. System Implementation

SOIAM, the AM-based SOINN, is embedded in the proposed system. The entire system including SOIAM, was developed in C++ on a GNU linux platform, Fig. 5 generalize the system assembly workflow, in which each block is developed separately and then integrated using the open source Robot Operating System(ROS) [18], which provides well-structured interfaces for all blocks. The open sourced AR.Drone Application Programming Interfaces (API) of CVDrone (OpenCV plus AR.Drone, https://github.com/puku0x/cvdrone) are used as the primary control unit, in which OpenCV library is utilized for image handling and processing. The API provides basic control functionality, enabling communication between AR.Drone and notebook for transferring sensor data and sending commands.

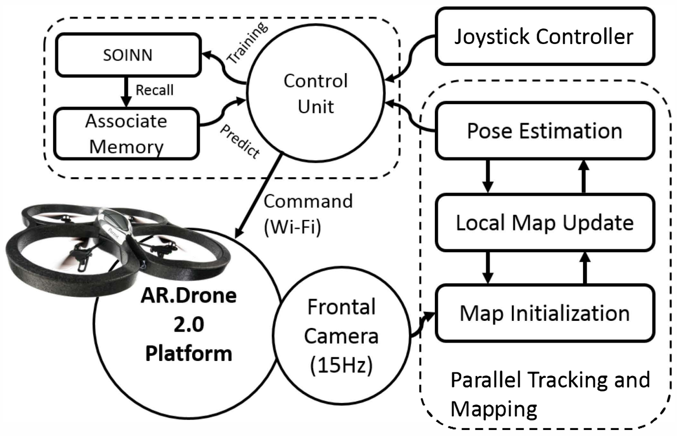


Fig. 5: Topological implementation of proposed AR.Drone system.

1) Visual self-localization and mapping: The received image data are processed by OpenCV and PTAM to maintain a local map of the nearby environment; PTAM is modified to interface with the system for the localization of AR.Drone; the on-board camera is recalibrated to optimize the performance of PTAM, the maximum number of detected features is set to 700, which has been seen to be a suitable value in practice. A consistent global map is not enforced because it is not the focus of this study; however it is what we wish to build and combine in the future with the proposed algorithm. Drift measurements of posture estimation are overcome by locking and serializing the map to reserve identical posture accuracy for the same task. Hence the ground truth of the real world metric coordination can be kept identical for fair comparison of displacement of the AR.Drone.

2) X, Y and Z coordination: The information exchanged in this study involves estimated postures from camera and game-

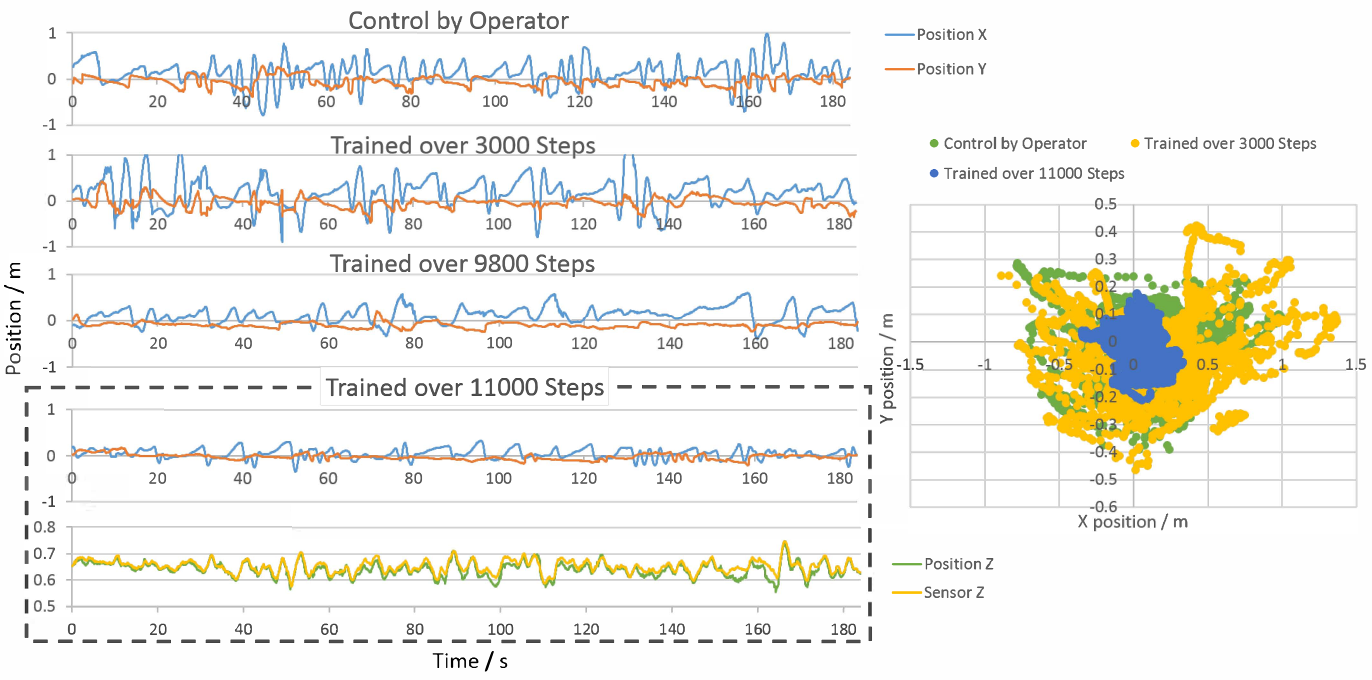
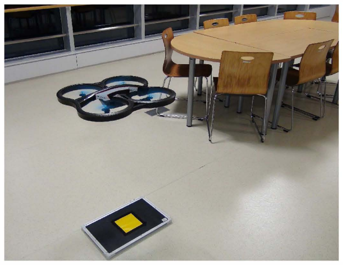


Fig. 6: The positional transition of commanded AR.Drone for autonomous hovering that measured along x-, y- and z-axis with different training iterations in stationary condition. The x and y plot on a scale of [1, -1] m are on left-hand side, where the sign represents direction of displacement from the hovering target, z plot shows hovering 0.64 m above ground.

pad commands, the state-command pair datasets that were applied to SOIAM are $\{p_x, p_y, p_z\}$ and $\{cmd_x, cmd_y, cmd_z\}$ respectively, where p_x, p_y, p_z are position vector along x, y, and z-axis in 3D space, as cmd_x, cmd_y, cmd_z are the corresponding velocity commands that control the AR.Drone's manoeuvre. Therefore, using the frontal camera as the pointing direction for forward manoeuvre, the positions along x- and y-axis are specified by the Cartesian coordinate system in yaw plane of the AR.drone, where the y-axis is defined by pitch that moves AR.drone forward and backward. The x-axis is defined by the roll that moves left and right. The z-axis is the elevation of the AR.Drone perpendicular to the yaw plane (x- and y-axis), which moves AR.Drone in the vertical direction. This study functions with two SOIAM networks to simplify the problem of dealing with 3D plane and to improve the efficiency of using the paired dataset. The x- and y-axis are studied together with associated pair being given to one SOIAM as $\{p_x, p_y\}$ and $\{cmd_x, cmd_y\}$, this yaw plane provide good flying conditions for learning because the AR.Drone is subject to serious drift during flight over this plane. The z-axis (or altitude) is learned separately, from experimental observation, the altitude control of AR.Drone being independent of the x- and y-axis. Hence, by simplifying the learning problem, the z-axis is treated individually by synthesizing a new set of SOIAM for this plane, with given associated pair of $\{p_z\}$ and $\{cmd_z\}$.

IV. EXPERIMENTS AND RESULTS

To evaluate the performance of the proposed system, fundamental studies in the flight mechanism of hovering are featured to train the system. In general a hovering task, the controller seeks dynamics on the basis of the position data collected from flight and adjust the dynamics to minimize the errors from a fixed position. The proposed SOIAM-based system is expected to estimate appropriate command outputs to control the AR.Drone. All experiments are conducted in an indoor of laboratory space: autonomous hovering control in stationary and interruptive environments, and the hovering results are evaluated in terms of precision and stability, where the precision shows how accurate the system can hover at same place over a period of time, and the stability accesses the average error of hovering over time for the system.


(a) Stationary hovering

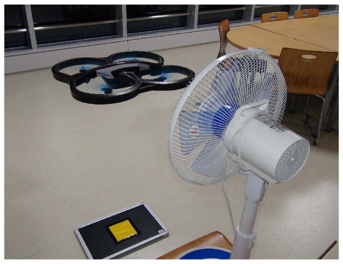

(b) Interruptive hovering

Fig. 7: Experimental environments for testing fly of AR.Drone. A yellow 5cm-by-5cm square marker that placed below the drone indicates the attempted hovering point.

TABLE I: Experimental Estimations on Precision and Stability of hovering

Experimental Condition		**On-board Sensors**		**Proposed SOIAM**		**Classical PID**	
		Precision	Stability	Precision	Stability	Precision	Stability
Stationary hovering	along x-axis	259.3	0.647	62.9	0.138	63.4	0.206
	along y-axis	249.8	0.541	50.7	0.114	54.5	0.196
Interruptive hovering	along x-axis	319.4	0.785	76.7	0.176	84.1	0.258
	along y-axis	298.4	0.622	54.2	0.115	72.4	0.266

A. *Stationary Hovering*

We first consider autonomous hovering in a stationary 3-D space, where stationary space is defined here as stable indoor environment without wind, Fig. 7 (a) shows the AR.Drone perform the test hovering fly in stationary environment, the error is primarily driven by the drift of AR.Drone, owing to its imprecisely tuned mechanics, such as propellers. Fig. 6 shows a 180 s testing results for different steps of training cycle over the x- and y- axis. This experiment provides results with 3000, 9800, and 11000 training cycle steps. The total training time is 796 s, resulting from the battery power constraints, which limited the flying time. For each stage of training cycle, the performance was studied for only 184s, with each data logging rate being 0.046 s; hence a total of 4000 data points per testing cycle are processed for evaluation. The results demonstrate the qualitative improvements in behaviour in the hovering precision (reduction of drifting error) over iteratively training of hovering manoeuvres. At the beginning of few training trials, the AR.Drone sometimes performs aggressive manoeuvres owing to the limited ability of the human pilot, leading the AR.Drone to pass the target position and cast away; however, during later iterative learning trials, the AR.Drone gradually learns how to retrieve from casting away from target point. After 11000 steps of training, the standard deviations of the measured x and y position are $\{\sigma_x, \sigma_y\}$ = {0.107, 0.06}m.

Along z-axis, The AR.Drone is controlled to hover at approximately 0.64 m above the ground. Similarly, Fig. 6 shows the testing results last 184 s after 11000 steps of training, with a standard deviation of σ_z = 0.026 m. Additionally, it shows the given state estimation comparison between measured height from the on-board Altitude Sensor and PTAM estimation of z position.

B. *Interruptive Hovering*

In this experiment, an interruptive environment is provided with artificial wind driven by an electrical fan, Fig. 7 (b) shows the AR.Drone performs hovering, flying in the interruptive environment. In this case, the error becomes more complex than for stationary hovering not only due to the mechanical problem but also the external wind excitation. For the experimental set-up, the fan is placed roughly 10 cm away from the right-hand-side of the AR.Drone, a single direction of continuous wind (approximated wind speed of 170 m/min) along the x-axis is assumed to perform the test and evaluation. On the basis of our observation, the pattern of the associative position input and its corresponding command output is stable across the stationary environment indicating that the previously trained system in stationary condition alone is unlikely to have good performance due to the absence of dynamics. Hence the sampling is not enough for performance robustness; the solution is to adapt the system to this new environment through on-line incremental learning, and the system is additionally trained to perform hovering in this interruptive environment. TABLE I concludes the quantitative results of the estimated precision and stability, on both stationary and interruptive environments. The evaluation is focused on the trajectories along the x- and y-axis respectively. Given a single direction wind along the x-axis of the AR.Drone, a continuous changing position causes the wind to attack at different angles, therefore mainly affecting the control on x- and y-axis. The test flying data are analysed along x- and y-axis separately, recorded at 180-s intervals and logged with a sampling rate of 21.7 Hz.

1) Precision: Hovering precision is estimated with the integration value for the trajectory from the hovering point(zero point). The integration value statistically has a unit, Absement, which is mathematically measured in ms (metre second), as the smaller value estimates more precise trajectory control of hovering. In general, a similar pattern can be observed across the x- and y-axis, and as expected, the hovering trajectories for control methods in TABLE I are less precise when in an interruptive environment, because the integration values ascend; however, for both environments, the integration value of hovering control, which relies on the on-board sensors of AR.drone (without learning control), is greater than the integration value, which is controlled by the proposed SOIAM-based system (with learning control). Therefore, SOIAM-based system is shown to improve and advance the control precision of the system. Additionally, referencing the SOIAM-based system to a classical approach of PID control (Proportional-Integral-Derivative Control), which assumed the PID parameters are finely tuned for the hovering test during stationary hover, the precision of SOIAM-based system stays closely to the PID control, whereas in interruptive hover, the SOIAM-based system shows more precise performance by providing a smaller integration value than that provided by PID control. This indicates adaptation is required to cope with dynamical change in the hovering environment.

2) Stability: Hovering control maintains the AR.Drone near a targeted position; hence, the system must not only to flexibly adapt the error but also to keep the error as small as possible. The square-rooted variation of the averaged integration value over periods of time gives the stability estimation, which is attempting to identify the volatility of the hovering trajectories. The averaged integration value is obtained by dividing the entire trajectory into small sections of sub-

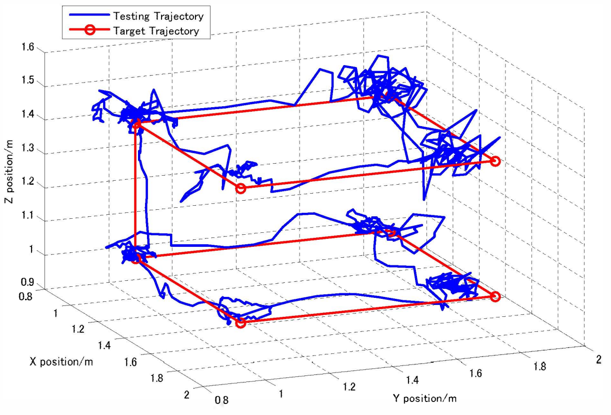


Fig. 8: Studied trajectories that consists of two 0.8 × 0.8-m squares in altitudes of 1.02 m and 1.42 m respectively: the trained SOIAM system is tested by giving four hovering way-points for each square.

trajectory. During this evaluation, the trajectory data is divided into 3-s sub-trajectory, with a total of 60 sub-trajectories. By working out the integration value of each sub-trajectory, the square-rooted variation can be estimated by calculating the standard deviation from the averaged integration value; hence, a stable flying trajectory features a small value of estimated result. Considering the results shown in TABLE I, SOIAM-based control provides comparatively lower stability value than the on-board sensor and PID control across both experimental environments. This validates a reasonable stability across the 180s flying test in both environments.

C. Trajectory Control

We now consider a study of autonomous navigation, simplifying the problem by generating a rectangular trajectory in a 3-D space. This is performed using the previously trained autonomous hovering system to command the AR.Drone, and by giving a sequence of way-points representing the different hovering targets for AR.Drone to approach. Fig. 8 presents the qualitatively resulting trajectories of a double tier rectangular trajectory(in blue) using 8 way-points to fly the planned trajectory, the trajectory is reconstructed from the estimated position given by PTAM.

V. CONCLUSION AND OUTLOOK

In this study, we introduce an SOIAM-base control system to allow generalized and precise autonomous control of low-cost quadcoptor in unconstructed environments. Validating the system through experiments, we benchmark the robustness of the learning-based control system that minimized the drift error of AR.Drone, and detailed the performance of the system with respect to its incremental manner from noisy environments. Our model-free approach was demonstrated, in particular, for quadcoptor control estimation with given estimated posture states while attempting hovering tasks with minimal prerequisites, i.e., additional sensors, advanced knowledge of flight models. Although a visual system for self-localization was not considered in this work, we must address this problem in order to have more precise and globally consistent control freedom. We are also interested in enabling the quadcoptor to learn more complex manoeuvres.

ACKNOWLEDGMENT

This work was sponsored by the Japan Science and Technology Agency's CREST project.

REFERENCES

[1] P. Benavidez and M. Jamshidi, "Mobile robot navigation and target tracking system," in *System of Systems Engineering (SoSE), 2011 6th International Conference on*. IEEE, 2011, pp. 299–304.

[2] A. de Rengervé, S. Boucenna, P. Andry, and P. Gaussier, "Emergent imitative behavior on a robotic arm based on visuo-motor associative memories," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE, 2010, pp. 1754–1759.

[3] R. Siegwart, S. Bouabdallah *et al.*, "Design and control of quadrotors with application to autonomous flying," 2007.

[4] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*. IEEE, 2007, pp. 225–234.

[5] M. Achtelik, S. Weiss, and R. Siegwart, "Onboard imu and monocular vision based control for mavs in unknown in-and outdoor environments," in *Robotics and automation (ICRA), 2011 IEEE international conference on*. IEEE, 2011, pp. 3056–3063.

[6] J. Engel, J. Sturm, and D. Cremers, "Scale-aware navigation of a low-cost quadrocopter with a monocular camera," *Robotics and Autonomous Systems*, 2014.

[7] P. Pounds, R. Mahony, J. Gresham, P. Corke, and J. M. Roberts, "Towards dynamically-favourable quad-rotor aerial robots," in *Proceedings of the 2004 Australasian Conference on Robotics & Automation*. Australian Robotics & Automation Association, 2004.

[8] G. M. Hoffmann, H. Huang, S. L. Waslander, and C. J. Tomlin, "Precision flight control for a multi-vehicle quadrotor helicopter testbed," *Control engineering practice*, vol. 19, no. 9, pp. 1023–1036, 2011.

[9] A. J. Calise, N. Hovakimyan, and M. Idan, "Adaptive output feedback control of nonlinear systems using neural networks," *Automatica*, vol. 37, no. 8, pp. 1201–1211, 2001.

[10] G. V. Chowdhary and E. N. Johnson, "Theory and flight-test validation of a concurrent-learning adaptive controller," *Journal of Guidance, Control, and Dynamics*, vol. 34, no. 2, pp. 592–607, 2011.

[11] D. Nodland, H. Zargarzadeh, and S. Jagannathan, "Neural network-based optimal adaptive output feedback control of a helicopter uav," *Neural Networks and Learning Systems, IEEE Transactions on*, vol. 24, no. 7, pp. 1061–1073, 2013.

[12] M. Ö. Efe, "Neural network assisted computationally simple pi d control of a quadrotor uav," *Industrial Informatics, IEEE Transactions on*, vol. 7, no. 2, pp. 354–361, 2011.

[13] P. Abbeel, A. Coates, and A. Y. Ng, "Autonomous helicopter aerobatics through apprenticeship learning," *The International Journal of Robotics Research*, vol. 29, no. 13, pp. 1608–1639, 2010.

[14] L. D. Minh and C. Ha, "Modeling and control of quadrotor mav using vision-based measurement," in *Strategic Technology (IFOST), 2010 International Forum on*. IEEE, 2010, pp. 70–75.

[15] G. V. Raffo, M. G. Ortega, and F. R. Rubio, "Backstepping/nonlinear h control for path tracking of a quadrotor unmanned aerial vehicle," in *American Control Conference, 2008*. IEEE, 2008, pp. 3356–3361.

[16] A. Sudo, A. Sato, and O. Hasegawa, "Associative memory for online learning in noisy environments using self-organizing incremental neural network," *Neural Networks, IEEE Transactions on*, vol. 20, no. 6, pp. 964–972, 2009.

[17] S. Furao and O. Hasegawa, "An incremental network for on-line unsupervised classification and topology learning," *Neural Networks*, vol. 19, no. 1, pp. 90–106, 2006.

[18] M. Quigley, J. Faust, T. Foote, and J. Leibs, "Ros: an open-source robot operating system."