A Generic Conceptual Framework for Autonomous Navigation with a case study on Wildlife Detection using UAV

Gopalakrishnan Saisubramaniam (A0178249N), Goh Chung Tat Kenrick (A0080891Y)
Master of Technology in Knowledge Engineering, Institute of Systems Science,
National University of Singapore

***Abstract*—The growing interest in autonomous navigation is seen through the emergence of several innovations within the last decade. From classical rule-based approaches to convolutional neural networks to reinforcement learning, the method adopted to make an agent learn to act on its own has varied based on the complexity of the task at hand and the environment it must work in. Reinforcement Learning in particular, has seen a resurgence in popularity with the successful implementation of it on classical Atari game. Based on learning through experience/ trial and error which is similar to a normal human learning process, it allows an agent having imperfect information to understand the inputs given to it through the environment and determine the best possible action that should be done at that point of time. The more number of times the agent plays, the more experience it gains and becomes better at the game. Adopting the concept for real world tasks is possible, however not in a strict sense. Time and resource constraints discourage training an agent from scratch. Hence simulations are often adopted to bootstrap the training process. We propose a conceptual framework general enough to allow the agent to learn faster by defining a boundary on the search space and rewarding it to achieve maximal coverage of the objective set for the agent. As a case study, we simulate an autonomous drone in an African environment setting to fly over a given set of source and destination points and achieve the set objective of detecting wildlife from a fixed height and distance while minimizing/avoiding collision with the artifacts of the environment.**

***Index Terms*— Autonomous navigation, reinforcement learning, computer vision, deep learning, artificial intelligence**

I. Introduction

The motive behind autonomous movement is to enable a machine powered by artificial intelligence to navigate around on its own without external supervision, relying on its own physical and intellectual capabilities. Technical advancements have been made since the last three decades, starting with rule-based approaches, [12] Kalman filters [13], convolutional neural networks [14] and more recently using reinforcement learning techniques. [15]

.

Reinforcement learning has been around since 1998 [1] and it has been receiving increasingly amount of attention in the past 4 years with the advancement in computation power. Reinforcement learning is a part of the multi-disciplinary field namely psychology and neuroscience as the agent attempts to learn through experience and then devise new strategies to improve on its action in the environment. Computer games allows the agent to work in a virtual environment where all the variables can be controlled by the game engine. In addition, considering from the viewpoint of a computer game, there is a comparative benchmark available easily in all games. By pitting the artificial agent against Real Players or its own, we can determine if the model is working well or not. Some examples of successful Deep Reinforcement Learning done is Alpha GO where AlphaGO [3], the artificial agent is able to defeat the Grand Master using Reinforcement Learning. In 2017, Reinforcement Learning has been expanded to more complex games such as Starcraft [4]. In Starcraft, it is a unique concept as it is a multiple agent problem with vastly large search space and imperfect information. Although the project is not able to succeed in creating an artificial agent to replicate a pro-player's level of expertise, it is deemed as a prelude to the next major achievement. In 2018, OpenAI Five has successfully create 5 artificial agents to co-operate with each other and compete with a human team of 5. It is able to defeat a semi-pro team which is a huge achievement, considering the imperfect information as well as the vast search space [5].

II. Related work

In 2015 Deep Q-Learning (DQN) [2] was published which is able to handle a variety of complex tasks normally requiring human level control. In particular, DQN is able to learn from its own experience in all the classic Atari 2600 games and achieve a comparable result compared to a professional human game tester. The work bridges the divide between high-dimensional sensory inputs and actions leading to the first artificial agent capable of learning to excel at a diverse array of challenging task.

III. Terminology

A. ***Autonomous Agent*** – a self-sufficient system independent of an external server/aid that has the capabilities to perceive sensory and visual input at state *s*, process and compute inference on the fly and take one of the predefined actions *a* and attempt to maximize its reward *r* during the course of its operation to achieve the defined objective *o* set for the episode *e*.

B. ***State*** – current location of the agent in the search space when receiving its input.

C. ***Action*** – the set of all possible moves performed by agent, predefined before the start of the algorithm.

D. ***Policy*** – Policy refers to the strategy that the agent will employ to determine the next action based on current state. It aims to choose the next action which provides the best rewards.

E. ***Reward*** – Reward refers to the short-term output given by the environment based on the last action. The agent learns to maximize the cumulative future reward which is denoted in the equation below; can be positive (good feedback) or negative(penalty)

$$R_t = r_{t+1} + r_{t+2} + r_{t+3} \dots = \sum_{k=0} r_{t+k+1}$$

F. ***Objective*** – the business goal to be achieved a.k.a. the purpose of the agent to exist.

G. ***Episode*** – the interval from the source to the destination; comprises of several states and actions between the start and end.

H. ***Value*** – Refers to the expected long-term return which differs from the short-term reward R. It is based on the state that the agent finds itself while choosing different actions during the training progress. It is noted that while the state may be the same for different circumstances, Different actions may take for different iterations in order to arrive at that state. As such, the reward R will differ slightly and have some randomness in it. As such, the expectation takes into account all this randomness and comes out with a mean value.

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left[R_t | s_t = s\right]$$

I. ***Q-value*** – Refers to the long-term return of the current state when taking action under a policy. This is similar to the variable Value. However, it tells us the expected Cumulative Reward given an action a. We aim to maximize this Q-value which gives us an indicator that the agent is being more proficient at performing the task

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}\left[R_t | s_t = s, a_t = a\right]$$

A. ***Environment*** – The environment represents the virtual world where the agent performs an action. The environment will take the actions and returns 2 outputs, namely the agent's reward as well as the current state in the environment.

Overall, the diagram below shows the architecture on how the whole algorithm works and how the agent learns from its action given the rewards and states which are output by the environment.

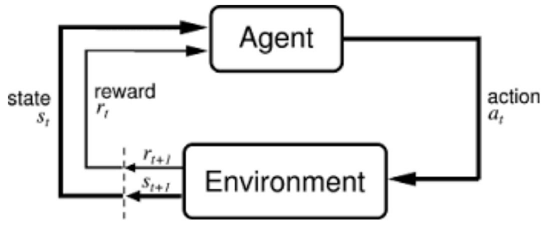


Figure 1 Agent Environment Interaction

IV. Proposed Conceptual Framework

A. Overview

We propose a framework that takes into consideration the navigation, objective satisfaction and route tracing aspects of an agent.

B. Business Objectives

Although not an exhaustive list, some of these objectives can be covered in an autonomous navigation project:

i. Cost – the cost assigned to travel a certain distance. The total number of steps taken by agent to reach the destination. Steps that are further away have either a lower positive or a higher negative cost and viz with the steps closer to the destination.

ii. Constraints – A set of constraints must be enforced on the agent w.r.t the environment designated to it. The viewpoint is to effectively reduce the search space using these constraints.

iii. Target – There must be a target for the agent to achieve, using which the agent reinforces its belief states and progresses towards the objective.

C. Navigation Algorithm

The algorithm suggested for navigation is DQN reinforcement learning proposed in the initial paper [2]. At each time step, it chooses an action a_i from a set of possible actions that has been pre-defined at the start. From the action, the scoring system will be updated, and the agent will receive a reward based on the change in to the scoring system.

A pseudocode of the DQN algorithm adopted from the paper is provided in Figure 2.

```
Initialize replay memory $D$ to capacity $N$
Initialize action-value function $Q$ with random weights $\theta$
Initialize target action-value function $\hat{Q}$ with weights $\theta^- = \theta$
For episode $= 1, M$ do
  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$
  For $t = 1$,T do
    With probability $\varepsilon$ select a random action $a_t$
    otherwise select $a_t = \text{argmax}_a Q(\phi(s_t),a;\theta)$
    Execute action $a_t$ in emulator and observe reward $r_t$ and image $x_{t+1}$
    Set $s_{t+1} = s_t,a_t,x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t,a_t,r_t,\phi_{t+1})$ in $D$
    Sample random minibatch of transitions $(\phi_j,a_j,r_j,\phi_{j+1})$ from $D$
    Set $y_j = \begin{cases} r_j & \text{if episode terminates at step } j+1 \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1},a';\theta^-) & \text{otherwise} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j,a_j;\theta))^2$ with respect to the
    network parameters $\theta$
    Every $C$ steps reset $\hat{Q} = Q$
  End For
End For
```

Figure 2 DQN Algorithm Pseudocode

Exploration – To find out all the different possible states in the solution space. Allows the agent to wander on its own to explore the space. After an action, compute the rewards and update the results into an action value function called *Q(s,a)* After 10,000 epochs of exploration, sample a mini-batch to train the Deep Neural Network Model to the update Q value.

Exploitation – Utilize gained experience in order to obtain the optimal solution at the situation and attempts to maximize the rewards over time, a.k.a. Greedy Approach. The downside of greedy approach is that it provides little or no exploratory potential. To solve this issue, we take a random action occasionally while choosing the optimal route.

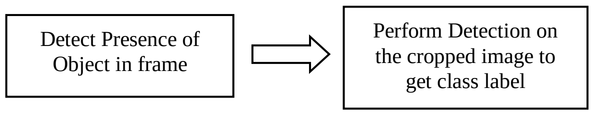
D. *Rewards*

Rewards are almost exclusive to each case, but most applicable ones are:

i. Distance to goal – More rewards for getting closer to the goal, penalize when the agent moves further away.
ii. Object detected – Rewards are given whenever the agent finds something interesting or performs an interactive action that is relevant to the set objective.
iii. Discovery of solution space – Some business use cases may require more exploration on new areas; in such cases the identification of new and potentially interesting areas can be rewarded.

E. *Computer Vision based Objectives*

Most of the objectives of autonomous agents require the use of computer vision techniques. In cases such as monitoring using object detection, approaches such as two-step approach as suggested in [10] can be adopted. First step can be to identify the presence of the object in the frame, and crop the image using the bounding box coordinates. After enlarging the image, a better prediction can be made to identify the exact object. This approach is suggested due to the possibility that the agent may not be in enough proximity to the object of interest, in such cases the two-step approach gives better results.


Detect Presence of Object in frame
Perform Detection on the cropped image to get class label

F. *Video Feed Transfer in Near Real Time*

To improve the speed of transfer of the video frames from the agent to a server (for future analysis) individual image frames are compressed using ZStd algorithm from Facebook [11] and serialized and transferred across the network. The process is reversed for restoration of the original image. A typical jpg image of 900KB can be reduced to 20KB and transferred as shown in Figure 3.

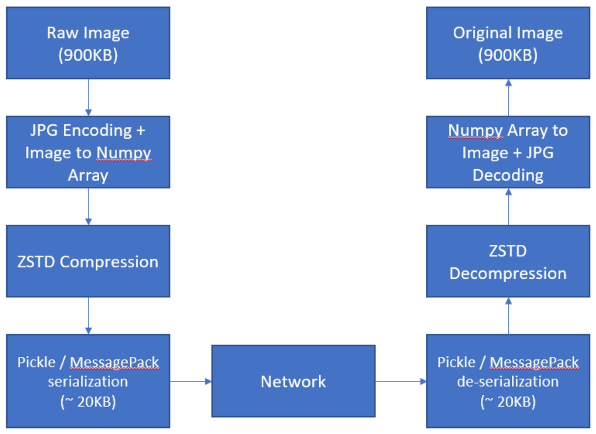

Raw Image (900KB)
JPG Encoding + Image to Numpy Array
ZSTD Compression
Pickle / MessagePack serialization (~ 20KB)
Network
Pickle / MessagePack de-serialization (~ 20KB)
ZSTD Decompression
Numpy Array to Image + JPG Decoding
Original Image (900KB)

Figure 3 Transferring image data from agent to server
The encoding-decoding process at both ends

V. Case Study – Monitoring Wildlife in the African Terrain using an Autonomous UAV

As a case study, we focus on monitoring wildlife using a UAV. The simulation framework used is Microsoft's AirSim [6] which is an open source simulator for autonomous vehicles built on Unreal Engine. There are two options to select –i. quadcopters and ii. cars. African wildlife [9] was chosen as the environmental setting as the simulation focuses on wild animals. The objective is to detect as many animals as possible while navigating between two fixed points. [7][8]. Here we assume that the animals under observation do not migrate to far away distances (unlike birds) and therefore the environment is relatively static.

The following sub-sections describe the case specific solution approach, training and results of DQN for navigation and visualizing the route the drone takes during its flight.

Agent – The agent will be a drone (quadcopter) which has been pre-defined in the AirSim. It is a standard 4 rotor off the shelf drone. In Unreal Engine, 4 cameras have been attached to it showcasing the different viewpoints of the camera. This is similar to an actual drone.

Functionalities of the drone includes

1) *4 Cameras (Left Center, Right Center, Center, Back and Bottom)*
2) *GPS Co-ordinates provided by Unreal Engine.*
3) *Velocity of drone provided by Unreal Engine*

Action – There will be 6 different actions available to the drone at each state. Since the drone moves in 3d space, we take into consider the three axes – x, y and z. The agent will choose one of the action available from this set which aims to maximize the Q-value. Each action will involve movement by a constant velocity factor.

1,2: Move in the positive, negative x-axis
3,4: Move in the positive, negative y-axis
5,6: Move in the positive, negative z-axis

Policy – The policy is influenced by the bottom camera, since the detection of animals results in more rewards.

Reward – The reward here is +100 whenever an animal is spotted by the bottom camera and detection by the object detection algorithm.

A. *Solution Approach*

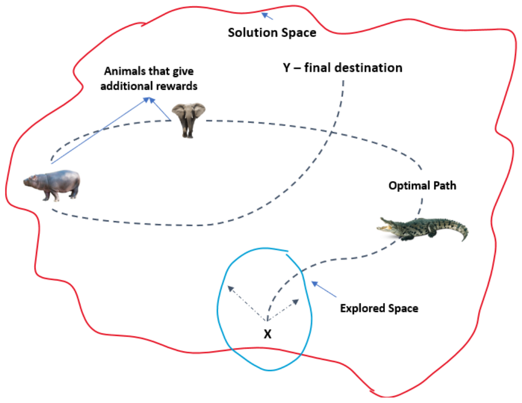


Figure 4 Finding optimal path to maximize count of animals

The solution for this case study is to navigate the drone from point X (source) to Y (destination) and find the optimal route that covers maximum animals along the path.

B. *Training with Deep Q-Learning*

The drone was trained on the environment for several episodes for days, the overall training was ~1.1M iterations. The graphs below provide the training metrics across iterations. Do note that the training has not converged and

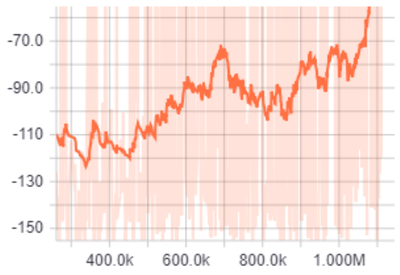


Figure 5 Mean Action-Value (Q) vs Training Iterations

Initial action-value (Q) is very low due to random movements by the drone, a.k.a. the *exploration* phase. Eventually, the drone starts improving (negative value decreases) as it starts *exploiting* the familiar route.

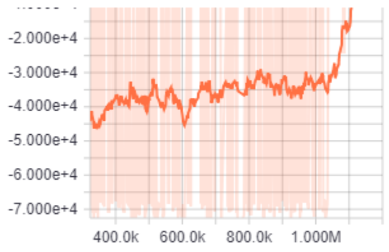


Figure 6 Rewards per Training Iteration

The rewards are negative because of high penalty assigned for straying away from the path. As the iterations increase, the rewards improve as shown in Figure 5.

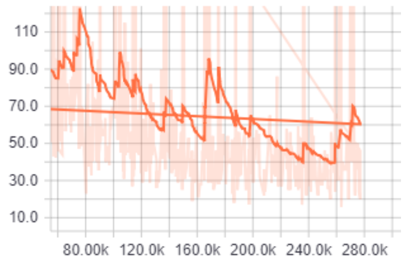


Figure 7 Average Loss per minibatch

Figure 6 shows the average loss decreasing with every iteration, computed in terms of minibatch size of 4.

C. *Objective – Detection of wildlife using Computer Vision*

Using the two-step approach similar to [10], the animals are first detected on the basis of an object captured is considered as present, the bounding boxes are cropped, and another detection algorithm identifies the species of animal from the enlarged image.

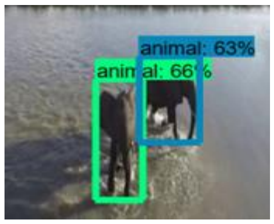


Figure 8 Object detection i. Presence of animal

⇩

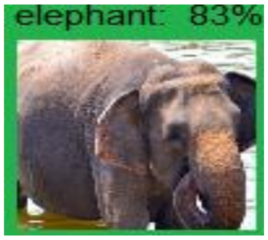


Figure 9 Object Detection ii. identification of specific animal

D. Oversight of the route taken

This sub-section describes the visualization of real-time route navigation from the drone as it traverses from start (star) to the destination (diamond).

After 1.1M iterations, the screenshot below shows the path traversed with an allowable distance deviation of 30m from the destination. It implies that the drone can land at any coordinate which is at most 30m away from the described destination.

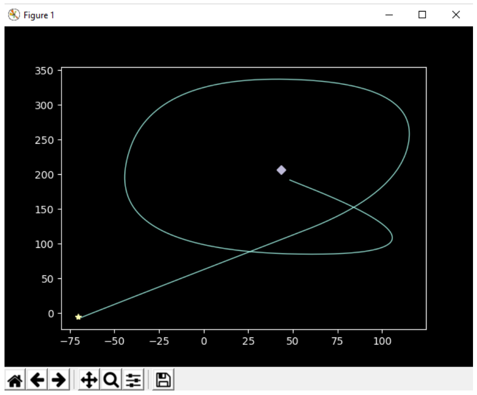


Figure 10 Route visualization with acceptable distance deviation of 30m from destination

It is observed that when decreasing this deviation distance, the drone takes a longer time and covers wider route before landing. The screenshot below shows the route when the deviation distance is 5m.

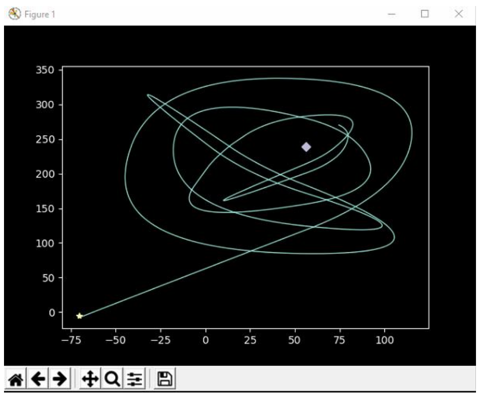
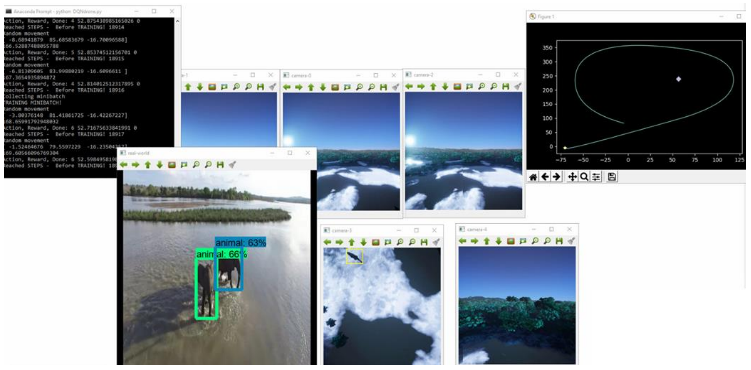


Figure 11 Route visualization with acceptable distance deviation of 5m from destination

Therefore, based on the allowable deviation distance, we can fine tune the training time, making it converge sooner or later. The overall distance traveled also depends on the agent's freedom of movement across dimensions – a drone has a lot more factors to consider due to its movement in the z-axis.



```
ction, Reward, Done: 4 52.875438985165026 0
eached STEPS -  Before TRAINING! 18914
andom movement
 -8.68941879  85.68583679 -16.70096588]
66.52887488055788
ction, Reward, Done: 5 52.85374512156701 0
eached STEPS -  Before TRAINING! 18915
andom movement
 -6.81309605  83.99880219 -16.6096611 ]
67.3654935894872
ction, Reward, Done: 4 52.814012512317895 0
eached STEPS -  Before TRAINING! 18916
ollecting minibatch
RAINING MINIBATCH!
andom movement
 -3.80376148  81.41861725 -16.42267227]
68.65991792948032
ction, Reward, Done: 6 52.71675633841991 0
eached STEPS -  Before TRAINING! 18917
andom movement
 -1.52464676  79.5597229  -16.23504
69.60566096769304
ction, Reward, Done: 6 52.598495819
eached STEPS -  Before TRAINING! 18
```

Figure 12 Overall framework components

VI. Challenges and Future work

One of the existing challenges with reinforcement learning is the sheer amount of time it takes for training a decent model. Despite the environment space being bounded to restrict infinite possibilities, the large search space multiplied by the dimensions accessible by the agent makes the problem on an exponential scale. A future work can be to parallelly train multiple agents locally and aggregate the best features globally every few iterations (adopting a genetic algorithm approach) so that the solution converges faster.

Another work that can be attempted is to transfer learn the agent's model across environments and even attempt cross agent transfer and observe the results.

VII. Conclusion

A generic conceptual framework for autonomous navigation was developed in an attempt to train an agent to move across in an environment and satisfy specified objectives by incentivizing with rewards based on performance. A case study was chosen – wildlife detection using drone in African terrain to showcase the implementation of the framework.

Acknowledgment

We would like to thank the lecturers of Institute of Systems Science, namely Dr. Matthew Chua for his continual support in the drone project, Dr Tian Jing for his wonderful advice in image processing, Dr Fun Wey on his insights for the drone architecture and possible features to be implemented in the drone. Last but the least, we would like to thank the Institute for the opportunity given to us to pursue this research topic.

References

[1] Sutton, R. & Barto, A. Reinforcement Learning: An Introduction (MIT Press, 1998)

[2] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G & Petersen, S. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529.

[3] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G & Dieleman, S. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, *529*(7587), 484.

[4] Vinyals, O., Ewalds, T., Bartunov, S., Georgiev, P., Vezhnevets, A. S., Yeo, M. & Quan, J. (2017). Starcraft II: A new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782*.

[5] OPENAI Release- Learning to Defeat Amateur players in Dota2 https://blog.openai.com/openai-five/

[6] AirSim: Open-Source Simulator for autonomous vehicles built on Unreal Engine https://github.com/Microsoft/AirSim

[7] Bondi, E., Fang, F., Hamilton, M., Kar, D., Dmello, D., Choi, J. & Nevatia, R. (2018). Spot poachers in action: Augmenting conservation drones with automatic detection in near real time. IAAI.

[8] Bondi, E., Kapoor, A., Dey, D., Piavis, J., Shah, S., Hannaford, R. & Tambe, M. (2018). Near Real-Time Detection of Poachers from Drones in AirSim. In *IJCAI* (pp. 5814-5816).

[9] AirSim-W: A Simulation Environment for wildlife conservation with UAVs

[10] Xulei Yang et al (2018) 'Deep Learning for Practical Image Recognition: Case Study on Kaggle Competitions' in KDD18

[11] Zstandard real time compression algorithm https://facebook.github.io/zstd/

[12] N. Zimmerman et al., (2004) Implementing a Rule-based System to Represent Decision Criteria for On-Road Autonomous Navigation, AAAI

[13] A. Stevens ; M. Stevens ; H. Durrant-Whyte (1995) "OXNAV": reliable autonomous navigation

[14] Yiyou Sun ; Tonghua Su ; Zhiying Tu (2017) Faster R-CNN based autonomous navigation for vehicles in warehouse

[15] Chao Wang ; Jian Wang ; Xudong Zhang ; Xiao Zhang (2017) Autonomous navigation of UAV in large-scale unknown complex environment with deep reinforcement learning