Proposal for deep learning obstacle avoidance

1. I think this is an innovative project. The idea of combining the latest development in machine learning industry (deep reinforced learning used in AlphaGo) with marine vessels attracts me. Traditionally, the motion planning for autonomous vehicles is done via either a rule-based model or an optimization-based model, which involve primarily analytical approaches that can give a definite solution by solving well-defined equations. For the application of obstacle avoidance, analytical approaches are deemed too complicated, and requiring too much computational power, due to that their success relies heavily on perfect modelling of obstacles as well as dynamics of marine vessels. This project, however, employs a data driven approach that is more concise and generic, one that could be easily modified to suit all kinds of obstacle conditions. Therefore I understand both the magnitude and potential of this project.
There is no doubt that artificial intelligence incorporated into vehicle automation is the wave of the future, and I want to be part of it. I want to, always, stand at the forefront of technologies.

2. The first step is to determine objectives. Through given information, I can assume that this is a navigation task consisting of two main objectives: one is to avoid obstacles during the course of motion, the other is to reach a given destination (objs could be wrong). Next is to use dynamic analysis to establish mathematical models for the vessel, i.e. assume three components of motion for vessel: surge, sway and yaw, identify the position and velocity vectors. Since vessel is also underactuated, I think it is needed to consider the control input vector, as it is possibly a 3DOF system but with 2 actuators i.e. the sway of vessel cannot be controlled, control vector takes only surge component and yaw component.

After initial modelling, there should theoretically be a design about data collection. But since I was told that simulated data would be provided, thus this step can be neglected. From my understanding, the data collection unit is responsible for observing current state of vessel, including position and velocity, detection of obstacles within sensor's proximity and the controller behaviour from the last step time. Observation should be made at certain time intervals. The whole observation data should contain many individual elements that aggregate as time elapses.

Next, to receive and process such data, an artificial neural network is needed. There are many types of network architecture i.e. Long short-term memory (LSTM) network, Convolution neural network (CNN) etc. By reviewing each network's structure and advantages, thus deciding its suitability for this obstacle avoidance application, I will select the best network that allows efficient and rapid response. It is also interesting to develop the structures of network, i.e. for CNN, decide how many convolutional layers and hidden layers are needed to achieve a desired nonlinear output. The next step is to develop an activation function for neurons. Again I will review different activation functions i.e. sigmoid function, tanh function and rectified linear units (Relu). It is also needed to investigate the relationship between number of layers in a neural network and the activation function employed. I propose series of experimentations to be done to determine the final network design.

Neural network maps the observation information into characteristic state vectors. The next step is to develop a decision-making algorithm. This should be a reinforced learning approach that can take on the characteristic state data and arrive at an optimal control strategy. I think I will use framework in fig.1 and fig.2 as the principle underpins my algorithm. Base on Markov decision process (MDP), State S_t is the characteristic state at a time step t, action a_t causes a transition into a new state.

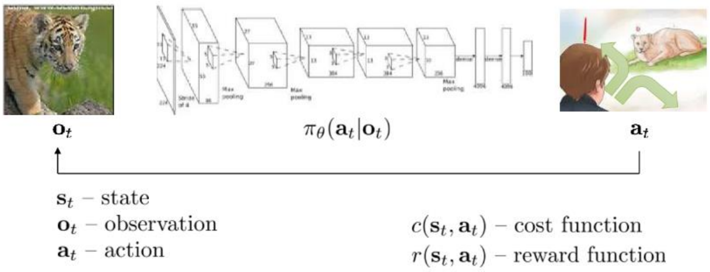


Fig.1[1]. From state to action to reward

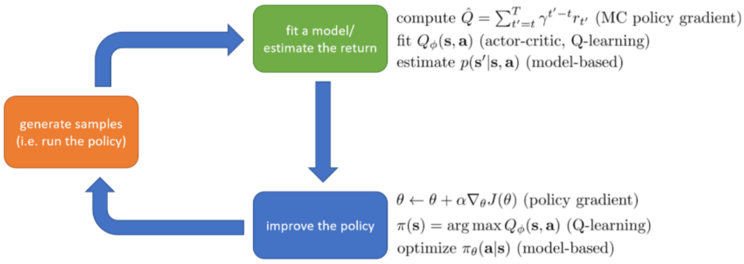


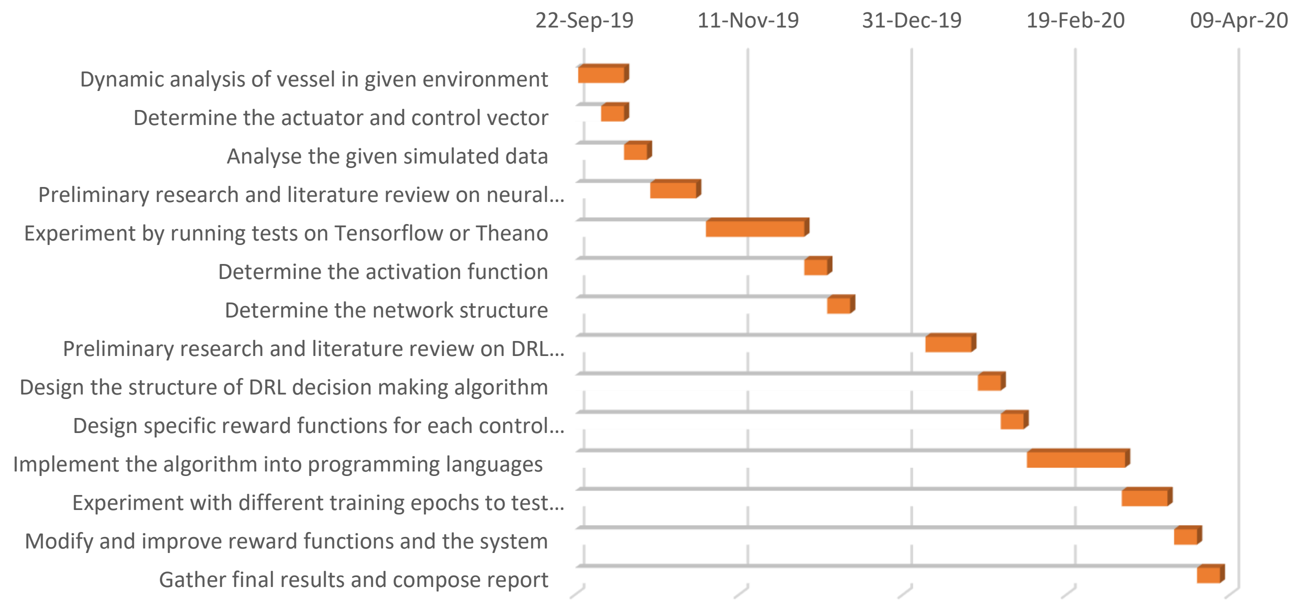
Fig.2[2]. Structure of RL algorithm

During each transition, reward $r(s_t,a_t)$ is received. A state-action function Q(s,a) is used to predict future rewards. Based on Q values, policy π is optimized, thus improving the performance. The reward function is vital to this DRL algorithm, and obstacle avoidance requires successive control inputs in different component, thus I will have to consider specific rewards for specific state and action. I think I will design reward function for destination approaching (as one of the main objectives of the project). I propose this reward function could be realised by distance control, which assigns negative values to the position of vessel, meaning the reward function only exert less punishment when the vessel is closing in to the target position. For the obstacle avoidance objective, I propose a radius comparison reward function, where it punishes when the actual distance (radius) to obstacle is within the safe radius range. Each reward term should be combined.

It is also necessary to carry out training process to optimize learning performance. I propose experimentation with training epochs that increases incrementally. This way we can deduce from experimental data that whether reward functions are effective i.e. how many epochs would the vessel take to find the optimal path under a specific reward function. Finally, the implementation of the algorithm, I propose, can be done in Matlab, Python, or C languages. I personally incline towards C, as it will also be used in my other projects.

3. I have full confidence in taking on this project. I have very good background knowledge of vehicle automation, especially analytical and data driven approaches, as last year I attended a summer course on auto-driving technology in Peking University, taught be Baidu Apollo researchers. For completion of that course, I also did a report on vehicle localization and route planning. During the same period of time, I also attended some Python lectures as a voluntary student in PKU. Just last term, I attended some night classes on html scripting, taught by student organizations of UCL CS department. Thus I used html and css to write my homepage. The endeavour to improve my programming skills never ends. I have finished certificated Lynda online courses on Matlab, Matlab Simulink, Python, and C languages. This term, I have made significant progress on self-learning C languages. My latest connected system project coded in C languages is a testament to that (see fig.5 at the end of proposal).
Indeed, I may not have every piece of knowledge needed, but I pledge every effort in learning to make up this gap of knowledge. For that effort, I have started a Coursera online course on machine learning, taught by Andrew Ng. (This course is due to finish at 15th April this year.)
My greatest capability is the capacity to learn. Last year for my first year in UCL MechEng, I averaged 80% for 8 modules. I have knowledge across different disciplines thanks to my A-level studies. Two years ago I finished my A-level degree with 4A* and an A, in 5 various disciplines: Physics, Chemistry, Maths, Further Maths and Biology. Today the artificial neural network used for this project reminds me of the neurons and synapses mechanism in human brain that I learned in Biology. My learning capacity is unlimited and I am willing to learn more.

4.



	Gather final results and compose report	Modify and improve reward functions and the system	Experiment with different training epochs to test system performance	Implement the algorithm into programming languages	Design specific reward functions for each control component	Design the structure of DRL decision making algorithm	Preliminary research and literature review on DRL algorithm	Determine the network structure	Determine the activation function	Experiment by running tests on Tensorflow or Theano	Preliminary research and literature review on neural network	Analyse the given simulated data	Determine the actuator and control vector	Dynamic analysis of vessel in given environment
Start Date	30-Mar-20	23-Mar-20	07-Mar-20	07-Feb-20	30-Jan-20	23-Jan-20	07-Jan-20	08-Dec-19	01-Dec-19	01-Nov-19	15-Oct-19	07-Oct-19	30-Sep-19	23-Sep-19
Days to complete	7	7	14	30	7	7	14	7	7	30	14	7	7	14

Fig.3. Gantt Chart for deep learning obstacle avoidance project

The task date and individual task in Fig.3. is all subjected to changes, as I have no specific dates on 19/20 third year project assessment so far. If there is still time beyond 07/04/2020, I would conduct more experiments(training) and further optimize the system. Finally down below is the risk management plan.

Risk type	Risk control	Priority
failure to deliver the algorithm or algorithm fails to meet requirements	The finishing schedule should be set at least 2 weeks before assessment. Meanwhile, in an unlikely event, I have friends at both UCL CS and Imperial College CS doing machine learning work, I can consult them on difficulties. But always, I believe I can finish this, alone.	first
failure to finish each task before their deadlines i.e. risk of behind schedule	Set a 2-day buffer zone: attempt to finish task 2 days before deadline for each task, leaving time to check	second
risk of over-budget	Frequently check expenditure and remaining budget, plan ahead to avoid	third

Fig.4. Risk management plan

```
#include <Wire.h>
#include<Energia.h>
#include "accelerometer.h"
#include <math.h>
#include <stdint.h>
void setup() {
  Serial.begin(9600);
  Wire.begin();
  pinMode(RED_LED, OUTPUT);
  digitalWrite(RED_LED, LOW);
  //pinMode(YELLOW_LED,OUTPUT);
  //digitalWrite(YELLOW_LED, LOW);
//  pinMode(GREEN_LED,OUTPUT);
 // digitalWrite(GREEN_LED, LOW);
}

void loop() {
//accelerometer
  AccData acc = readAccelerometer();
  float z_gravity= 66.0;
  float gravity = 9.81; //in m/s^2
  float x_range = 127.0; //digital unit
  float y_range = 127.0; //digital unit
  float z_range = 130.0; //digital unit

  float para_z = acc.z-z_gravity; //to eliminate gravity element in z-axis of accelerometer

  float x = (acc.x/x_range)*2*gravity; //convert digital unit to m/s^2
  float y = (acc.y/y_range)*2*gravity;
  float z = (para_z/z_range)*2*gravity;

  float force_acceleration= sqrt(sq(x)+sq(y)+sq(z)); //in m/s^2
  //float force_acceleration = acc.z;
  float shock_boundary = 2.0; // in m/s^2 ,   to be set by users

 //print
   Serial.println("Current acceleration onto the package");
   Serial.print(force_acceleration);
   Serial.println(" m/s^2");

 //delay(1000);

   if (force_acceleration > shock_boundary) {
     digitalWrite(RED_LED, HIGH);
     Serial.println("A serious shock has just occurred!");
     delay(1000);// 1s
   }

   digitalWrite(RED_LED, LOW);

 //LDR
 int sensorValue = 0;
 int sensorPin = 2; //a.k.a P58 select the input pin for LDR
 sensorValue = analogRead(sensorPin); // variable to store the value coming from the sensor
   Serial.println("LDR Sensor Value");
   Serial.println(sensorValue);

  if(sensorValue<4090){ //it's 4095 when there's shadow or LDR is covered
     digitalWrite(RED_LED, HIGH);
     Serial.println("The package is opened or not entirely sealed!");
     delay(1000);//1s
   }
  digitalWrite(RED_LED, LOW);

 }
```

Fig. 5. Energia codes for controlling the accelerometer and LDR on cc3200, in C languages

References:

[1],[2]: UC Berkeley CS 294: Deep Reinforcement Learning, Fall 2017,
http://rail.eecs.berkeley.edu/deeprlcourse-fa17/f17docs/lecture_3_rl_intro.pdf