# Project Proposal

## Project Name

Avoiding Small and Deformable Obstacles Using Deep Reinforcement Learning

## Domain Background

Mobile robots have been used widely across different industries and sectors. It is crucial that the robot can navigate freely in the environment, reaching its goal position while avoiding obstacle along the path. The problem of how to avoid obstacles accurately and plan its path efficiently has been a hot topic for the robotic research communities for decades. Numerous methods have been proposed (Kamil F, 2015) and the solutions have reached a good level of maturity. The obstacle avoidance sub system, closely linked to path planning module, decides the motion of the robot when knowing the position of the robot and the obstacles. The detection of obstacles is normally achieved by using a sensing system. There are many obstacle sensors such as sonar, radar, time of flight, and recently camera. Famous and efficient obstacle avoidance algorithm such as the bubble rebound could be applied on a low power low cost robot (Susnea, 2010). However, systems like these need path planning module and sensing module to work together. Computer vision method using a camera is efficient and low cost which doesn't rely on additional sensors. There are some attempts by using image segmentation, depth extraction or optical flow to detect and avoid obstacles (Zdzislaw Gosiewski & Ambroziak, 2011). However, they do not performance as well as conventional sensing method.

## Problem Statement

For a robot to navigate efficiently in home settings, it needs to be able to avoid obstacles along its path. Modern sensors such as time of flight can detect large objects but fail to detect small and deformable things such as cables. With camera, it provides rich information and gives a better opportunity to detect cables. Conventional path planning methods require turning of a large number of parameters, so it is very constrained by the system it operates in. Furthermore, modern image analysis based techniques cannot reliably detect and avoid small and deformable objects. We would like to propose a deep reinforcement learning based approach to solve this problem. Given the start and the goal position, the robot can avoid obstacles and reach the goal in the shortest time using monocular images. There has been some research in this area. Recently proposed solutions are using a deep Q network (DQN) (Fangyi Zhang, 2017) or double-Q network (Linhai Xie, 2017) on monocular camera image to detect large objects. Here we would like to apply some of these methods and explore further in the case of small and deformable objects. This project proposal is linked to my work at Dyson where we would like to seek alternative and more advanced method using vision rather than time of flight sensors.

## Datasets and Inputs

The environment I chose to simulate the world environment is Unity. Half of the world's games are built with Unity game engine. Unity offers excellent graphic rendering, supports 3D development and realistic physics engines.

From the Unity asset store, a simulated robe that has physical properties can be used as the obstacle to imitate small and deformable cables in the real world (Store).
Unity Machine Learning Toolkit (ML-Agents) is an open source Unity plugin that allow games and simulations to serve as the environment for training the agent (ml-agents).
All data set used for training can be generated using ML-Agents and Unity game engine. The agent could be trained in different terrain surfaces and the cable could be placed in different configurations and positions in the environment.
The input for the agent is a sequence of images as the robot moves around. The position of the camera can be either a forward facing camera on the robot or a fixed position camera in the setting.

## Solution Statement

To solve the problem, the agent would use the deep reinforcement learning method, safely navigating to the target locations in the grid world. The implementation of grid world could also be found in Unity, the ML-Agent grid world example (The ML Agent Grid World Example).
The reward function could be -0.01 for every step, +1.0 if the agent navigates to the goal position and -1.0 if the agent navigates to the grid where the obstacle is. The action space is of size 4, which corresponding to cardinal movements.
There are a few deep reinforcement learning algorithms to be experimented and compared. If time allows, more research will be explored to propose an improved version.

## Benchmark Model

To benchmark the algorithm, conventional obstacle sensing and path planning can be used to provide a comparison. We will implement a sensing model using two line of sight sensors to detect the object, and a basic path planning module such as the pure pursuit algorithm to decide the robot's next movement. We could measure the number of bumps occurring during navigation in test data, compared against deep reinforcement learning method. The agent path difference between different algorithms could be calculated from the start to the goal, and this would be used to evaluate the efficiency of the algorithms.

## Evaluation Metrics

*The evaluation metrics could be:*
*1. Average rewards and variation of rewards*
*2. Complexity of the model, e.g. CNN layers, episode length*
*3. Learning rate, policy entropy, policy loss*
*4. Path difference based on benchmark model*
*Inevitably, the metrics will change during the development of the project implementation.*

## Project Design

This project scope is relatively large and challenging given the tight project timeline. But it is worthy of implementing

and exploring. The steps I would take are the following.

1. Set up the Unity environment and follow through the tutorials
2. Set up ML-Agent and familiarise its structure through examples
3. Experiment on different locations to obtain the camera images
4. Choose and build a reinforcement learning algorithm to train
5. Build Simulation of the Rope as obstacles
6. Train the robot agent in different home settings and cable shapes
7. Evaluate obstacle avoidance performance on simulated test data

## Reference


Kamil F, T. S. (2015). A Review on Motion Planning and Obstacle Avoidance Approaches in Dynamic Environments. *Advances in Robotics & Automation , 4* (2).

Linhai Xie, S. W. (2017). Towards Monocular Vision based Obstacle Avoidance through Deep Reinforcement Learning. *Robotic Science and Systems.* Cambridge.

*ml-agents*. (n.d.). Retrieved from https://github.com/Unity-Technologies/ml-agents

Store, U. A. (n.d.). *Unity Asset Store Robe*. Retrieved from https://assetstore.unity.com/packages/tools/physics/obi-rope-55579

Susnea, I. &. (2010). The bubble rebound obstacle avoidance algorithm for mobile robots. *Control and Automation (ICCA).*

*The ML Agent Grid World Example*. (n.d.). Retrieved from https://github.com/Unity-Technologies/ml-agents/blob/master/docs/Learning-Environment-Examples.md

Volkan Sezer, M. G. (2012). A novel obstacle avoidance algorithm: "Follow the Gap Method". *Robotics and Autonomous Systems , 60* (9), 1123-1134.

Yang, S. K. (2017). Obstacle avoidance through deep networks based intermediate perception. *CoRR.*

Zdzislaw Gosiewski, C. J., & Ambroziak, L. (2011). Vision-based obstacle avoidance for unmanned aerial vehicles. *International Congress on Image and Signal Processing.* Shanghai.