# R2/Sanit Gupta/160100010

August 6, 2020

Flying a helicopter is a very hard control problem. Both the state and action spaces are large. Even a task like hovering which seems to be simple on first impression turns out to be complicated (because simply turning on the main rotor to get the appropriate amount of thrust has unintended consequences as mentioned in the paper). Although there have been previous attempts at this task, they have only succeeded in making the helicopter hover in simulation. I found the results to be extremely impressive as they were successful not just in making the helicopter hover, but also in making the helicopter perform maneuvers from flying competitions too and all of that in the real world.

A Kalman filter that takes inputs from all the sensor systems (GPS, INS, digital compass) estimates the current position of the helicopter. We need a model to choose the right action given the current position. A body coordinate system is used instead of a world coordinate system to encode the useful symmetries in the world (e.g. even simple motions which are actually identical can look completely different in a world coordinate system). A locally-weighted regression model is used to learn the dynamics of the system i.e. to predict $s_{t+1}$ given $s_t$ and $a_t$. To facilitate learning the dynamics of the system, a lot of human knowledge is incorporated into the model. This also ends up reducing the number of parameters needed to be learned. Even if we have a computer simulator of the system, estimating the utilities of policies is a hard problem. The standard method is to use Monte Carlo estimates but they fail with probability 1 to be a good estimate (?). Also, experimentally they are unsuccessful in learning anything useful. This is where the PEGASUS algorithm comes in. The algorithm is based on the fact that every POMDP can be converted into an equivalent deterministic POMDP. Utility estimation and policy search are much easier after this conversion. The PEGASUS algorithm, exploiting the fact that the probabilistic sample $s'$ in a simulator is actually a deterministic function of s, a and a random number p, gets us a good estimate of the utility with a relatively small number of samples. Reward functions for both tasks (hovering and expert maneuvers) are designed with great care. A lot of effort is put into designing reward functions for the second task to make sure that the helicopter doesn't end up always lagging behind and that the reward isn't coupled with the evolution of the space coordinates in time.

Some questions I have regarding the paper:

- In Section 4, it says that even with a arbitrarily large number of samples, the Monte Carlo estimate will fail to be a good estimate with probability 1. Why is it so?
- A lot of human knowledge is incorporated into the model. If this wasn't done, would the model just take some more time to learn to perform these tasks or is there a chance that it would fail to learn at all? .