# PEX 02: Behavioral Cloning with Rovers

**Due @2359 on 23 March; 200 Points**

## Help Policy

**AUTHORIZED RESOURCES:** Any, except another cadet's assignment or published solutions to the assigned problem.

**NOTE:**

- Never copy another person's work and submit it as your own. Here are a few blatant examples of copying:
    - Making an electronic copy of another cadet's solution and then modifying it slightly to make it appear as your own work.
    - Reading a printout or other source of another cadet's work as you implement your solution.
    - Completing your entire solution by following explicit instructions from another cadet, while he/she refers to his/her own solution.
    - Copying solutions that might related to these exercises from GitHub.
- This is a team effort; however, **NO CROSS-TEAM WORK IS ALLOWED**!
- Helping your classmates learn and understand the homework concepts is encouraged, but extensive assistance should generally be provided by DFCS instructors. Only provide assistance up to your depth of understanding, beyond which assistance by more qualified individuals is more appropriate and will result in greater learning. If you have to look at your solution while giving help, you are most likely beyond your depth of understanding.
- ***Help your classmates maintain their integrity*** by never placing them in a compromising position. Do not give your solution to another cadet in any form (hard copy, soft copy, or verbal).
- DFCS will **recommend a grade of F** for any cadet who **egregiously violates** this Help Policy or contributes to a violation by others. Allowing another cadet to see your assignment to help them will result in a zero on this assignment.

## Documentation Policy

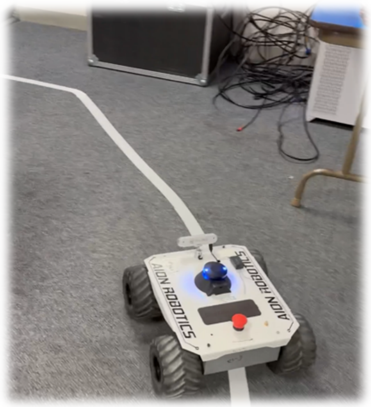
- You must document all help received from sources other than your instructor or instructor-provided course materials (including your textbook).
- The documentation statement must explicitly describe WHAT assistance was provided, WHERE on the assignment the assistance was provided, and WHO provided the assistance.
- If no help was received on this assignment, the documentation statement must state "NONE."
- If you checked answers with anyone, you must document with whom on which problems. You must document whether or not you made any changes, and if you did make changes you must document the problems you changed and the reasons why.
- Vague documentation statements must be corrected before the assignment will be graded and will result in a grade deduction equal to 5% (ceiling) of the total possible points.

## Purpose & Objectives

We're going to attempt to teach a robotic rover how to use a line as a guide for driving around a race track. This problem will be solved using a single sensor – a simple RGB camera. All of the control will be derived from a single image, a single frame from streamed video that represents a snapshot in time. This poses both a challenge and a great opportunity for applied machine learning. As discussed, we will approach this problem by applying a technique called ***behavioral cloning*** – a subset under a larger group of techniques called **imitation learning** (IL).

In general, at the heart of IL is a *Markov Decision Process* (MDP) where we have a set of states **S**, a set of actions **A**, and a **P(s' |s,a)** *transition model* (the probability that an action **a** in the state **s** leads to **s'**). This may or may not be associated with an unknown reward function, **R(s,a)**. For now, there will be no **R** function. If you're thinking to yourself that this more or less resembles reinforcement learning (RL), then you'd be correct; however, our rovers will attempt to learn an ***expert's*** optimal ***policy*** **$\pi^*$** through a simple supervised learning approach.

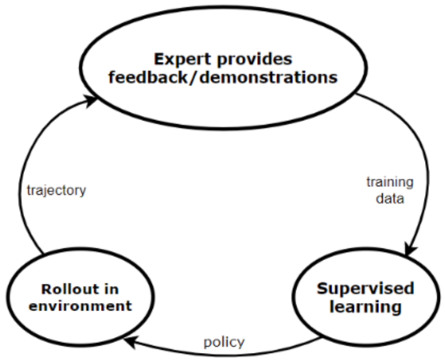
### Description of the Problem



The environment consists of the following: (1) State **s ∍ S** is represented by an observation through our video camera (i.e. a frame/image), along with any other data you might find useful. Our camera has a resolution of 640x480 with 3 8-bit (RGB) color channels. (2) A deceptively simple action space, that allows for a throttle amount and a steering amount (remember, this is a skid steer system). The **goal** is to perform **n** laps around a track defined by colored material (some sort of matte tape) laid out in an arbitrary line pattern in the fastest time possible. Ideally, our ***model will generalize*** to drive around any shape of track.

### Artifacts

When your project is complete, you'll be delivering to me several artifacts that make up a working solution: (1) access to your git repository housing your entire codebase, (2) your best serialized CNN model, (3) your data repository, (4) a **log detailing your experiments, challenges, and outcomes** with (a) data preprocessing, (b) data collection, and (c) model creation, (5) information that details the work each team member contributed to the solution, (6) an in-class presentation of your working solution.



## Getting Organized

Start out with the code each team member put together during recent labs – merge the best working code to date into one solution. Immediately create a git repository, giving each member of the team access, including me; check your initial codebase into the repository. Remember that you will need code that (a) performs data collection, (b) data preprocessing, (c) model creation, (d) model training, (e) model assessment, and (f) mission execution (i.e., model inference) in real time. Organize your code accordingly, stubbing out modules and/functions where you anticipate working in your code. Consider using OOP concepts with procedural coding design. For example, create an Agent or Rover class that will wrap and use your model during mission execution. Create organized modules that separate and group code according to functional relationships. Try to organize your code so that each team member may work on different sections of it simultaneously, preferably requiring little to no merging upon check-in.

## Data Collection

You should already have a good start on the coding effort. The idea is to collect raw streaming RGB video into a Rosbag file. Consider each recording a separate session or "lesson" that you will use to train your CNN. Come up with a way to organize your videos and their related data files so that you know who created the session, what that session consists of, for example clock-wise, counter clock-wise, steering recovery, fast, med, or high speed, etc. – whatever details you feel might be important to know about the specific session so that you may systematically choose to include or exclude it when training and performing experiments. The way you choose to organize and manage your data may become extremely important to your success.

You will need to finish your module that reads throttle and steering inputs from the "expert"/human driver, matches that data with other data from rover sensors, indexes the data with a specific frame from the video stream, and appends rows of data into a data file that is matched with a specific video session. A Rosbag file needs to be matched with a specific data file. This is all considered ***raw data***.

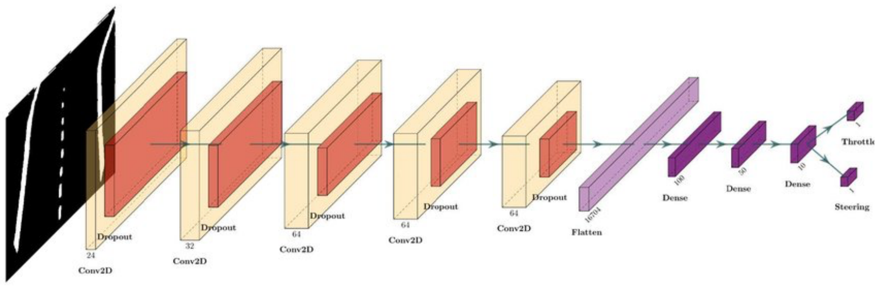
## Data Preprocessing

You may find yourselves revisiting this area most often. Keep in mind that the majority of your data is pixel-based, and the majority of your model's training time will be taken up by feature extraction from these pixels. Your model's performance will be impacted by how informative the data's features are. The objective is to come up with ways to process your data that makes it effective and efficient for your CNN to learn over. You may experiment with a host of ideas: image down sampling, color correction or transformation, image cropping, attribute enhancements (blurring, edge detection, color masking, Hough or Radon transform lines in the images, etc.). You may also consider data augmentation – ways to modify or change your data that adds new information to your dataset, effectively growing it as well as contributing towards a more balanced distribution.

The possibilities are endless, but the time you have is not. It's critical to be meticulous and organized within your experimentation efforts. Keep in mind that you may find other data to combine with the imagery that could be helpful. For example, you may find the change in heading could be useful information to your model, instead of using the raw heading values – or you may discover that heading is irrelevant. Perhaps you might combine gyro or accelerometer data with your final preprocessing. This area of the project is perhaps the most critical. Expect to spend quite a bit of your time here. Whatever you go with will have an effect on the overall framerate that your rover can operate at.

The more processing you incorporate, the lower your framerate will be. If you stick with effective, efficient image processing, you will maintain a good framerate. A good framerate should be considered >=8 FPS. The higher, the better… the more responsive your rover will be.

## Model Design



You will probably be going back to the drawing board a few times before you get your model right. I recommend starting out with (reasonably) simpler models, growing them out in complexity based on your model assessments. Start with a handful of CNN layers and perhaps three or four dense layers. I will not tell you how to build your models, but will be available to make suggestions to individual teams based on questions and analysis you provide to me. Keep track of your iterations and attach notes to each. Refer to earlier CNN lectures and your notes (hopefully, you took some). You might find yourself jumping back to earlier models if your performance is completely wrecked via experimentation with various models. Organize your efforts so that you can minimize wasted time. Try not to make changes based on blind guessing; instead, try making educated guesses as to where you might provide improvements (and in what form) based on your model analysis. The more complex your model is, the lower your final framerate will be. You will need to balance this with your data preprocessing.

## Model Training

Here you will create ways to organize, combine, and feed your preprocessed data to your model during training. You should do this through a ***data generator***. We will discuss this technique in an upcoming lab/lecture. You will need functions to randomize, split, and feed data to your model. You should also build in functions to randomize sequences of data. You may want to experiment with data that is completely randomized, semi-randomized (i.e., randomized sequences), or non-randomized (models training on entire video sequences as-is). Keep in mind that you may also want to experiment with your loss functions as well.

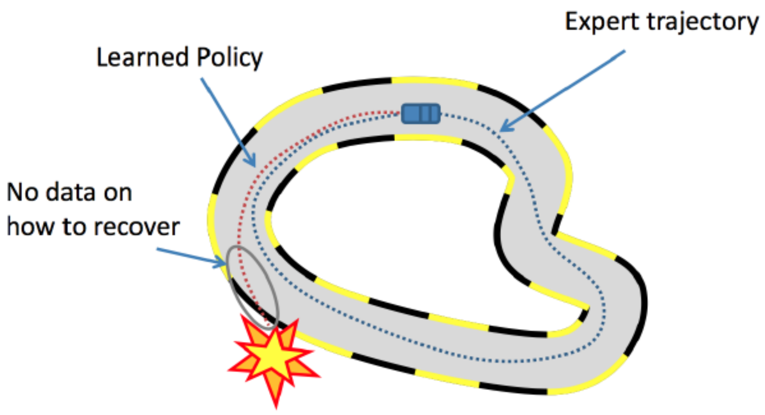
## Model Assessments

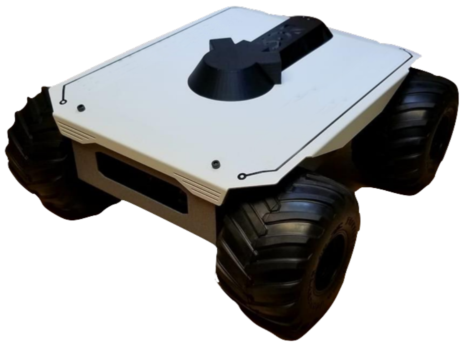
You will need to provide a way to record and plot your learning curves. You may save the raw data to regenerate your curves at any time, or you may simply export your graphs at the end of the training sessions, or both. Look for overfitting, underfitting, lack of data, other anomalies that might indicate the need for more data, better preprocessed data, regularization, or a better model. You might be surprised to discover that a lot of your own learning experience and skillset building with machine learning will happen in this area of the project. Be organized here, too. Make sure to match these assessments and store them with your model iteration and project log entries. **I may ask your team to explain/justify your final model based on your assessment artifacts**.

## Model Execution

You will need to create a Python module with a looping script (loops while rover is armed) that takes images from the camera, runs those images through whatever preprocessing your team ultimately settled on, matches that info to other sensor data (that you also decided on), and runs that data through your model's inference, and overrides the autopilots steering and throttle with the model's output. All of this, of course, is performed in real time.

The final evidence stemming from your efforts will be how well your rover is able to perform its tasks and accomplish its objective. Each team will be assessed and each rover will be sent through the race track multiple times. Each will be scored on the rover's ability to navigate the track as well as the execution speed. Each team will be ranked and a 15% portion of the overall project grade will be attributed to your team's tanking. This means that your team will want to strive to be the "winning" team.





## Final Thoughts

This is an end-to-end project. Much of the hard work will go into feature engineering and constraining the problem to make it simpler. These are precisely the tasks that take all the time in industrial ML projects as well. In industry, these solutions become standardized over time, which leads to the situation where you won't need to touch your ML algorithms as much with each modification.