**Motivation:**

The chosen project is to create a path planning algorithm which creates a trajectory for the dVRK robot moving from a desired starting position to an ending position around obstacles. The dVRK was chosen due to its interesting architecture, and the preexisting tools provided for the robot in class. Path planning with obstacle avoidance was chosen as it is an incredibly useful tool for autonomous robot control, which is an increasingly important task.

There are multiple different existing path planning algorithms that exist which could potentially be used to solve this problem. One of these algorithms is PRM, in which a roadmap of the free space of the robot is created and an efficient path through this space is solved using a searching algorithm such as A\*. The upside to this option is that once the roadmap is created an optimal path can be quickly built around any obstacles. The drawbacks are that a representation of the available free space must be created, and there are multiple options as to how to sample the joint space and detect collision points. Another option that was considered was to use RRT\*, or a variation of the rapidly-exploring random trees method which continuously improves its initial solution to find a time or distance efficient path. This method will improve the solution over time and approach an optimally short path, but it takes multiple attempted paths to reach a solution which can be more computationally intensive than our chosen method.

The path planning algorithm that was chosen for this project was the virtual potential fields method. This method was chosen due to it being more intuitive and for the ability to check the accuracy of the model through plots of the potentials and gradients that were created. The main downside to the virtual potential fields method is that you can get trapped in local minima. To prevent this, the strength of the potential fields and gradients for the goal and the obstacles can be altered to prevent the algorithm from falling into these local minimas.

**Hardware Demonstration:**

The trajectory generation code shown above was used to create a path through the joint space and around an obstacle. The output of this code was a .mat file named Trajectory, which can be loaded with our move\_trajectory function. This function was used to move the dVRK through a generated path, as shown in the video below:

**Summary:**

To solve for the trajectory, the obstacle and goal potentials were both found in the joint space using the equations for each potential in the textbook. The gradient for the obstacle and goal could then be found in 3D space over each potential. Both gradients and potentials can then be added to give a total gradient and potential across the joint space. The overall gradient was then moved through beginning at the start position until the desired goal position was reached. This was then saved as a trajectory in the joint space, which was passed to the dVRK to cause it to move through the calculated path.