# AI for Robotics II
# Assignment 2: Task and Motion Planning

## 1 Introduction

The aim of this assignment is to familiarize you with modeling integrated task and motion planning problems. Kindly note that there exists different approaches to integrate task planning and motion planning and what we discuss here is one such approach.

Let us revisit the coffee shop scenario from the previous assignment. The waiter robot need to serve the orders in an optimal way (example, distance travelled) which might require choosing a certain sequence of tables to deliver the orders. Thus, deciding the discrete sequence of table visits require reasoning about the distance traversed by the robot. Keeping this scenario in mind, we will model a very abstract version of the same wherein we consider a robot starting at a certain region in the environment and has to navigate to four different regions. A region, for example correspond to a coffee table. In reality the robot can be anywhere around the coffee table while delivering the order. However for the sake of brevity of the assignment we associate a single way-point (location and orientation considered here) to a region. Thus a task of delivering to a table geometrically correspond to the robot going to the way-point associated with that table.

The first thing to do is to install the popf-tif planner, the details of which can be found here- https://github.com/popftif/popf-tif. Make sure that the planner is running without any errors on your laptop. Once this is done, extract the domain and module folders given along with this assignment. The domain folder contains a PDDL domain file consisting of a single action and a corresponding problem file where a robot has to visit four regions. The REGION_POSES file contain the mapping from a region to its corresponding way-point. The WAYPOINT.TXT file contain the geometric way-point locations for the four regions to visit as well as the starting region. The LANDMARK.TXT file contain landmarks to localize the robot (more on this later).

The module folder contains the external module files. Let us take a look at the domain file given. The functions *triggered, act-cost, dummy* are the ones that involves the external module or semantic attachment part. The results of the computations in the external module part are stored in the *dummy* variable and this is assigned to *act-cost*. The function *triggered* is implemented so that the task planner can communicate the start and goal region to the external module. These can be comprehended upon careful examination of the **loadSolver**, **callExternalSolver** and **calculateExtern** functions within the VisitSolver.cpp file in the module folder. The instructions to build the external module can be found in the buildInstruction.txt file. Once built, the planner can be run using the command (refer- https://github.com/popftif/popf-tif)

```
../popf3-clp -x dom1.pddl prob1.pddl
                        ../visits_module/build/libVisits.so region_poses
```

If the external module is build correctly, the planner should run without any errors. *dummy* is randomly given a value of 2 and the cost returned by the planner should therefore be 8. You can check its working by varying the value in **calculateExtern** function. Study the domain, problem files (in the domain folder) and the files main.cpp, VisitSolver.cpp. This should give you sufficient understanding regarding the external module implementation required for this assignment.

## 2 To do

You are required to write an additional action in the PDDL domain file that computes the Euclidean distance travelled between two regions. Thus, the given action `goto_region` now models only the fact that the robot has reached a particular region and therefore this actions can be modified. The additional action that you define should compute the `act_cost`, that is, the distance traversed by the robot. This action will also thus require the *triggered* function– you can also try to come up with your own trick to communicate the start and goal region to the external module; since computing the distance require start and goal locations. Kindly see the commented **localize** function call within the **callExternalSolver** function in VisitSolver.cpp– this is where you will have to call the function to compute the distance between the two regions as the `from` (start) and `to` (goal) regions are extracted here. Also note the parsing functions for the way-point file to compute the required distance.

Once the distance computation is done, additionally you could also compute the cost due to uncertainty. For this purpose, assume an initial covariance of $diag(0.02m^2, 0.02m^2, 0.02rad^2)$. Assuming any motion model for the robot (odometry model or velocity model) estimate the final robot state using extended Kalman filter (EKF) or any other estimation technique. Since landmark locations are given, the distance and orientation to the landmarks should be taken as the measurements (corrupt this computation with some added zero mean noise). The trace of the resulting covariance matrix should be taken taken as the cost due to uncertainty and then added to the distance cost. Note that one needs additional modalities to keep track of the covariance as the robot arrives at each region. To make this simple enough, you can assume the covaraince to be $diag(0.02m^2, 0.02m^2, 0.02rad^2)$, each time you perform EKF at different regions. You can also modify the WAYPOINT.TXT and LANDMARK.TXT files and use your own values.

Finally, you will have to submit a report that briefly outlines what you have implemented, that is, the action that you have added and how you modeled it. Also submit the PDDL domain and problem files (WAYPOINT.TXT and LANDMARK.TXT files, if you have modified them) along with the modified files in the module folder.