Practical seminar task assignment

Multi-robot Systems (MRS) group at Czech Technical University in Prague

September 2020

Practical seminar task

The practical seminar focuses on solving the min-max Multiple Traveling Salesman Problems with Neighborhoods (MTSPN) using two drones. The problem consists of n target locations placed throughout the fly area with a predefined circular neighborhood which has to be visited. Both UAVs have a predefined starting position and limited maximal velocity and acceleration. The objective of the task is to minimize the maximal flight time of the drones and to visit all target location neighborhoods. During the simulated and real flight, each unvisited target location is penalized by 5 s and a potential collision between the drones by 10 s. The assignment is given with already working planner. However, the planner has poor performance and thus can be significantly improved.

How to start

Clone or download template of the seminar task at https://github.com/ctu-mrs/mtsp_planning_task to ~/workspace/src/, e.g., by running

```
cd ~/workspace/src/
git clone git@github.com:ctu-mrs/mtsp_planning_task
```

Make sure to install dependencies by running *install_dependencies.sh*. Afterward, you can compile the workspace by running *catkin build* in the ~/workspace. After installing the dependencies, you can try to run the provided planner. It can be run either in ROS (1.) with the simulated flight of UAVs or independently (2.) with plots of the plans and corresponding velocity profiles. The second option is viable even on a computer without ROS and the MRS simulation pipeline.

1. Run tmux script in `mtsp_state_machine/tmux/test/start.sh` to start Gazebo simulation of two drones that fly the planned trajectory.
2. Execute the provided planner by running `python mtsp_planner/scripts/planner.py` in a terminal. This will plot the trajectories and velocity profile for both drones.

Please keep your code with you e.g. by using USB stick or uploading to cloud. Do not commit to our git project and expect your code in the provided PCs to be cleaned before next practicals.

Current code structure

There are four packages containing parts of the assignment. The *mtsp_msgs* contains ROS message definitions for, e.g., sending MTSP problem definition. The *mtsp_planner* contains the MTSP(N) planning part that is the one you will work on. Package *mtsp_problem_loader* contains MTSP problem loader that parses given problem file and sends it to the planner. Finally, the *mtsp_state_machine* control the task execution (takeoff, flying to start, following the trajectory) and distribution of the messages.

Important files that are to be improved or modified are:

- *planner.py* in mtsp_planner/scripts contains the high-level part of the planning. It can be run both in ROS, where the problem is sent by a message from mtsp_problem_loader, and also independently from a terminal (with the plotting of trajectories and velocity profiles).
- *tsp_trajectory.py* in mtsp_planner/scripts/solvers creates sampled trajectory for UAVs to be used by the MPC trajectory tracker. An important part is that the sampled trajectory respects the vehicle velocity and acceleration constraints. Otherwise, the employed MPC tracker would make it feasible by sacrificing the position precision.
- *tsp_solvers.py* in mtsp_planner/scripts/solvers contains methods for planning various types of TSP, called in planner.py.
- *simulation.yaml* in mtsp_planner/config contains default configuration of parameters such as *max_velocity*, *max_acceleration* and *turning_velocity* to be used in the MTSP solvers and trajectory generation.

Provided planners

Look inside planner.py for *TSP SOLVERS PART BEGIN* part where four solution approaches are provided (three of them commented out). All approaches use initial clustering of target locations (see TARGET LOCATIONS CLUSTERING BEGIN part) and consequently the LKH TSP solver [1] for a single vehicle. Initially, the target locations are divided into two clusters by simple splinting of the target location list. The approaches that are already prepared are:

1. method *tsp_solver.plan_tour_etsp* finds a solution of Euclidean Traveling Salesman Problems (ETSP) over individual clusters of target locations using LKH
2. method *tsp_solver.plan_tour_etspn_decoupled* finds a solution of Euclidean Traveling Salesman Problems with Neighborhoods (ETSPN) over individual clusters of target locations by graph search of the sampled neighborhood using sequence of targets found as ETSP,
3. method *tsp_solver.plan_tour_dtspn_decoupled* finds a solution of Dubins Traveling Salesman Problems with Neighborhoods (DTSPN) over individual clusters of target locations,
4. method *tsp_solver.plan_tour_dtspn_noon_bean* finds a solution of Dubins Traveling Salesman Problems with Neighborhoods (DTSPN) using Noon-Bean transformation [2] to convert the DTSPN to ATSP and solved using LKH.

Assignment tasks

The tasks are designed such that their complexity increases, and each task can improve the solution quality compared to the previous ones.

1. **Testing of various planners** - initially try to test various MTSP planning approaches that are available in the code already (by commenting out and uncommenting appropriate solvers) and try to find the ones with the shortest maximal time of trajectory execution
2. **Try different parameters** - try different turning velocity, maximal velocity, and acceleration (with respect to the UAV limits) and see the effects on the trajectory flight times
3. **Improve naive distribution of targets among UAVs** - try different distribution/clustering of targets using, e.g., k-means clustering
4. **Use the time of flight instead of length in DTSPN solver** - in DTSPN with Noon-Bean transformation you can modify the planner to use time of flight instead of length for the TSP arc weights

If you are fast enough, you can optionally implement further features/solvers to even more improve the solution quality:

5. **Implement operators that improve solution quality** - you can implement, e.g., Local Iterative Optimization (LIO) [3] for hill climbing improvement of heading and neighborhood position samples. You can also try to implement e.g. two-opt operator that improve single trajectory, or other similar operators that switches target locations between the clusters for individual drones
6. **Implement MTSP construction heuristic** - try to implement Initialization procedure (5.1) from [4] preferably with a time of flight metric used in 4.
7. **Implement operators that fix mutual collisions** - as the possible collisions are penalized and are avoided by changing altitude and velocity, the trajectories can be improved in case of collision by, e.g., changing assignment of the targets between drones
8. **Try different motion primitives** - you can try to implement different motion primitives, e.g., splines, and appropriate velocity profile, e.g., using QP optimization, to further improve the solution quality

Explore another possible usage of the MRS UAV system

Based on the presentation of the MRS system, you can also try other capabilities of the system. You selected a group of practicals based on your scientific interest, feel free to ask during the summer school and especially during the seminars how the system can be used for your area of interest.

[1] Helsgaun, K. An effective implementation of the Lin–Kernighan traveling salesman heuristic, European Journal of Operational Research, Volume 126, Issue 1, 2000, Pages 106-130, (http://akira.ruc.dk/~keld/research/LKH/)

[2] Noon, C. E., & Bean, J. C. (1993). An efficient transformation of the generalized traveling salesman problem. INFOR: Information Systems and Operational Research, 31(1), 39-44.

[3] P. Váňa and J. Faigl, On the Dubins Traveling Salesman Problem with Neighborhoods, 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, 2015, pp. 4029-4034.

[4] Faigl, J, Váňa, P, Pěnička, R, Saska, M. Unsupervised learning-based flexible framework for surveillance planning with aerial vehicles. J Field Robotics. 2019; 36: 270– 301. https://doi.org/10.1002/rob.21823