Read Me Design Project

This is a Read Me to the Design Project of “Collision-free Formation Trajectory Generation for Autonomous Transport Robots”. The code of the previous group has mainly been modified in the cost function. The functions *cost_cargo_inside_static_object()* and *cost_cargo_outside_bounds()* were added which compute additional costs if the cargo and obstacles/bounds collide. We designed multiple approaches that are discussed in the corresponding paper. Additionally, we found a library that computes a collision value. We converted this library into casadi code (*collision_cs*) by stripping many parts and converting if-statements into fmin and fmax functions. But still, in the current state, the solver can’t compute a solution for this approach. All approaches can be chosen in the function-call.

Regarding the rest of the code, we added some parameters in the solver

- self.vx_static_unpad
- self.vy_static_unpad
- self.bs_static_unpad
- self.a0s_static_unpad
- self.a1s_static_unpad
- self.b_bounds_unpad
- self.a0_bounds_unpad
- self.a1_bounds_unpad
- self.vx_bounds_unpad
- self.vy_bounds_unpad

from which only the last 6 are used in the current state of the code. We developed the two first approaches corresponding to the cost function *cost_inside_static_object* which was designed by the students from the master thesis we build upon. There the H-representations (half-space representations) of the obstacles are computed with parameters from the solvers. They also statically defined how many obstacles there are for each obstacle type. This is done so that it’s not needed to build the solver newly every time the obstacles or map changes. On the other hand, it is an immense computational effort compared to computing these parameters only for the given obstacles. The latter is what we have done at the beginning in *obs_import()*. The rest of the solver parameters mentioned above are for changing to a fixed size implementation like in *cost_inside_static_object()* and *cost_cargo_outside_bounds()*.

In computing the unpadded obstacle vertices there is an error in the computation of the original polygon from inverting the padding in the *obstacle_handler*.

Furthermore, we made changes in plotter.py to animate the cargo and change its color when it’s colliding.

Here is a list of important TODOs which need to be addressed such that the code is more robust:

- Need to create separate weighting for the cargo collision detection in the yaml and then call this one. Then this parameter needs fine-tuning for robust solver performance (feasibility and yet collision avoidance for most of the time).
- Possibly the calculation of d_obs is not really necessary. For us, it improved feasibility but maybe can be robustly designed by smart weighting instead
If this is kept, the center point calculation for any convex polygon needs to be generalized. Or come up with a better way to influence the cost factor to perform robustly.
- Weirdly enough, it happens often that the ATRs don't actually reach their final goal. The simulation thus never stops. This needs to be investigated.
- In the fnc *convert_static_obs_to_eqs_and_verts* we have set arbitrary values for not used parameters (e.g. not the maximal number of obstacles chosen). These values are chosen such that they are unrealistic to be used for most scenarios. For robust behavior, this needs to be fixed somehow. It can’t be 0 since then the cargo couldn’t be moved over the origin in the plane (since then the vertex 0,0 would be inside the cargo).
- Every polygon needs to be defined in order and anti-clockwise to set the sign of the half-spaces correctly (for clockwise definition all half-spaces would be positive outside and negative inside of the polygon. Then a collision would be detected when they are not colliding). This needs to be checked for the bounds outside of the MPC structure (and then not revert the order in *cost_cargo_outside_bounds()* as well as somehow for the cargo.

In general, for collision checking both “collision” and “shapely” as well as the methods presented in the paper were used.
When the solver has problems finding a nontrivial solution it helps to increase the aggressive_factor to let the solver have more freedom in the solution.
For relevant questions, you can contact one of us with: johannes.kuebel@gmx.de