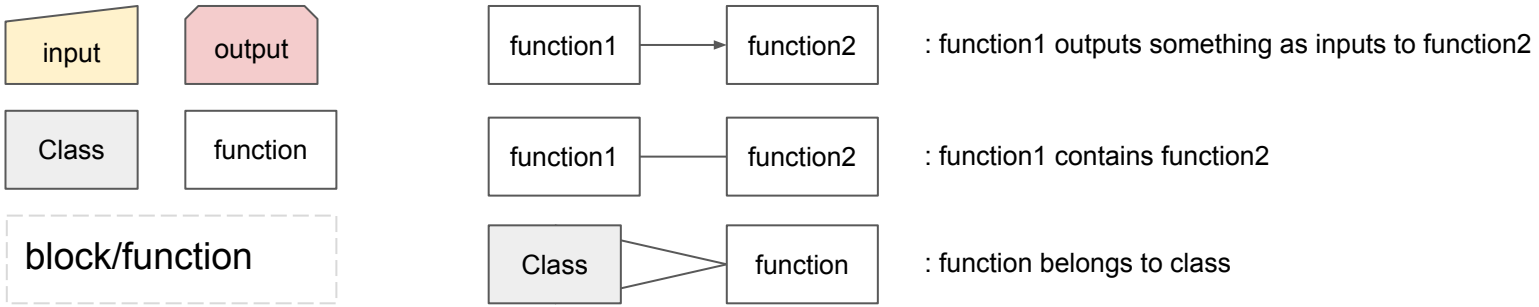
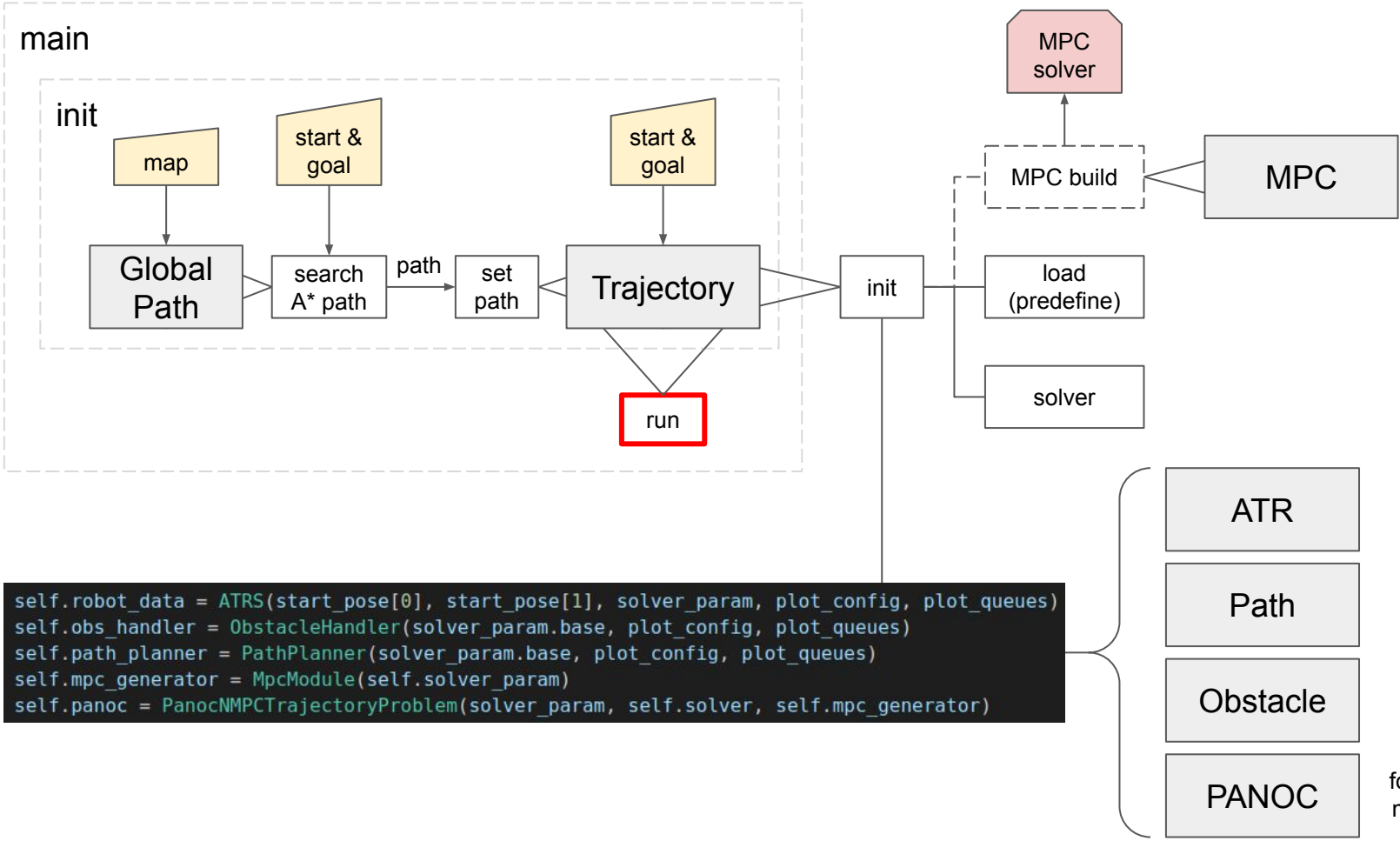
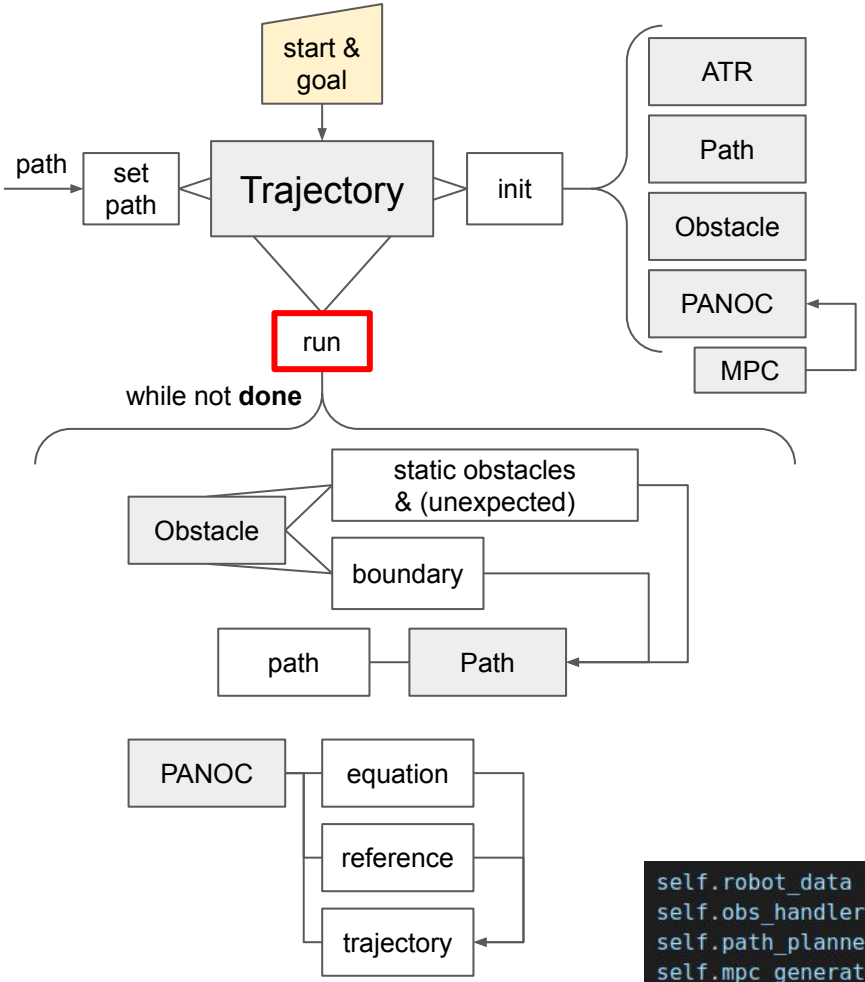
Explanation:

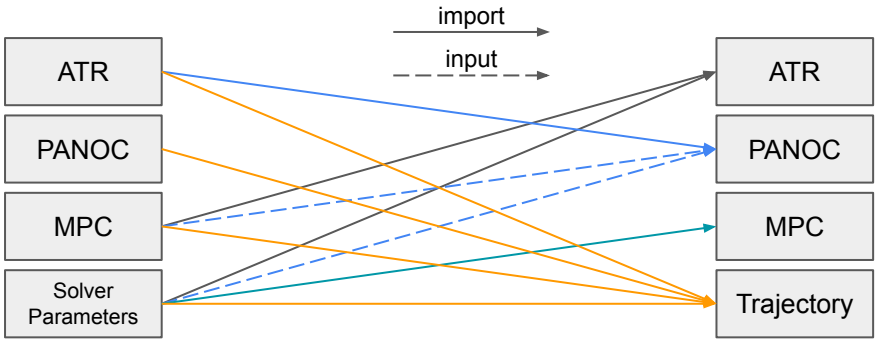


The basic structure of the code is:

1. Form MPC and build the solver.
2. Generate global path and local path (for unexpected static obstacles) as reference path.
3. Put reference and the solver to the trajectory generator.


main
init
map
start & goal
start & goal
Global Path
search A* path
path
set path
Trajectory
run
MPC solver
MPC build
MPC
init
load (predefine)
solver
self.robot_data = ATRS(start_pose[0], start_pose[1], solver_param, plot_config, plot_queues)
self.obs_handler = ObstacleHandler(solver_param.base, plot_config, plot_queues)
self.path_planner = PathPlanner(solver_param.base, plot_config, plot_queues)
self.mpc_generator = MpcModule(self.solver_param)
self.panoc = PanocNMPCTrajectoryProblem(solver_param, self.solver, self.mpc_generator)
ATR
Path
Obstacle
PANOC
get and store states of ATR
form the NMPC mathematically





Todo:

1. Simplify all the functions if possible;
2. Get rid of redundant variables and functions;
3. Reorganize the structure;
4. Replace CGAL by “scipy.spatial.Delaunay”;

5* Add “emergency stop” and quick restart;

6* Optimize the visualization.

7* Get rid of or optimize the ATR class.

```
self.robot_data = ATRS(start_pose[0], start_pose[1], solver_param, plot_config, plot_queues)
self.obs_handler = ObstacleHandler(solver_param.base, plot_config, plot_queues)
self.path_planner = PathPlanner(solver_param.base, plot_config, plot_queues)
self.mpc_generator = MpcModule(self.solver_param)
self.panoc = PanocNMPCTrajectoryProblem(solver_param, self.solver, self.mpc_generator)
```