Quadrotors Scenarios

This documents provides details on the test scenarios chosen for the quadrotors application platform of the CompLACS project.

1 Scenario 1: Cats and Mouse game

Objective: Control N quadrotors[1] (cats) in order to catch another quadrotor (mouse) at the end of the allotted time.

Setup: At the beginning of the task the mouse is placed at the center of the flight space[2] and the cats are positioned randomly (within reason) around the mouse. Initially all the helicopters are stationary. As the task starts the mouse helicopter tries to escape the cats by moving at a constant[3] (max) speed and following a predefined control law that prioritize escaping from the closest cats. The task final reward is computed as the sum of the squared distances of the cats to the mouse at the end of the allotted time. A large negative reward is returned if any of the helicopters goes outside of the flying area or if any collision happens during the task.

Since how hard the control problem is depends on the level of sensor noise and wind disturbance, we provide three versions of the task with in increasing level of difficulty:

- *noiseless:* the dynamics of the quadrotors is deterministic and the state returned is the true platform state;
- *noisy:* the dynamics of the quadrotors is stochastic and the state returned is a noiy estimate of the platform state;
- *noisy and windy:* the dynamics of the quadrotors is stochastic and affected by wind disturbances, the state returned is a noiy estimate of the platform state.

State: For each of the helicopters the full estimated state eX of cats and mouse is made available:

$$eX = [\tilde{p}_x; \tilde{p}_y; \tilde{p}_z; \tilde{\phi}; \tilde{\theta}; \tilde{\psi}; 0; 0; 0; \tilde{p}; \tilde{q}; \tilde{r}; 0; \tilde{a}_x; \tilde{a}_y; \tilde{a}_z; h; \dot{p}_x; \dot{p}_y; \dot{h}]$$

Actions: To limit the space of control inputs we do not expose directly the four control input of each quadrotor, instead we provide each quadrotor with a PID controller that allows it to

[1] N is usually 3.

[2] Without loss of generality since the control problem depends on the relative positions of mouse and cats as long as the flying area is sufficiently large.

[3] While the PID controller that governs the quadrotor attempts to maintain a fix maximum speed, in practise this will not always be the case due to sensor noise wind disturbance and dynamic effects.

maintain a fixed altitude and heading and to receive commands in terms of a 2D linear velocity[4]. Therefore for each of the (cats) helicopters the available action is in the form:

$$u = [v_x, v_y]^T,$$

the mouse insteads moves autonomously.

Code: All the ingredients of the scenario described above are implemented as a task class for the quadrotor simulator QRSim[5]; the three variation of the scenario are named (rather obviously) `TaskCatsMouseNoiseless.m` `TaskCatsMouseNoisy.m` and `TaskCatsMouseNoisyAndWindy.m`.

An example file `main_catsmouse.m` shows how to initialize and run a task and can be easily modified to test learning and control algorithm that generate actions for the cats.

The task, the configurations and the example main files are in the directory `scenarios/catsmouse` within the qrsim simulator.

$\tilde{p}_x$	x position estimate from GPS (NED coordinates)	m
$\tilde{p}_y$	y position estimate from GPS (NED coordinates)	m
$\tilde{p}_z$	z position estimate from GPS (NED coordinates)	m
$\tilde{\phi}$	roll attitude in Euler angles right-hand ZYX convention	rad
$\tilde{\theta}$	pitch attitude in Euler angles right-hand ZYX convention	rad
$\tilde{\psi}$	yaw attitude in Euler angles right-hand ZYX convention	rad
$\tilde{p}$	rotational velocity around x body axis from gyro	rad/s
$\tilde{q}$	rotational velocity around y body axis from gyro	rad/s
$\tilde{r}$	rotational velocity around z body axis from gyro	rad/s
$\tilde{a}_x$	linear acceleration in x body axis from accelerometer	m/s^2
$\tilde{a}_y$	linear acceleration in y body axis from accelerometer	m/s^2
$\tilde{a}_z$	linear acceleration in z body axis from accelerometer	m/s^2
h	altitude[6]from altimeter NED	m
$\dot{p}_x$	x velocity from GPS (NED coordinates)	m/s
$\dot{p}_y$	y velocity from GPS (NED coordinates)	m/s
$\dot{h}$	altitude rate from altimeter NED	m/s
v_x	desired x velocity control (NED coordinates)	m/s
v_y	desired y velocity control (NED coordinates)	m/s

[4]In addition to reducing the number of control inputs this makes the task closer to what is possible to implement and test using real quadrotors.

[5]We refer to the QRSim manual for details on the simulator API `http://complacs.cs.ucl.ac.uk/complacs/simulator/manual.pdf`.