VenomSim Simulator Document

Xiaotian Hu

2018/4/21

Contents

C++ interface

siminit

C++ specification

void **siminit** (double initx, double inity, double initz,
double initroll, double initpitch, double inityaw,
double initx_t, double inity_t, double initz_t,
double initroll_t, double initpitch_t, double inityaw_t,
double speed_upbound_hunter, double speed_upbound_target,
double yawdot_bound, double yawdot_bound_t)

Parameters

initx, inity, initz

the position of the hunter. they represent the 3d coordinates of the hunter position in x, y, z axis respectively.

initroll, initpitch, inityaw

the orientation of the hunter. they represents the rotation w.r.t x,y,z axis respectively. Its unit is degree.

initx_t, inity_t, initz_t

the position of the hunter. they represent the 3d coordinates of the target position in x, y, z axis respectively.

initroll_t, initpitch_t, inityaw_t

the orientation of the target. they represents the rotation w.r.t x,y,z axis respectively. Its unit is degree.

speed_upbound_hunter

The speed upbound for the hunter.

speed_upbound_target

The speed upbound for the target.

yawdot_bound

The yaw angular speed upbound for the hunter. It is a float scalar. Its unit is degree/second. The default value is 180°/s.

yawdot_bound_t

The yaw angular speed upbound for the target. It is a float scalar. Its unit is degree/second. The default value is 180°/s

Return

no return

Description

It is the interface to init the simulator. To init the simulator, the initial position and orientation of the hunter and the target should be provided. High order states including the velocity and acceleration of the target and the hunter is initialized with 0. And the speed upbound and angular

speed for yawing should also be set.

simrun

C++ specification

void **simrun** (double roll, double pitch, double yaw, double throttle,
double roll_t, double pitch_t, double yaw_t, double throttle_t,
unsigned long long period)

Parameters

period

the time span of this simulation step in nanosecond. It is a integer scalar. The smallest time span is 5555555 nanoseconds(corresponds to 180 simulation steps per second or fps = 180).

roll, pitch, yaw, throttle

The control commends for the hunter. row and pitch specify the angle by which the hunter should rotate with respect to the x axis and y axis respectively. yaw specify the angular speed the hunter should rotate in with respect to the z axis. The throttle specifies force the hunter should generate, These four elements are normalized with value range [-1,1]

roll_t, pitch_t, yaw_t, throttle_t

The control commends for the target. It is a list of length 4 in the form of [row, pitch, yaw, thrust]. The meaning of these four elements are the same as the huntercmd mentioned above. targetcmd is by default set to be none, which corresponds to the target being static.

Return

no return

Description

simrun is the interface to control the hunter drone and target drone. The length of the simulation step is also set by this interface. Please notice the different meaning of roll, pitch and yaw.

siminfo

C++ specification

infoformat * **siminfo**()

Parameters

no parameter.

Return

siminfo returns a pointer to a chunk of memory storing the simulated data. The data in stored in infoformat structure, whose definition is:

```
struct infoformat
{
        double posx;        double posy;        double posz;        //position of hunter
        double velocityx;   double velocityy;   double velocityz;   //velocity of hunter
        double accx;        double accy;        double accz;        //acceleration of hunter
        double thetax;      double thetay;      double thetaz;      //orientation of hunter

        double posx_t;      double posy_t;      double posz_t;      //position of  target
        double velocityx_t; double velocityy_t; double velocityz_t; //velocity of  target
        double accx_t;      double accy_t;      double accz_t;      //acceleration of target
        double thetax_t;    double thetay_t;    double thetaz_t;    //orientation of target

        double thrust;                                              //thrust of hunter
};
```

Description

siminfo is the interface to access the simulated data.

simstop

C++ specification

simstop()

Parameters

no parameter.

Return

no return.

Description

simstop is the interface to stop simulation. It must be called to release the resource occupied by the simulator when the simulation is supposed to stop.

installcamera

C++ specification

void **installcamera** (double roll, double pitch, double yaw,
double FOVleft, double FOVright, double FOVbottom, double FOVtop,
double FOVnear, double FOVfar,
double screenwidth, double screenheight)

Parameters

roll, pitch, yaw

the install orientation of the on-drone camera. roll, pitch and yaw represent the rotation with respect to x, y, z axis respectively. they are in degree ranging from -180° - 180°. Shown as the figure 1, to make the camera point the positive direction of x axis, the camera should be rotated with respect to the z axis for 180° (yaw for 180°). In this case, the installori should be set as [0,0,180]

FOVleft, FOVright, FOVbottom, FOVtop

The left, right, bottom and top position of line segment of square from the nearest plane of view cone. Shown in the figure 2.

FOVnear, FOVfar

the nearest and furthest plane of the camera view cone, depicted by figure 2. Its unit is degree

return

no return

Description

installcamera is used to initialize the projection function. Through the interface, the orientation and view cone of the on-drone camera can be adjusted.

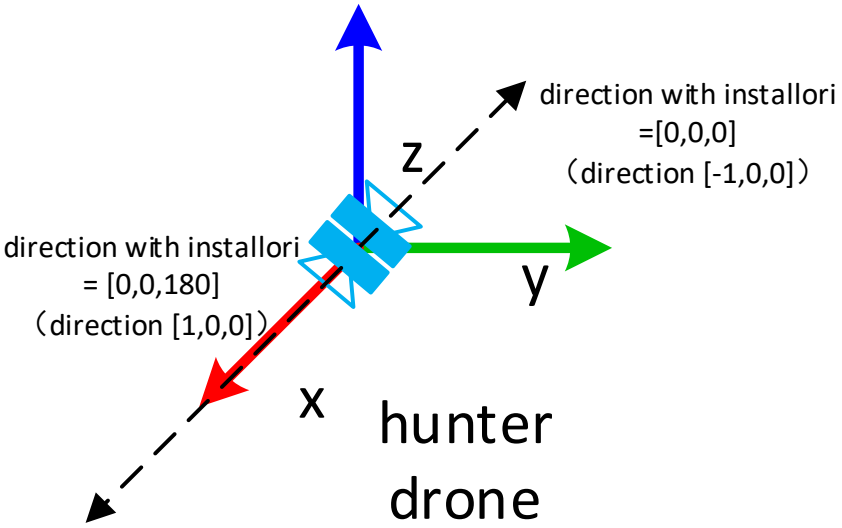

Figure 1 the illustration for installori

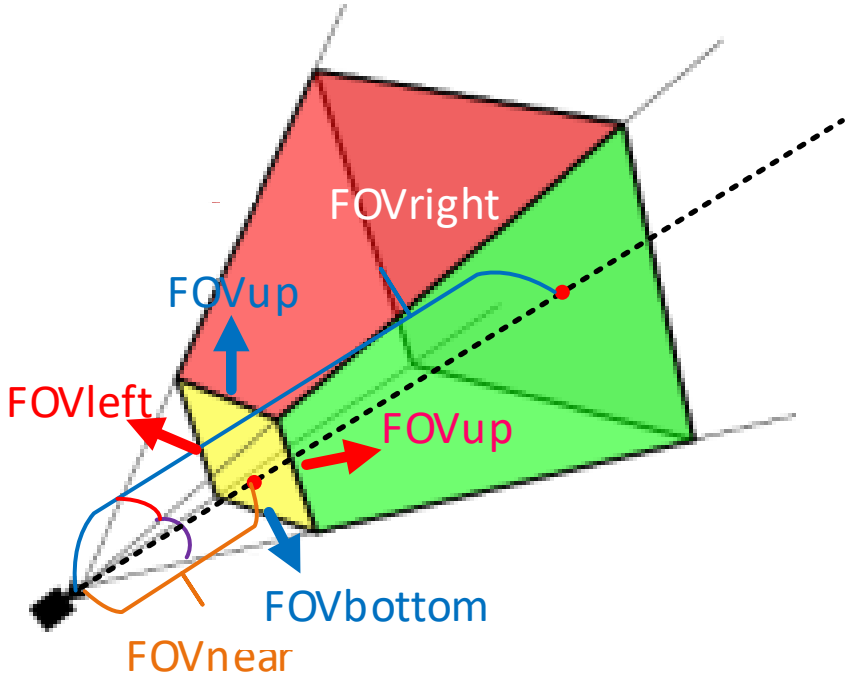

Figure 2 the illustration for view cone

simprojection

C++ specification

imagecoor * **simprojection** (double hx, double hy, double hz,
double hroll, double hpitch, double hyaw,

double tx, double ty, double tz,
double troll, double tpitch, double tyaw,
double screenwidth, double screenheight)

Parameters

hx, hy, hz

the position of the hunter. hx, hy and hz represent the 3d coordinates of the hunter position in x, y, z axis respectively.

hroll, hpitch, hyaw

the orientation of the hunter. hroll, hpitch, hyaw represents the rotation w.r.t x,y,z axis respectively. Its unit is degree.

tx, ty, tz

the position of the target. hx, hy and hz represent the 3d coordinates of the target position in x, y, z axis respectively.

hroll, hpitch, hyaw

the orientation of the hunter. troll, tpitch, tyaw represents the rotation w.r.t x,y,z axis respectively. Its unit is degree.

screenwidth

the width of the image.

screenheight

the height of the image.

Return

simprojection returns a pointer to a chunk of memory storing the projection result. The data in stored in imagecoor structure, whose definition is:

```
struct imagecoor
{
    double u;      //horizontal coordinates in image coordinates
    double v;      //vertical coordinates in image coordinates
    double w;      //pseudo depth
    double area;  //projection area
};
.
```

Description

projection function is used to compute the projection of target in the image. Its return includes the projection coordinates of the geometry center of the target and the projection area.

python interface

siminit

python specification

siminit(pos_hunter,ori_hunter,pos_target,ori_target,speed_upbound_hunter,speed_upbound_target,yawdot_bound_hunter = 180, yawdot_bound_target = 180)

Parameters

pos_hunter

the position of the hunter. It is a list of length 3 in form of: [x,y,z], which represents the 3d coordinates of the hunter position

ori_hunter

the orientation of the hunter. It is a list of length 3 in form of [row, pitch, yaw], which represents the rotation w.r.t x,y,z axis respectively. Its unit is degree.

pos_target

the position of the target. It is a list of length 3 in form of: [x,y,z], which represents the 3d coordinates of the target position

ori_target

the orientation of the target. It is a list of length 3 in form of [row, pitch, yaw], which represents the rotation w.r.t x,y,z axis respectively. Its unit is degree.

speed_upbound_hunter

The speed upbound for the hunter. It is a float scalar.

speed_upbound_target

The speed upbound for the target. It is a float scalar.

yawdot_bound_hunter

The yaw angular speed upbound for the hunter. It is a float scalar. Its unit is degree/second. The default value is 180°/s.

yawdot_bound_target

The yaw angular speed upbound for the target. It is a float scalar. Its unit is degree/second. The default value is 180°/s

Return

no return

Description

It is the interface to init the simulator. To init the simulator, the initial position and orientation of the hunter and the target should be provided. High order states including the velocity and acceleration of the target and the hunter is initialized with 0. And the speed upbound and angular speed for yawing should also be set.

simrun

python specification

simrun(period, huntercmd, targetcmd = None)

Parameters

period

the time span of this simulation step in nanosecond. It is a integer scalar. The smallest time span is 5555555 nanoseconds(corresponds to 180 simulation steps per second or fps = 180).

huntercmd

The control commends for the hunter. It is a list of length 4 in the form of [row, pitch, yaw, thrust]. row and pitch specify the angle by which the hunter should rotate with respect to the x axis and y axis respectively. yaw specify the angular speed the hunter should rotate in with respect to the z axis. The thrust specifies force the hunter should generate, These four elements are normalized with value range [-1,1]

targetcmd

The control commends for the target. It is a list of length 4 in the form of [row, pitch, yaw, thrust]. The meaning of these four elements are the same as the huntercmd mentioned above. targetcmd is by default set to be none, which corresponds to the target being static.

Return

no return

Description

simrun is the interface to control the hunter drone and target drone. The length of the simulation step is also set by this interface. Please notice the different meaning of roll, pitch and yaw.

siminfo

python specification

siminfo ()

Parameters

no parameter.

Return

pos_hunter

the position of the hunter. It is a list of length 3 in form of: [x,y,z], which represents the 3d coordinates of the hunter position

ori_hunter

the orientation of the hunter. It is a list of length 3 in form of [row, pitch, yaw], which represents the rotation w.r.t x,y,z axis respectively. Its unit is degree.

acc_hunter

the acceleration of the hunter. It is a list of length 3, which represents the 3d coordinates of the hunter acceleration.

pos_target

the position of the target. It is a list of length 3 in form of: [x,y,z], which represents the 3d coordinates of the target position

ori_target

the orientation of the target. It is a list of length 3 in form of [row, pitch, yaw], which represents the rotation w.r.t x,y,z axis respectively. Its unit is degree.

acc_target

the acceleration of the target. It is a list of length 3, which represents the 3d coordinates of the target acceleration.

thrust

The thrust force the hunter in m/s^2. It is a float scalar.

speed_hunter

the velocity of the hunter. It is a list of length 3, which represents the 3d coordinates of the hunter velocity.

speed_target

the velocity of the target. It is a list of length 3, which represents the 3d coordinates of the target velocity.

Description

siminfo is the interface to access the simulated data.

simstop

python specification

simstop()

Parameters

no parameter.

Return

no return.

Description

simstop is the interface to stop simulation. It must be called to release the resource occupied by the simulator when the simulation is supposed to stop.

installcamera

python specification

installcamera(installori, F, H, FOVnear, FOVfar, SCR_width, SCR_height)

Parameters

installori

the install orientation of the on-drone camera. It is a list of length 3 in form of [roll, pitch, yaw] each of which represent the rotation with respect to an axis. These three elements are in degree ranging from -180° - 180°. Shown as the figure 3, to make the camera point the positive direction of x axis, the camera should be rotated with respect to the z axis for 180° (yaw for 180°). In this case, the installori should be set as [0,0,180]

F

the vertical angle of the camera view cone, depicted by figure 4. Its unit is degree

H

the horizontal angle of the camera view cone, depicted by figure 4. Its unit is degree

FOVnear

the nearest plane of the camera view cone, depicted by figure 4. Its unit is degree

FOVfar

the furthest plane of the camera view cone, depicted by figure 4. Its unit is degree

SCR_width

the width of the image. It should be float

SCR_height

the height of the image. It should be float

return

no return

Description

installcamera is used to initialize the projection function. Through the interface, the orientation and view cone of the on-drone camera can be adjusted.

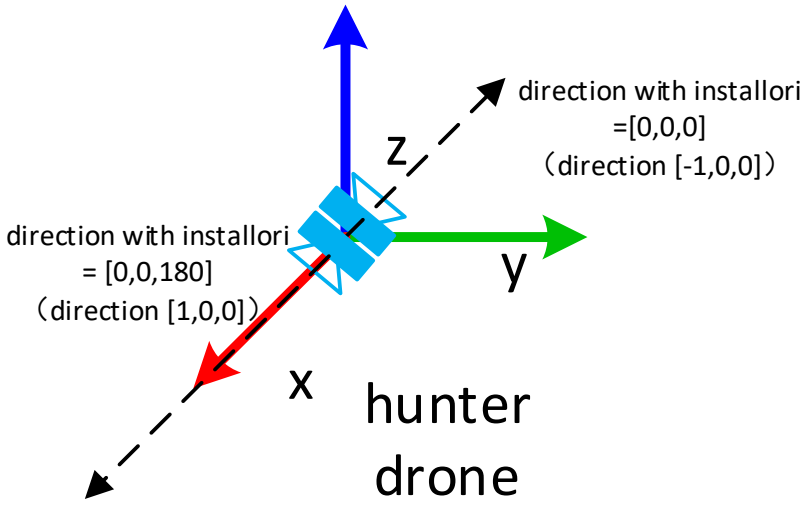

Figure 3 the illustration for installori

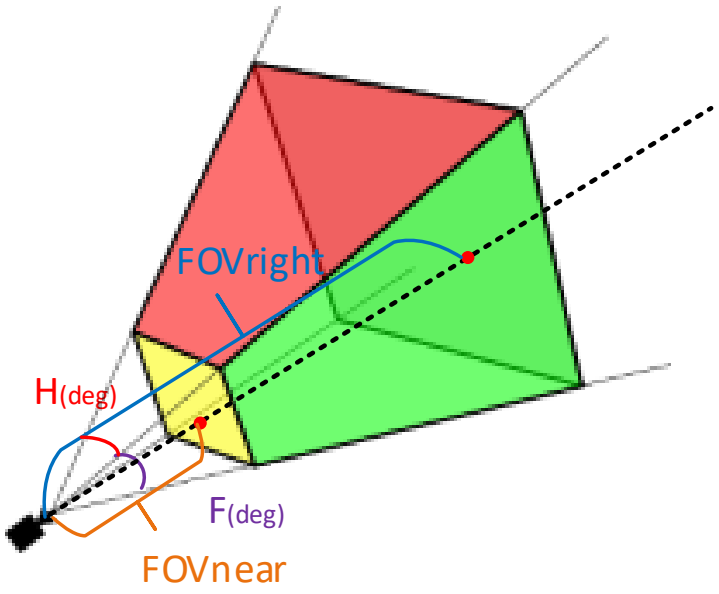

Figure 4 the illustration for view cone

projection

python specification

projection(pos_hunter, ori_hunter, pos_target)

Parameters

pos_hunter

the position of the hunter. It is a list of length 3 in form of: [x,y,z], which represents the 3d coordinates of the hunter position

ori_hunter

the orientation of the hunter. It is a list of length 3 in form of [row, pitch, yaw], which represents the rotation w.r.t x,y,z axis respectively. Its unit is degree

pos_target

the position of the target. It is a list of length 3 in form of: [x,y,z], which represents the 3d coordinates of the target position

pos_target

the position of the target. It is a list of length 3 in form of: [x,y,z], which represents the 3d coordinates of the target position

Return

u, v, inscrean

u,v are the coordinates of the target position (target_pos mentioned in the parameters) in the image coordinate system. inscreen is bool value. It will be false if the coordinates u, v is none or inf or the pseudo depth of the projected point is not in the range of [0,1](which means the target is out of the view cone of the camera). Otherwise, inscreen will be true.

projectedarea

The area of projection of the target in the image.

Description

projection function is used to compute the projection of target in the image. Its return includes the projection coordinates of the geometry center of the target and the projection area. A flag is also returned to indicate the nan or inf return.

Tutorial

1. How to deploy the simulator

The Venom Simulator is developed in C++ and OpenGL. It was designed for Ubuntu platform and depends on Eigen and glm. To use the rendering version, another dependence glfw3 should be installed.

**note: To make sure the simulator run correctly, you must put drone_sim.so and simpledrone.obj in the same direction of dronesim.py

To deploy the simulator, first download the source code from:

https://github.com/DoubleBiao/VenomSim

Then go to the path /src and type in

```
$ make
```

If the compiling succeeds, the deployment is finished! The compiling built a shared library called drone_sim.so, you can find it in /build. The python interface is provided in file dronesim.py

2. How to run the simulator via python interface

To run the simulation, a simulation session should be initialized by calling siminit(). The simulation is run in the created session steps by steps by calling simrun(). When the simulation is supposed to stop, simstop() should be called to delete the session. The code used by this tutorial can be found in /build/test.py. To make sure the simulator run correctly, you must put drone_sim.so and simpledrone.obj in the same direction of dronesim.py

firstly, to use the simulator in python, we need import the interface wrapper:

```
from dronesim import *
```

then create a new simulation session, set the initial posture of the hunter and the target and speed upbound and yawing speed upbound of two drones:

```
siminit([1,2,0],[0,0,180],[4,6,0],[0,0,0],5,10,180,180)
```

So far, the simulation session is created with the hunter locates at [1,2,0], the target locates at [4,6,0]

To visualize the simulation, we can create a visualization class:

```
renderer = visualdrone()
```

Then we can run the simulation for a step by calling:

```
simrun(5000000,[0,0,0,1])
```

This line of code commends the hunter to control its propellers to generate the largest thrust force it can get to move upwards, while keep the target static. The simulation time span is 5000,000 nanoseconds. Then we can fetch the simulated data of concern:

```
pos_hunter,ori_hunter,acc_hunter,pos_target,ori_target,acc_tar
                get,thrust,spd_hunter,spd_target = siminfo()
```

To visualize the movement, type in:

```
renderer.render(pos_hunter,ori_hunter,pos_target,ori_
target)
```

You can repeat these lines of code to simulate the movement in a long period. At the end of simulation, do remember to delete the session before you close the python:

```
dronesimapi.simstop()
```

3. How to use the projection function

The projection function separated with the simulator. To use it, we should firstly call the installcamera() to initialize the function. To compute the projection of the target we can call projection(). The code for this tutorial can be found in /build/projectiontest.py. To make sure the simulator run correctly, you must put drone_sim.so and simpledrone.obj in the same direction of dronesim.py.
Firstly, import the interface into python:

```
from dronesim import *
```

Before using the projection function, we should initialize it:

```
installcamera([45,0,90],80,80, 0.01, 100.0,400.0,300.0)
```

This line of code let the camera module load the drone model from the simpledrone.obj and initialize the direction and view cone of the camera. Then we can call the projection function to compute the projection by sending the posture of the target and hunter:

```
u,w,area,inscreen=
```

```
projection(hunterpos,hunterori,targetpos,targetori)
```