

AI PRoject

TORCS



JAHANZAIB MANZOOR PALH

TORCS (Logic Manual)

Creating an Artificial Intelligent Bot was very fruitful in learning and developing an interest for Artificial intelligence. We observed the working of the concepts that were taught to us in the class. We Made sure to understand the concepts of Artificial Intelligent and implement them as far as we could.

Major Events that we took care of were taking live data from Sensors I.e clutch, changing gear, accelerate, brake, steering the wheel and detecting immediate obstacles. We changed the top speed manually changed values in the time after the next gear is shifted. We take live values from the location sensors and detect how close we are to upcoming obstacles and apply brakes afterwards we take turns while making sure our car stays in the middle of the car.

At the turns brakes are applied while making sure the fact that the car remains in the middle of the track and doesn’t go off track until unless in the worst of case.

We tweaked the following constants for improved results in the race.

turnPrecautionDistance=45;

maxSpeedTurn=160;

maxSpeedStraightRoad=173;

turnSpeedConstant = (float)1.35;

brakeConst = (float) 1.5;

accConst = (float) 1.5;

// if car is pointing in the correct direction we increase to the next gear and steer

if (sensors.getAngleToTrackAxis()\*sensors.getTrackPosition()>0)

{

gear = 1;

steer = -steer;

}

clutch = clutching(sensors, clutch);

To make sure that initially the car accelerates to its full potential we changed the clutch a bit later and reduced the time in the case of higher gears.

// Adjust the current value of the clutch

if(clutch > 0)

{

double delta = clutchDelta;

if (sensors.getGear() < 2)

{

// Apply a stronger clutch output when the gear is one and the race is just started

delta /= 2;

maxClutch \*= clutchMaxModifier;

if (sensors.getCurrentLapTime() < clutchMaxTime)

clutch = maxClutch;

}

We steer the car w.r.t the track and adjust the next position of the car w.r.t to the middle of the track to make sure our bot doesn’t lose control of the car at higher speeds we reduced the turning angels at higher speeds.

private float getSteer(SensorModel sensors){

// steering angle is compute by correcting the actual car angle w.r.t. to track

// axis [sensors.getAngle()] and to adjust car position w.r.t to middle of track [sensors.getTrackPos()\*0.5]

float targetAngle=(float) (sensors.getAngleToTrackAxis()-sensors.getTrackPosition()\*0.5);

// at high speed reduce the steering command to avoid loosing the control

if (sensors.getSpeed() > steerSensitivityOffset)

return (float) (targetAngle/(steerLock\*(sensors.getSpeed()-steerSensitivityOffset)\*wheelSensitivityCoeff));

else

return (targetAngle)/steerLock;

}

While accelerating the car we make sure that the car is currently on track if it is on track accelerate it else calculate the distance from the middle of the track and reduce gears until we reach at the middle of the track and then accelerate it .

private float getAccel(SensorModel sensors)

{

// checks if car is out of track

if (sensors.getTrackPosition() < 1 && sensors.getTrackPosition() > -1)

{

// reading of sensor at +5 degree w.r.t. car axis

float rxSensor=(float) sensors.getTrackEdgeSensors()[10];

// reading of sensor parallel to car axis

float sensorsensor=(float) sensors.getTrackEdgeSensors()[9];

// reading of sensor at -5 degree w.r.t. car axis

float sxSensor=(float) sensors.getTrackEdgeSensors()[8];

float targetSpeed;

// track is straight and enough far from a turn so goes to max speed

//if (sensorsensor>maxSpeedDist || (sensorsensor>=rxSensor && sensorsensor >= sxSensor))

//targetSpeed = maxSpeed;

if (sensorsensor>maxSpeedDist || (sensorsensor>=rxSensor && sensorsensor >= sxSensor))

{

targetSpeed = maxSpeed2;

//System.out.println("IF"); //check

}

else

{

float mlp = currentConfig.turnSpeedConstant;

//System.out.println("ELSE"); //check

// approaching a turn on right

if(rxSensor>sxSensor)

{

// computing approximately the "angle" of turn

float h = sensorsensor\*sin5;

float b = rxSensor - sensorsensor\*cos5;

float sinAngle = b\*b/(h\*h+b\*b);

// estimate the target speed depending on turn and on how close it is

targetSpeed = maxSpeed\*(sensorsensor\*sinAngle/maxSpeedDist)\*mlp;

}

// approaching a turn on left

else

{

// computing approximately the "angle" of turn

float h = sensorsensor\*sin5;

float b = sxSensor - sensorsensor\*cos5;

float sinAngle = b\*b/(h\*h+b\*b);

// estimate the target speed depending on turn and on how close it is

targetSpeed = maxSpeed\*(sensorsensor\*sinAngle/maxSpeedDist)\*mlp;

}

}