RCSnail - *Interactive Racing*

RCSnail is a high tech educational and entertainment solution where small RC (Remote Control) cars are driven on a race track with steering wheels and pedals via video feed. For the players the experience is similar to driving the real race car. At the moment there is one race track shopping centre Eeden in Tartu where players can drive the cars on site. In development is a solution where the players can use their mobile phones to drive the cars over the internet from anywhere in the world. The same technology can be used by AI to drive the cars. Human drivers and AI can have the same input in form of video and same outputs to drive the cars. The goal is to use different AI algorithms to compete with each other and eventually compete safely with human drivers. For external use by AI an open-source Python based SDK is provided.

Technology used

RCSnail is using standard WebRTC real-time protocol to send the low latency video feed from the cars to drivers and drivers commands to cars and sensor/telemetry data back to drivers. Additional servers in the cloud are used to establish the communication between the car and drivers and save the video feed recordings with driving and telemetry data for later analysis and AI training.

Manual driving for collecting the data

For manual driving in Windows

1. download https://gstreamer.freedesktop.org/download/ GStreamer 64bit Mingw version 1.18 runtime. Direct link for installer https://gstreamer.freedesktop.org/data/pkg/windows/1.18.0/mingw/gstreamer-1.0-mingw-x86_64-1.18.0.msi
2. Make sure that Windows PATH contains “C:\gstreamer\1.0\mingw_x86_64\bin”
3. Download the RCSnail Windows application from https://drive.google.com/file/d/1-pmJ4Cjv6OYoSaxXsuekWhnriwy-3B9a/view?usp=sharing
4. Extract the RCSAgent.zip to some folder with write permissions
5. Create an account if not created already https://api.rcsnail.com/
6. Start the RCSAgent.exe and Sign in with created account
7. In RCSAgent.exe enter “deltax” as a track and “deltax_i8_01” or “deltax_i8_02” as a car

8. Press “New session” button and wait a little until session is established and video becomes visible
9. Microsoft XBox or any other XInput compatible gamepad can be used to control the car. https://en.wikipedia.org/wiki/Xbox_One_controller
   Keys are assigned as:
   a. LT - brakes
   b. RT - throttle
   c. LB - holding it down activates the reverse gear R
   d. any analog stick acts as a steering input

10. Optional. Recording of the video and the log file about the telemetry is stored in sub folder “rec”. There will be one raw video file with extension .h264 and log file with extension .json. This will save the data in format that is moved over the internet
11. ☑ ZMQ Enables ZMQ communication with Python application
12. ☐ AI Enables AI/Python application to send the driving commands to the car. If this is unchecked then RCSAgent sends gamepad commands to the car. This option can be checked all the time if similar functionality is implemented in the Python application.
13. For Python AI handling use https://github.com/martinliivak/RCSnail-AI-lite This will save expert driver commands and car telemetry together with video

Driver protocol

Driver command JSON objects are sent periodically with short 15-20 millisecond intervals via WebRTC datachannel: {"p":16565,"c":1593708369816,"g":0,"s":-0.08,"t":0.2703,"b":0}

Field	Description
p	Packet number. Can be used to detect missing packets.
c	Driver timestamp in milliseconds since the Unix Epoch
g	Gear: -1 - reverse; 0 - neutral; 1- forward
s	Steering in range -1.0...1.0
t	Throttle: 0.0...1.0
b	Braking: 0.0...1.0

Car and telemetry protocol

Car uses JSON objects to send its telemetry and race related information and events back via the same datachannel. There are different messages based on event type. Most frequent message is cars telemetry message:
{"t":"A","sp":587,"sc":1593708372793,"c":1593708369816,"sd":7389,"cs":0.0,"cg":0,"ct":0.0,"cb":0.0,"b":3549,"sa":374}

Field	Description
t	Message type. A - Driving command answer.
sp	Server sent packet number. Can be used to detect missing packets.
sc	Server timestamp in milliseconds since the Unix Epoch
c	Last received driver timestamp sent back. Can be used to calculate the message round trip.
sd	Server delay in microseconds. Time it takes to pass driver command between server and car. Can be used to calculate detailed message round trip.
cg	Car engaged gear: -1 - reverse; 0 - neutral; 1- forward
cs	Car’s last steering command in range -1.0...1.0
ct	Car’s last throttle value in range 0.0...1.0
cb	Car’s last braking value: 0.0...1.0
b	Car battery voltage in mV units
sa	Car steering position sensor raw value. Depends from the car and can be in range 0..1023

Log format

Log files are saved in JSON file. Whole file is one JSON array where each element is command and telemetry JSON packet with additional timestamps and metadata.

VSCode Anaconda Python

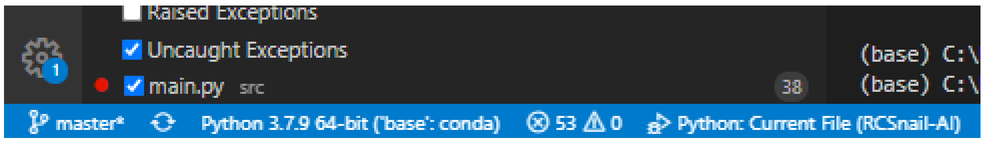
Sources

git clone https://github.com/martinliivak/RCSnail-Connector.git
git clone https://github.com/martinliivak/RCSnail-AI-lite.git
mkdir training

Anaconda

Download Miniconda (lighter version of Anaconda)
https://docs.conda.io/en/latest/miniconda.html

Open RCSnail-AI-lite folder with VSCode
Make sure conda Python interpreter is activated in VSCode:



☐ Raised Exceptions
☑ Uncaught Exceptions
● ☑ main.py src 38

```
(base) C:\
(base) C:\
```

master* Python 3.7.9 64-bit ('base': conda) ⊗ 53 ⚠ 0 Python: Current File (RCSnail-AI)

In VSCode terminal prompt:
# activate Python 3.7.9 as Keras is not compatible with Python 3.8:
# conda search python
conda install python=3.7.9

conda install -c anaconda numpy
conda install -c anaconda opencv
conda install -c anaconda pillow
conda install -c anaconda keras

conda install -c anaconda pyzmq
conda install -c anaconda ruamel.yaml
conda install -c anaconda scikit-learn
conda install -c anaconda pandas

# for installing install local package
conda install -c anaconda conda-build

# use local packages like src
conda develop .

# use common packages

```
cd ..\RCSnail-Commons
conda develop .
```

VSCode launch.json configuration

```
        {
            "name": "Python: Current File",
            "type": "python",
            "request": "launch",
            "cwd": "${workspaceFolder}/src/",
            "program": "${workspaceFolder}/src/main.py",
            "console": "integratedTerminal"
        }
```