FACSvatar Documentation

Release 0.3.2-alpha

Stef van der Struijk

Jan 13, 2019

Contents:

Orchestrate modules to enable OS and software independent interactive facial animation.

FACSvatar is a combination of the terms **FACS** and **Avatar**. FACS stands for Facial Action Coding System and is a way to describe a person's facial expression in terms of muscle contraction and relaxation. A single muscle group is called an Action Unit (AU) and either has a value between 0-5 or 0-1 (FACSvatar). It is a system found by Paul Ekman and is often used in affective research. Since FACS describes facial configurations, it is not only useful for affective analysis, but also for animation. More information about FACS can be found here: https://en.wikipedia.org/wiki/Facial_Action_Coding_System

FACSvatar is a modular framework that connects different software over ZeroMQ to enable both the animation and analysis of AUs. At present, a standard workflow is to use a modified OpenFace. (FACS input) that is transported and manipulated (smoothed) through FACSvatar and then forwarded to either Unity3D (real-time) or Blender (high-quality) animation.

CHAPTER 1

Links

- **GitHub** code and a longer explanation of FACSvatar's usefulness can be found at: https://github.com/NumesSanguis/FACSvatar
- Homepage: https://surafusoft.eu/facsvatar/
- Discussion of FACSvatar on reddit (for code problems, use GitHub issue tracker): https://www.reddit.com/r/FACSvatar/

1.1 Quickstart

See *Default set-up* for FACSvatar setup without Docker.

FACSvatar is tested on Ubuntu and Windows, but should work on MacOS as well.

Please see the FACSvatar GitHub page for the Quickstart instructions.

1.1.1 Quickstart video

See the quickstart video: SOON

1.1.2 Docker advanced

For more advanced Docker commands, and e.g. how to use your own .csv files created with OpenFace, see: *Docker*

1.2 Default set-up

Probably you want to either run FACSvatar in real-time mode for interactive purposes or create high-quality facial animation/pictures.

- Real-time: Modified OpenFace –> FACSvatar modules –> Unity3D

- Offline: (Modified) OpenFace –> .csv –> FACSvatar modules –> Blender

There 3 things that have to be setup:

1. *FACS input* (Modified OpenFace)
2. *FACSvatar modules* (Python 3.5+)
3. *Animation-Visualization* (Unity3D / Blender)

If you're already done setting FACSvatar up, please head here: *First run*

but before that, let's download the FACSvatar GitHub repro with:
`git clone https://github.com/NumesSanguis/FACSvatar.git`
and download all other necessary files from the release page:
https://github.com/NumesSanguis/FACSvatar/releases

1.2.1 FACS input

Animation can be either in real-time or in offline mode. For the moment, FACSvatar is only working with OpenFace, however any module that provides FACS based data could be used.

Real-time allows for interactive systems, but at present the quality of FACS tracking from OpenFace is of lower quality in this mode. If interactivity is not your goal, the offline version is most likely a better choice.

Real-time

Note: Requires 1 Windows PC (due to ZeroMQ being integrated in the GUI)

For the real-time version of FACSvatar we need to use a modified OpenFace which includes a ZeroMQ component to stream AU, gaze and head pose data out of it into. You can either

- **Download a modified OpenFace v2.1.0**
  - All copyright of OpenFace belongs to Carnegie Mellon University. By using this software, you agree to their licensing terms found here: https://github.com/TadasBaltrusaitis/OpenFace/
- Or build the modified version yourself following these instructions:

Build OpenFace with ZeroMQ

- Download Source code OpenFace 2.1.0 from https://github.com/TadasBaltrusaitis/OpenFace/releases
- Install Visial Studio 2015 / 2017
- Run download_models.ps1 / .sh or copy `cen_patches_x.xx_of.dat` to `OpenFace\lib\local\LandmarkDetector\model\patch_experts`

1. Overwrite *MainWindow.xaml.cs* in *OpenFaceguiOpenFaceOffline* with openface/MainWindow.xaml.cs from FACSvatar GitHub
2. Open visual studios:
   - Open *OpenFace/OpenFace.sln* with Visual Studio 2015
   - Open *OpenFace_vs2017.sln* with Visual Studio 2017 (didn't work for me so far)
3. (In visual studio) Right click in "Solution Explorer" on "OpenFaceOffline" –> *Manage NuGet Packages...*

4. Browse and search for *netmq*; install NetMQ by NetMQ with version v4.0.0.1 (AsyncIO.0.1.26) note: Search under "Browse" not "Installed"
    - Don't update AsyncIO to a newer version (v0.1.40)
5. Search for *json*; Install Newtonsoft.Json by James Newton-King v11.0.2
6. Select OpenFaceOffline –> Release, x64, OpenFaceOffline –> Build –> (Re)build OpenFaceOffline
7. Copy *config.xml* from FACSvatar GitHub and put it at *OpenFacex64Releaseconfig.xml* # DON'T FORGET - otherwise crashes at startup

Offline

- (Windows GUI) Use the modified OpenFace found under the header *Real-time* or Download OpenFace_2.0.6_win_xYY.zip from: https://github.com/TadasBaltrusaitis/OpenFace/releases
- (Other) Follow instructions here: https://github.com/TadasBaltrusaitis/OpenFace/wiki#installation

1.2.2 FACSvatar modules

At present, all the core modules work with Python, so let's setup an environment. FACSvatar recommends using Anaconda for managing packages and virtual environments for Python, therefore code instructions assume Anaconda. Probably `pip install .` will do the same without problems.

This project uses the Asyncio library for asynchronous code execution, hence we use Python 3.6+ (although some modules work with Python 3.5). I wanted to keep it Python 3.5 compatible, but due to the use of asynchronous generators used in some standard modules, the default version is 3.6+.

Anaconda setup

```
conda create --name facsvatar python=3.7  # new virtual env and force python 3.x
#conda install python=3.7  # IF you already have an existing env
source activate facsvatar  # activate env (Windows: conda activate facsvatar)

conda install pyzmq  # make sure it's for py3.6
conda install pandas  # library for dataframes; used for .csv reading and JSON-to-
↪Dataframe

# Basic environment setup finished, but ipykernel setup recommended for control panel␣
↪GUI

conda install ipykernel  # allows the use of env kernels in jupyter notebook
conda install ipywidgets  # GUI elements in jupyter notebook
python -m ipykernel install --user --name facsvatar --display-name "py3 facsvatar"  #␣
↪enable our env as kernel in jupyter notebook
```

Test new environment

Go into a Python environment in your terminal with: `python` - *enter*

```
import zmq
print("Current libzmq version is %s" % zmq.zmq_version())  # 4.2.5 at time of writing
print("Current  pyzmq version is %s" % zmq.__version__)  # 17.1.2 at time of writing
```

1.2.3 Animation-Visualization

Unity3D - game engine

Recommended for real-time or game like interaction applications. Unity3D version 2018.2.10f1 recommended.

1. Download either Unity3D (single version) or UnityHub (recommended; manages Unity3D versions)
    - Windows/Mac: Download Unity(3D/Hub)
    - Linux: Download UnityHub
    - Linux: Download Unity3D
2. Open the FACSvatar project in Unity3D by navigation to `FACSvatar/unity_FACSvatar` folder in the FACSvatar GitHub repro.
3. (In the `Asset Store` tab: Search for JSON .NET for Unity (by PARENTELEMENT, LLC) and click `Download`.)
    - This step is probably not needed anymore.

Blender - open source 3D creation suite

Sorry, these instructions are still a mess.
Recommended for high-quality image/video rendering and post-modification.
Hopefully going to be real-time and as a Blender add-on when version 2.8 with EEVEE is released.

1. Download Blender
2. Download Manuel Bastioni LAB (MBLAB) add-on v.1.6.1a for Blender
    - The author of MBLAB unfortunately stopped with his project (http://www.manuelbastioni.com/)
    - However, the community is still active: https://github.com/animate1978/MB-Lab
3. Start Blender in terminal by opening a terminal in the folder `blender-2.79` and run:
    - Windows: `blender.exe`
    - Ubuntu: `./blender`
4. Import the .zip into Blender to install add-on: File –> User Preferences –> Add-ons –> Install Add-on from File –> manuelbastionilab_161a.zip –> check-mark in front of `Characters: ManuelbastioniLAB`
5. Create a model with MBLAB by clicking `Init character` (leave default options for export to Unity3D), modify and press `Finalize tools --> Finalize`
6. If Blender version is below 2.8 (likely the case if done in 2018 or earlier):
    - Create a Python 3.5 environment by following the instructions under *Anaconda setup* , but replacing `--name facsvatar python=3.7` for `--name blender python=3.5` (you can skip commands about `Jupyter Notebook`)
7. Change line 7 in `FACSvatar/blender/facsvatar_zeromq.py` to correctly point to your blender anaconda environment.
    - Windows (something like): c:\Users*you*\AppData\Local\conda\conda\envs\blender\Lib\site-packages
    - Ubuntu (something like): /home/*you*/anaconda3/envs/blender/lib/python3.5/site-packages

Enabling FACS sliders in MBLAB add-on

Copy .json files found in `FACSvatar/modules/process_facstoblend/au_json` to:

- Windows: `C:\Users\*user*\AppData\Roaming\Blender Foundation\Blender\2.79\scripts\addons\manuelbastionilab\data\expressions_comb\human_expressions\`
- Ubuntu: `/home/*user*/.config/blender/2.79/scripts/addons/manuelbastionilab/data/expressions_comb/human_expressions/`

1.3 First run

Make sure you've setup FACSvatar (real-time with Unity3D or offline Blender) by following the instructions here: *Default set-up*

- Real-time: Modified OpenFace –> FACSvatar modules –> Unity3D
- Offline: (Modified) OpenFace –> .csv –> FACSvatar modules –> Blender

1.3.1 Visualization: Unity3D - real-time

0. Have FACSvatar project opened in Unity3D
1. Press play button

1.3.2 Visualization: Blender - offline

Sorry, these instructions are still a mess. Look at the Blender tutorial video for clearer instructions.

0. Do these steps before running other FACSvatar modules.
1. Copy the first 2 lines of code found in `FACSvatar/blender/facsvatar_zeromq.py` into the Blender terminal (but change path to match your system's path to that file).
   - Windows: Change `/` to `\\`
2. Your Blender now freezes and waits for data from Blend Shapes module. (You can safely send 1x a cancel command in terminal that runs Blender to cancel listening for data).
3. Yes, I know, the freezing part is bad. This, and all these instructions, should be gone once the Blender component of FACSvatar is turned into a Blender add-on.
4. Data is streamed into the timeline of Blender and can be modified following the normal workflow of Blender.

1.3.3 FACSvatar modules

Open 3 terminals

1. Start bridge module
   (a) `source activate facsvatar`
   (b) `cd *your_path*/FACSvatar/modules/process_bridge`
   (c) `python main.py`
2. Start FACS to Blend Shapes module

   (a) `source activate facsvatar`

   (b) `cd *your_path*/FACSvatar/modules/process_facstoblend`

   (c) `python main.py`

3. Start OpenFace offline module (input)

   (a) `source activate facsvatar`

   (b) `cd *your_path*/FACSvatar/modules/input_facsfromcsv`

   (c) `python main.py`

Congratulations, you should now see an avatar in Unity3D / Blender move its head, eyes and face!

If you want to use your own FACS values extracted from a video do:

1. `OpenFaceOffline.exe` –> menu: `Recording settings` –> `Output location`
2. `OpenFaceOffline.exe` –> menu: `File` –> `Open Video`.
3. Copy .csv from your output location to *your_path*/FACSvatar/modules/input_facsfromcsv/*some_folder*
4. Run OpenFace offline module with: `python main.py --csv_folder some_folder --csv_arg -1`

1.3.4 Real-time (needs Windows PC)

You want to use your own facial expressions in real-time you say? Skip starting the OpenFace offline FACSvatar module and instead start the modified OpenFace GUI: `OpenFaceOffline.exe` –> menu: `File` –> `Open Webcam`.

Warning, pretty heavy on the CPU of the PC.

1.3.5 Next

Every Python module has a help function, giving more information on what arguments you can give. Type: `python module_x.py --help` You can for example run modules on different PCs by changing the IP where the module is subscribed to for data with: `python module_x.py --sub_ip 192.168.xxx.xxx`

Coming soon: Advanced instructions

For importing models to Unity3D, follow the instructions here: http://manuelbastionilab.wikia.com/wiki/Unity_3D

1.4 Docker

Advanced Docker commands

1.4.1 Docker install

It is best to follow the official Docker instructions for installation.

Ubuntu

Direct links:

- Docker install: https://docs.docker.com/install/linux/docker-ce/ubuntu/
- Docker compose: https://docs.docker.com/compose/install/
- https://docs.docker.com/install/linux/linux-postinstall/
- https://docs.docker.com/compose/completion/

Warning: Check first what every command does before executing. Commands might be outdated.

Add current user to group docker on Linux systems. This prevents the need of typing `sudo` before every docker command.

```
sudo usermod -a -G docker $USER
```

Docker machine (ip):

```
base=https://github.com/docker/machine/releases/download/v0.16.0 &&
  curl -L $base/docker-machine-$(uname -s)-$(uname -m) >/tmp/docker-machine &&
  sudo install /tmp/docker-machine /usr/local/bin/docker-machine
```

Docker machine bash completion:

```
base=https://raw.githubusercontent.com/docker/machine/v0.16.0
  for i in docker-machine-prompt.bash docker-machine-wrapper.bash docker-machine.bash
  do
    sudo wget "$base/contrib/completion/bash/${i}" -P /etc/bash_completion.d
  done
```

1.4.2 Docker own OpenFace .csv (untested)

If you want to use your own OpenFace .csv, you've obtained by running some video through OpenFace, in combination with Do

- Copy file from host to container: `docker cp foo.csv facsvatar_facsfromcsv:/openface/*your_folder/foo.csv`
- Go inside Docker container: `docker-compose exec facsvatar_facsfromcsv bash`
- Inside container: python main.py –pub_ip facsvatar_bridge –csv_folder openface/*your_folder* –csv_arg -1

1.4.3 Useful Docker commands

```
sudo usermod -a -G docker $USER  # add current user to group docker on Linux systems␣
↪(Ubuntu)

docker build . -t foo/bar  # build docker image
docker run -it foo/bar  # run build docker image and enter interactive mode
docker run -p 5550:5550 -it foo/bar  # same as above with mapping Docker port to host
docker-compose up  # run docker-compose.yml
docker-compose build / docker-compose up --build  # rebuild images in docker-compose.
↪yml
```

(continues on next page)

(continued from previous page)

```
docker image ls  # show docker images
docker container ls  # show docker containers
docker exec -it pyzmq-docker_pub_1  # enter bash in container
docker attach pyzmq-docker_sub_1  # get
docker inspect -f '{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' pyzmq-
↪docker_sub_1  # get ip of container

To detach the tty without exiting the shell, use the escape sequence Ctrl+p + Ctrl+q

docker stop $(docker ps -a -q)  # stop all CONTAINERS
docker rm $(docker ps -a -q)  # remove all CONTAINERS
docker rm $(docker images -q)  # remove all IMAGES
```

1.5 Deep learning setup

WARNING: These instructions might be slightly outdated.
Instructions related to module `process_facsdnnfacs`.

We're going to add the deep learning library requirements *TensorFlow* & *Keras* to our *facsvatar* anaconda environment.

Currently tested with Ubuntu 16.04 (not yet on Windows, but instructions provided)

If you didn't setup your Python environment yet, look here: *Default set-up*

Make sure your terminal has *facsvatar active*:

```
source/conda activate facsvatar  # Ubuntu: `source`, Windows `conda`
```

1.5.1 Dependencies

- Python 3.4+ (tested 3.6)
  - TensorFlow 1.7.0
    - CUDA Toolkit v9.0 # GPU training
    - cuDNN v7.1.3 # GPU training
  - Keras (TensorFlow backend)

1.5.2 Anaconda install all - untested

Make sure your terminal has *facsvatar active*.

```
source/conda activate facsvatar  # Ubuntu: `source`, Windows `conda`

# Keras
conda install -c anaconda keras-gpu
```

1.5.3 TensorFlow - Manual

Instructions are based on the Anaconda instructions found here: https://www.tensorflow.org/install/ , so look there for the most recent instructions.

You can skip the GPU sections if you want to run it on a CPU ((much) slower). Untested for now on Windows.

GPU

CUDA Toolkit v9.0

- Official instructions (Ubuntu): https://docs.nvidia.com/cuda/cuda-installation-guide-linux/#axzz4VZnqTJ2A
- Official instructions (Windows): https://docs.nvidia.com/cuda/cuda-installation-guide-microsoft-windows/

Short instructions:

1. Go to: https://developer.nvidia.com/cuda-90-download-archive
2. Select .deb / .exe for your system
3. Download file (and Ubuntu: open terminal at download location)
4. Follow instructions
    - Ubuntu: Do step 1 for all patches before step 2 (*sudo dpkg -i cuda-xxx-update-xxx.deb*)
    - Ubuntu: If this fails, install 'GDebi Package Installer' from Ubuntu Software and open '.deb' with that.

cuDNN v7.1.3

1. Go to: https://developer.nvidia.com/cudnn –> DOWNLOAD cuDNN –> Join / Login
2. Download cuDNN v7.1.3 (April 17, 2018) (or newer?), for CUDA 9.0
    - Ubuntu: cuDNN v7.1.3 Runtime Library for Ubuntu16.04 (Deb)
    - Ubuntu: cuDNN v7.1.3 Developer Library for Ubuntu16.04 (Deb)
    - Windows: cuDNN v7.1.3 Library for Windows 7/10
3. Install by running in terminal: - Ubuntu: *sudo dpkg -i libcudnn7_7.1.3.16-1+cuda9.0_amd64.deb* - Ubuntu: *sudo dpkg -i libcudnn7-dev_7.1.3.16-1+cuda9.0_amd64.deb* - Windows:
4. Setup your environment variable to link to cuDNN
    - Ubuntu (in terminal): *echo 'export PATH=/usr/local/cuda-9.0/bin${PATH:+:${PATH}}' >> ~/.bashrc*
        - Manually: Add above line to .bashrc (in your home directory, ctrl+h to show hidden files)
    - Windows: add the directory where you installed the cuDNN DLL to your %PATH% environment variable.

NVIDIA CUDA Profile Tools Interface (Ubuntu only) - untested

https://github.com/tensorflow/tensorflow/issues/16214

1. Locate cuda-command-line-tools: *sudo apt-cache search cuda-command-line-tools-9-0*
2. Install: *sudo apt install cuda-command-line-tools-9-0*
3. Path to environment variable: *echo 'export LD_LIBRARY_PATH=${LD_LIBRARY_PATH:+${LD_LIBRARY_PATH}:}/usr/local/c*
   >> ~/.bashrc

Install TensorFlow with Anaconda (GPU/CPU)

If you didn't setup your Python environment yet, look here: *Default set-up*

Make sure your terminal has *facsvatar active*:

```
source/conda activate facsvatar  # Ubuntu: `source`, Windows `conda`

# GPU - Python 3.6
pip install --ignore-installed --upgrade \
https://storage.googleapis.com/tensorflow/linux/gpu/tensorflow_gpu-1.7.0-cp36-cp36m-
↪linux_x86_64.whl

# CPU - Python 3.6
pip install --ignore-installed --upgrade \
https://storage.googleapis.com/tensorflow/linux/cpu/tensorflow-1.7.0-cp36-cp36m-linux_
↪x86_64.whl

# test installation
python
>>> import tensorflow as tf  # no error
>>> tf.__version__  # 1.7.0
>>> ctrl+z / ctrl+Break  # leave Python; z: Ubuntu, Break: Windows
```

1.5.4 Keras - Manual

Official instructions: https://keras.io/

Make sure your terminal has *facsvatar active*.

```
source/conda activate facsvatar  # Ubuntu: `source`, Windows `conda`

# Keras
pip install keras

# Only do the following commands if Keras doesn't use GPU
pip uninstall keras  # Remove only Keras, but keep dependencies
pip install --upgrade --no-deps keras  # and install it again without dependencies
```

Test Keras GPU

```
cd jupyter_notebooks  # FACSvatar folder containing Jupyter notebooks
jupyter notebook  # starts jupyter notebook and opens browser page
```

1. Click Keras_GPU_test.ipynb
2. Check right-top shows "py3 facsvatar" (our python env)
3. Kernel –> Restart & Run All
4. If you can find a *device_type: "GPU"*, Keras should be using GPU
5. Congratulations, Deep Learning setup complete!

1.6 Contribute

The main idea is that researchers/developers write their own micro modules with ZeroMQ and wrap them up as Docker containers. Docker will make the modules OS independent and a default ZeroMQ interface let's these modules communicate with each other. The ideal case is where a researcher/developer uploads a module, an other user downloads it, and he/she only has to add a few lines to `docker-compose.yml` to start using it.

- Image that you have created a better FACS value extracter. You only have to add a ZeroMQ component, and no modification is needed in the other modules or visualization. or
- You want to enable facial animation in Unreal. You only need to include a ZeroMQ subscriber component to your Unreal project and you can use the other modules as is.

By creating a repository of FACSvatar compatible modules, we can make more advanced Human-Agent Interaction (HAI) systems and don't have to start a project from scratch if we only want to focus on 1 component.

1.7 License info

FACSvatar integrates many projects, which can contain different licenses. Everything in this repro that has been created by the project owner(s) of FACSvatar are under the GNU LGPLv3 where not conflicting with other licenses. Depending on which modules/parts you use/modify of this framework, different licening conditions may apply.

If you make your modifications to the code and release it open source under a (L)GPL license, you should not have to worry.

The project owner(s) are not responsible for any missing / incorrect licensing information described here. Please confirm by yourself and with a lawyer.

Here is a list of licenses we have currently identified in our project:

- Data generated by OpenFace (FACSvatar/modules/01_facs-from-csv/*.csv): ACADEMIC OR NON-PROFIT ORGANIZATION NONCOMMERCIAL RESEARCH USE ONLY
  - https://github.com/TadasBaltrusaitis/OpenFace
  - Commercial license: contact innovation@cmu.edu
- Manuel Bastioni Lab:
  - Models created: CC Attribution 4.0 International or AGPLv3 (you can choose)
  - Python code: GPLv3
  - Other: AGPLv3
  - http://www.manuelbastioni.com/guide_license.php
- UnityMainThreadDispatcher: Apache License v2.0 (Pim de Witte)

If there is anything unclear about licensing, please ask here: https://www.reddit.com/r/FACSvatar/

CHAPTER 2

Indices and tables

- genindex
- modindex