README file for Part III Project

Audio-driven upper-body motion synthesis on a humanoid robot

Jan Ondras (jo356@cam.ac.uk)
Trinity College, University of Cambridge

June 1, 2018

Contents

1 Overview

In my Part III Project I developed a novel automatic audio-driven upper-body motion synthesis (AUMS) system targeted to the humanoid robot Pepper. The system takes audio input from its user and uses the trained neural network to predict time-series of angles between upper-body joints that are used to control the robot upper-body pose. The simplified operation of the system is illustrated in Figure 1.

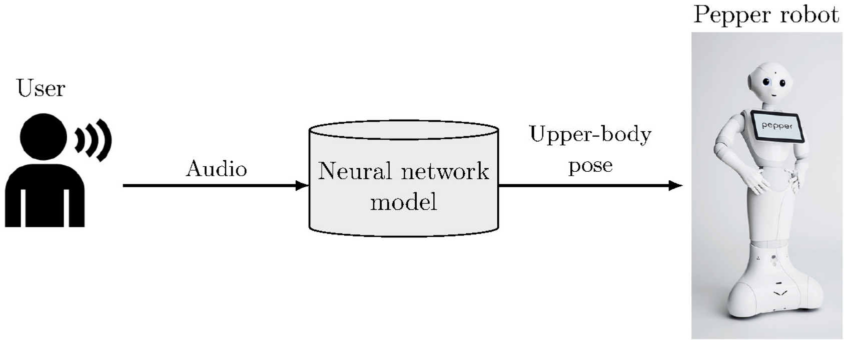

Figure 1: Simplified operation of the audio-driven upper-body motion synthesis system. Left and right pictures were taken from [1] and [2] respectively.

The system supports two synthesis modes:

- **Offline synthesis:** the whole audio input must be provided upfront, and the prediction and synthesis of movements are performed at any later time.
- **Online (real-time) synthesis:** the movements are predicted and synthesised on-the-fly while the input audio is being captured.

The AUMS system was developed in Python 2.7 and all source code files are in a form of Jupyter Notebooks [3] (.ipynb file extension).

2 Running the system

To run the AUMS system, the following requirements have to be met:

- SoftBank Robotics Python SDK and NaoQi framework (version 2.5.5) [2]
- Keras machine learning library [4]
- (only for synthesis on real robot) Pepper robot (version 1.7 and body type V16) by SoftBank Robotics [2]
- (only for synthesis on virtual robot) Choregraphe environment [5]
- (only for online synthesis) Microphone
- (only for online synthesis) *pyaudio* library [6]
- (only for online synthesis) Kalman filter module in `./SourceCode/KFClass.py` [7]

To run the system in

- *offline synthesis* mode use `./SourceCode/OfflineSynthesis.ipynb`
- *online synthesis* mode use `./SourceCode/OnlineSynthesis.ipynb`

Further instructions and settings are provided at the beginning of these files.

3 Project files

There are three groups of project files described in the following sections.

3.1 Source code

The source code files reside in the folder `./SourceCode` and are organised according to the system development and evaluation stages as follows. The detailed system diagram shown in Figure 2 might aid understanding.

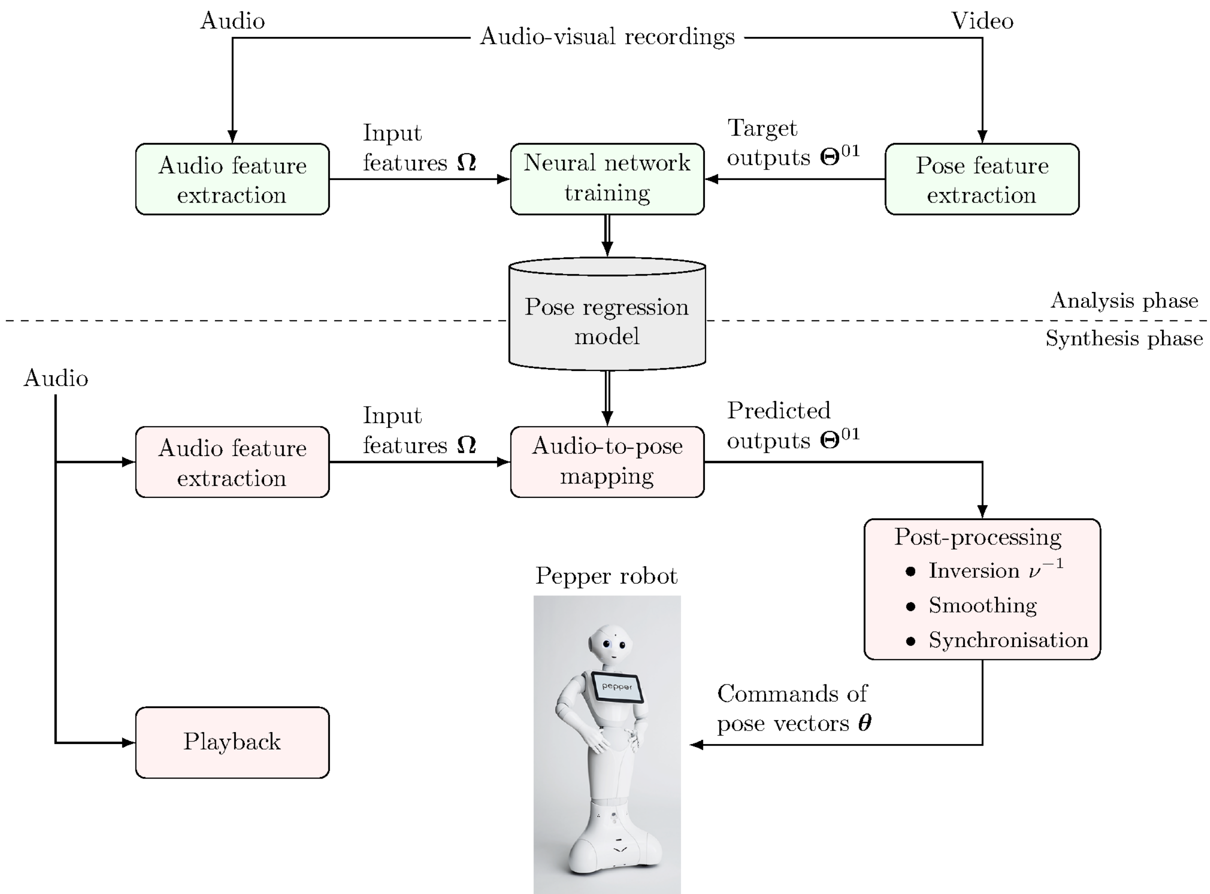

Figure 2: Diagram of the audio-driven upper-body motion synthesis system.

Dataset:

- `Data_analysis.ipynb` – data distributions (video durations, number of frames) and elimination of the outlier subject 19.
- `Dataset_split.ipynb` – dataset split into train/val/test partitions for both SI and SD model development.
- `Segment_intoSequences_SI.ipynb` – generate sequences (of $N_\tau = 300$ time-steps) for LSTM-SI models.
- `Segment_intoSequences_SD.ipynb` – generate sequences (of $N_\tau = 300$ time-steps) for LSTM-SD models.

Audio feature extraction:

- `Extract_audio_features.ipynb` – extract four types of audio features (MFCC-13, LogFB-26, LogFB-52, LogFB-78).
- `Overall_znorm.ipynb` – calculate z-normalisation parameters (mean, std) for audio features over the whole dataset (used for prediction on new subjects in online synthesis mode).

Pose feature extraction:

3D skeleton reconstructions:

  - `Render_labeled_3D_skeletons.ipynb` – render 3D skeletons (with labeled body joints) reconstructed by each of the four 3D pose estimation methods (OP+M, OP+A, LFTD, VNect).
  - `OP+M_3D_render.ipynb` – simulation of 3D pose reconstructed by OP+M.
  - `OP+A_3D_render.ipynb` – simulation of 3D pose reconstructed by OP+A.
  - `LFTD_3D_render.ipynb` – simulation of 3D pose reconstructed by LFTD.
  - `VNect_3D_render.ipynb` – simulation of 3D pose reconstructed by VNect.

- `QuantCompare_2poseExtractionMethods.ipynb` – quantitatively compare the OP+A with the LSFT pose feature extraction method.
- `Extract_pose_features_LFTD.ipynb` – extract pose features using the LFTD method.
- `Movements_frequencyAnalysis.ipynb` – analysis of frequency spectra of movements (extracted by LFTD).

Model training, validation and testing:

- `MLP_SI_trainValTest.ipynb` – training, validation and testing of the MLP-SI model.
- `MLP_SI_dropout.ipynb` – investigation of the effect of dropout regularisation on the MLP-SI model performance (with plots and tables).
- `MLP_SI_crossval.ipynb` – 10-fold subject-independent cross-validation of the MLP-SI model.
- `MLP_SD_trainTest.ipynb` – training and testing of the MLP-SD model (uses the best architecture found for MLP-SI).
- `LSTM_architectureCapacity.ipynb` – comparison of LSTM architecture capacities.
- `LSTM_SI_trainValTest.ipynb` – training, validation and testing of the LSTM-SI model (for various dropout probabilities).
- `LSTM_SI_crossval.ipynb` – 10-fold subject-independent cross-validation of the LSTM-SI model.
- `LSTM_SD_trainTest.ipynb` – training and testing of the LSTM-SD model (uses the best architecture found for LSTM-SI).
- `SD_predict_whole.ipynb` – using SD models (MLP and LSTM), make predictions on the whole subject's video (not just the test set partition).
- `Results_validation_MLP.ipynb` – results (tables, plots) from validation of the MLP models (architecture and feature set choice).
- `Results_validation_MLP_extra.ipynb` – further results (tables, plots) for feature set choice from validation of the MLP-SI models.

- `Results_validation_LSTM.ipynb` – results (tables, plots) from validation of the LSTM-SI models (architecture choice and effect of dropout).
- `Results_testing_all.ipynb` – results (tables, plots) from testing of all four model types.

Synthesis on the robot:

- `OfflineSynthesis.ipynb` – run the system in the offline synthesis mode.
- `OnlineSynthesis.ipynb` – run the system in the online synthesis mode.
- `KFClass.py` – Kalman filter module (based on [7]) for smoothing of movements predicted during online synthesis.
- `AutomaticallyRecordSynthesis.ipynb` – automatically record synthesis on virtual robot. Requires the program *SimpleScreenRecorder* [8] and the *pyautogui* library [9].

Evaluation via web-surveys:

- `TextToSpeech_andPredict.ipynb` – generate synthetic speech from a given text using the MaryTTS text-to-speech system [10] and perform predictions using the pose regression models (MLP-SI and LSTM-SI). For synthetic speech based survey, the four texts (provided in `./Surveys/TextsForSyntheticSpeech`) from Strange Stories [11] were used.
- `maryTTS.py` – MaryTTS client module.
- `CreateVideoClips.ipynb` – automatically create side-by-side videos from the recordings of the robot and save randomisations of the videos used in the surveys (i.e. ground truths).
- `EvaluateSurvey_SyntheticSpeech.ipynb` – evaluate synthetic speech based survey responses.
- `EvaluateSurvey_NaturalSpeech.ipynb` – evaluate natural speech based survey responses and compare with synthetic ones.

Relationships between evaluation metrics and personality traits:

- `RelationshipToPersonality_quantitative.ipynb` – examine associations between several quantitative measures and 5 personality traits.
- `RelationshipToPersonality_qualitative.ipynb` – examine associations between the qualitative measure (appropriateness) and 5 personality traits.

Developed libraries (utility functions):

- `geoutils.py` – angle conversions and calculation of 11 upper-body joint angles from the extracted 3D joint positions.
- `evalutils.py` – evaluation (various metrics) and plotting of results.
- `postprocessingutils.py` – post-processing operations (smoothing, calls to evaluation methods and saving of results) after the prediction of movements.

Bash scripts:

- `extractAudio.sh` – extract audio (.wav) from videos (.mp4), downmix stereo to mono channel audio, and downsample to 16 kHz.
- `extractImgSeq.sh` – extract image sequences from videos.
- `extract2DposeOP.sh` – extract 2D joint positions from videos using OpenPose [12].
- `render2DposeOP.sh` – render 2D pose detected by OpenPose [12] into the original video.
- `extract3DposeVNect.sh` – run VNect [13] on all image sequences to estimate 3D joint positions.

3.2 Models

The folder `./Models` contains the trained neural network models MLP-SD and LSTM-SD for each subject as well as the subject-independent models MLP-SI and LSTM-SI.

3.3 Surveys

As detailed in Section 5.2 of my dissertation, the qualitative system evaluation involved two web-surveys, one based on natural speech and the other on synthetic speech.

The associated files are located in the folder `./Surveys` containing:

- Ethics Committee Approval.
- Texts used for synthetic speech generation (`./Surveys/TextsForSyntheticSpeech`). Infixes `6,7,8,9` denote the stories Banana, Picnic, Army and Glasses from Strange Stories [11] respectively.
- Videos of the robot for each survey type
  - single videos as recorded (`./Surveys/Videos/Recorded/`)
  - side-by-side (SBS) videos (`./Surveys/Videos/SideBySide/`)

  The naming conventions of the video files are as follows:

  - Infix `MLP/LSTM` denotes model type.
  - Infix `SI/SD` denotes model variant.
  - Infix `PIDXTaskY` denotes video based on natural speech of subject X performing task Y in the dataset [14].
  - Infixes `OB,SP,PR,PO` denote videos based on synthetic speech by Obadiah, Spike, Prudence and Poppy character respectively.
  - Infixes `6M,7M,8M,9M` denote videos based on synthetic speech synthesised from Banana, Picnic, Army and Glasses texts [11] respectively.

- Collected survey responses by 31/05/2018[1] are saved in (`./Surveys/Responses/`). File infix `NATURAL/SYNTHETIC` denotes the speech type and infix `SI/SD` the model variant. Rows represent individual responses and columns the survey questions. The ground truth positions of the MLP/LSTM model are saved in `.npz` files, named analogously to responses files. Each ground truth file contains a matrix where the first column identifies the SBS video and the second column has value 0 (if MLP was on the left side of the SBS video) or 1 (otherwise).

References

[1] Lucian Dinu. Icons for everything. `https://thenounproject.com/term/speak/1064313/`. [Online; accessed 25/05/2018].

[2] Softbank robotics. `https://www.ald.softbankrobotics.com/en`. [Online; accessed 27/04/2018].

[3] Jupyter notebook. `http://jupyter-notebook.readthedocs.io/en/stable/index.html`. [Online; accessed 31/05/2018].

[4] François Chollet et al. Keras. `https://github.com/keras-team/keras`, 2015.

[5] Choregraphe simulation environment. `http://doc.aldebaran.com/2-5/software/choregraphe/index.html`. [Online; accessed 27/04/2018].

[6] Hubert Pham. Pyaudio: Portaudio v19 Python bindings. `https://people.csail.mit.edu/hubert/pyaudio/`, 2006. [Online; accessed 09/05/2018].

[7] Jan Ondras, Oya Celiktutan, Evangelos Sariyanidi, and Hatice Gunes. Automatic replication of teleoperator head movements and facial expressions on a humanoid robot. In *Robot and Human Interactive Communication (RO-MAN), 2017 26th IEEE International Symposium on*, pages 745–750. IEEE, 2017.

[8] Maarten Baert. SimpleScreenRecorder: screen recorder for Linux. `https://github.com/MaartenBaert/ssr`. [Online; accessed 22/05/2018].

[9] PyAutoGUI: Python module for programmatically controlling the mouse and keyboard. `https://pyautogui.readthedocs.io/en/latest/`. [Online; accessed 22/05/2018].

[10] The MARY Text-to-Speech System. `http://mary.dfki.de/`. Version 5.2. [Online; accessed 17/05/2018].

[11] Therese Jolliffe and Simon Baron-Cohen. The strange stories test: A replication with high-functioning adults with autism or asperger syndrome. *Journal of autism and developmental disorders*, 29(5):395–406, 1999.

[12] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *CVPR*, 2017.

[13] Dushyant Mehta, Srinath Sridhar, Oleksandr Sotnychenko, Helge Rhodin, Mohammad Shafiei, Hans-Peter Seidel, Weipeng Xu, Dan Casas, and Christian Theobalt. Vnect: Real-time 3d human pose estimation with a single rgb camera. volume 36, July 2017.

[1] The dissertation is based on responses obtained by 16/05/2018.

[14] Paul Bremner, Oya Celiktutan, and Hatice Gunes. Personality perception of robot avatar tele-operators. In *Human-Robot Interaction (HRI), 2016 11th ACM/IEEE International Conference on*, pages 141–148. IEEE, 2016.