Title
Automatic tongue contour extraction in ultrasound images with convolutional neural networks

Permalink
https://escholarship.org/uc/item/0qn1d2n1

Journal
The Journal of the Acoustical Society of America, 143

Authors
Zhu, Jian
Styler, Will
Calloway, Ian C

Publication Date
2018

Peer reviewed
Objectives and Research Question

Ultrasound imaging of the tongue provides detailed articulatory data for phonetic research, but current approaches require time-consuming manual labeling of tongue contours in images. Here, we present MTracker, a method for automatic identification and extraction of precise tongue contours using a convolutional neural network (CNN) in combination with the Active Contour Algorithm.

Can a neural network automatically label tongue contours, with human-like levels of accuracy and consistency?

About the Ultrasound Data

Midsagittal ultrasound data was collected as MPEG video using a Zonare Z-One Ultrasound Unit, recording at 60fps. Human annotation used Mark Tiede’s GetContours package for MATLAB, generating 100 point splines.

About the Data:
• Training data consisted of 17,581 human-annotated frames from 11 American English speakers producing vowel and vowel-lateral syllable nuclei in CVC and CVC pairs (e.g. ‘bulk’ and ‘buck’)
• Testing data consisted of 4,360 frames from two additional American English speakers, reading ‘The North Wind and the Sun’

About the Annotation:
• Training frames: Single-annotated by a pool of annotators
• Testing frames: Annotated by three annotators (A, B and C), who were given similar training and who each had prior experience annotating ultrasound data

MTracker Neural Network Structure

We implemented the U-net architecture (Ronneberger et al. 2015) in Python 3.5, Keras, and Tensorflow, which learns from convolutional neural networks (CNN) in combination with the Active Contour Algorithm. MTracker is based entirely in open-source software, and can be downloaded and used at no cost. You’ll just...
• Install the dependencies (Keras, Tensorflow, CUDA, etc)
• Download the code, documentation, or trained model: https://github.com/lingjzhu/mtracker.github.io
• Follow the documentation on Github to run the software
• Export completed splines as X-Y series by frame for analysis

Using MTracker for your data

MTracker is based entirely in open-source software, and can be downloaded and used at no cost. You’ll just...
• Install the dependencies (Keras, Tensorflow, CUDA, etc)
• Download the code, documentation, or trained model: https://github.com/lingjzhu/mtracker.github.io
• Follow the documentation on Github to run the software
• Export completed splines as X-Y series by frame for analysis

Future Work

• Improving robustness
  – Training and testing with non-English data
  – Error detection to identify common failure modes
  – Using post-processing to mitigate/eliminate bad splines
• Supervised Use
  – Developing a workflow for second pass human verification and correction of splines in GetContours
• Automatic splining with manual correction is faster than manual splitting

Acknowledgements and References

MTrackr is based entirely in open-source software, and can be downloaded and used at no cost. You’ll just...

Training and implementing the MTracker network

Using a neural network is a two stage process. In the first stage, the model is trained with both ultrasound frames and annotator-provided tongue contour data (in a format similar to the desired output data). In the second stage, the model can automate the annotation process by offering spline predictions for novel input frames.

Data Preprocessing
• Point-based annotator splines are ‘thickened’ upwards by 10 pixels to better match the tongue signal
• The input image is cropped to remove unneeded regions, and downsampled to 64x128 pixels

Training the Neural Network
• The Dice Coefficient (a measure of pixel-wise overlap) is used as a loss function to measure model performance during training
• Training takes ~2 hours using NVIDIA Tesla K40 GPU in Michigan’s FLUX computing cluster

Data Post-Processing
• Tongue splines are extracted from output images, then processed using linear interpolation, followed by B-splines for smoothing
• The Active Contour (Snake) algorithm is then used to refine the predicted splines and improve accuracy

Output: Human Annotators vs. MTracker

We tested the correspondence between annotator and final MTracker splines on individual frames in the ‘North Wind’ test data by computing and comparing the mean sum-of-distance (MSD) from pixel to pixel.

Testing: Mean Sum of Distance

The density of pairwise errors between the three humans (red) and between MTracker and humans (blue) shows wider error distribution for MTracker, and evidence of a consistent offset.

Testing: Density of Disagreement

The density of pairwise errors between the three humans (red)
and between MTracker and humans (blue) shows wider error distribution for MTracker, and evidence of a consistent offset.

Problematic Frames and Output

Annotator Error

Difficult frames

Implausible tongues

Noise-as-signal

Supervised Use

– Developing a workflow for second pass human verification and correction of splines in GetContours
– Automatic splining with manual correction is faster than manual splining

MTracker: Strengths and Weaknesses

Overall, MTracker performs well, with approximately the same accuracy as our three trained human annotators, but there are areas which can be improved.

System Strengths
• Consistency
  – All annotations use the same criteria, even in gray areas
• Speed
  – Can annotate 2-4 frames per second on a standard laptop with an average GPU (vs. ~0.14 fps for humans)
  – MTracker annotation runs unsupervised (and 21x faster!)
• Completeness
  – Annotates all frames, allowing easy combination with acoustic forced alignment for large corpora

Ongoing Issues
• Recognizing and reliably annotating difficult frames
  – Gaps, noise, and ‘thick’ silhouettes
  – Frames with unclear or missing tongue are still annotated
• Implantable Tongue Shape Generation
  – Noise can trigger non-tongue-like shapes

System Weaknesses

• Implausible Tongue Shape Generation
• Completeness

Implementation

Training takes ~2 hours using an NVIDIA Tesla K40 GPU in Michigan’s FLUX computing cluster.

Data Post-Processing

– Can annotate 2-4 frames per second on a standard laptop
– Can download the code, documentation, or trained model: https://github.com/lingjzhu/mtracker.github.io
– Follow the documentation on Github to run the software

Future Work

• Improving robustness
• Supervised Use
• Automatic splining with manual correction is faster than manual splining

Acknowledgements and References

Work supported by NSF CoCEN 1318883 (Director: Perdicaris and Coetzee) and thanks to Mark Tiede, for participants, annotation, and often articulate ultrasound image processing and design training.

Ronneberger et al. 2015: U-Net: Conv. Networks for Biomedical Image Segmentation, DOI:10.1007/978-3-319-24574-4_28

Copyright © 2015 via CC BY-SA 3.0. If you use this in an academic setting, please cite: Ronneberger et al. 2015: U-Net: Conv. Networks for Biomedical Image Segmentation, DOI:10.1007/978-3-319-24574-4_28

Experimental and Model Development: Mark Tiede

– https://github.com/mktiede/GetContours
– https://github.com/lingjzhu/mtracker.github.io

– Follow the documentation on Github to run the software