Automatic Tongue Contour Extraction in Ultrasound Images with Convolutional Neural Networks

Jian Zhu, Will Styler, Ian Calloway

University of Michigan Department of Linguistics

lingjzhu@umich.edu - https://github.com/lingjzhu/mtracker.github.io

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:

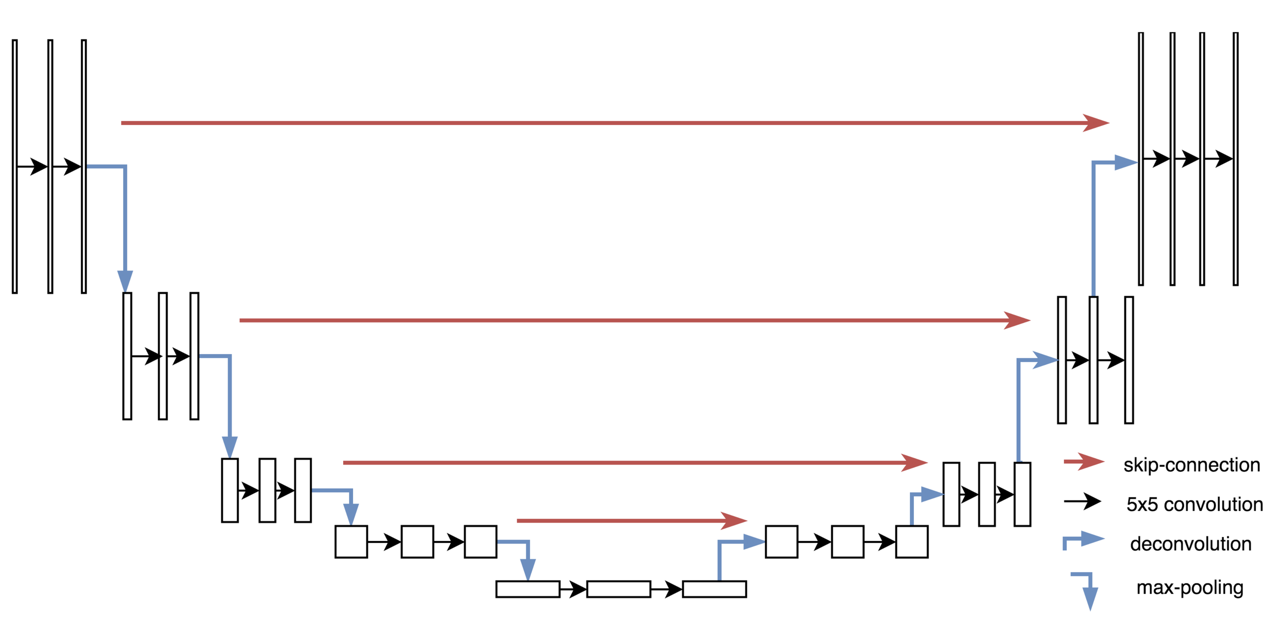
- **Training data** consisted of 17,581 human-annotated frames from 11 American English speakers producing vowel and vowel-lateral syllable nuclei in CʌlC and CʌC pairs (e.g. 'bulk' and 'buck')
- **Testing data** consisted of 4,360 frames from two additional American English speakers, reading 'The North 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 human-annotated splines using repeated convolution and max-pooling layers for feature extraction (which simplify the image in feature-identifying ways), as well as skip connections, which reuse low level features to generate more spatially precise predictions of the tongue contours.



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

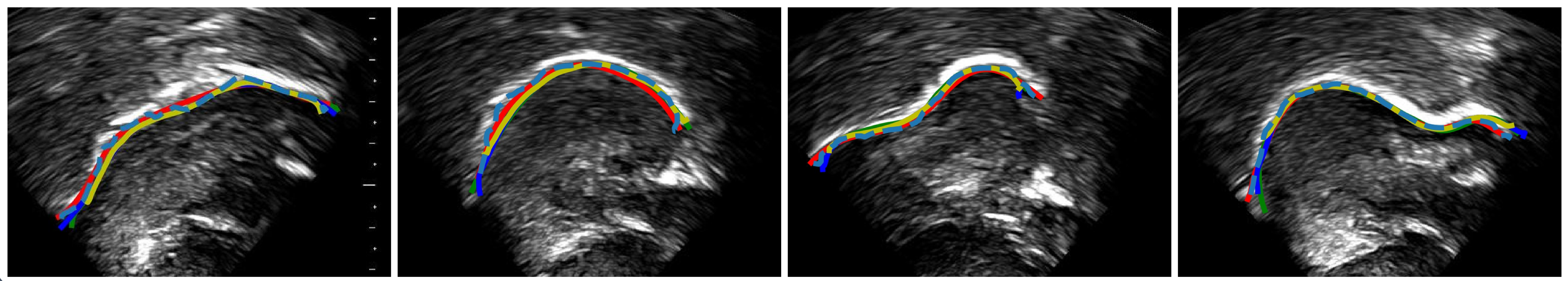
- The Dice Coefficient (a measure of pixel-wise overlap) is used as a loss function to measure model performance during training
- Training takes ~2 hours using NVIDIA Tesla K40 GPU in Michigan's FLUX computing cluster

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

Annotator A - Annotator B - Annotator C - MTracker before post-processing - MTracker's final output (dashed).



Testing: Mean Sum of Distance

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.

MSD (in Pixels, 1 pixel ≈ 0.25mm)

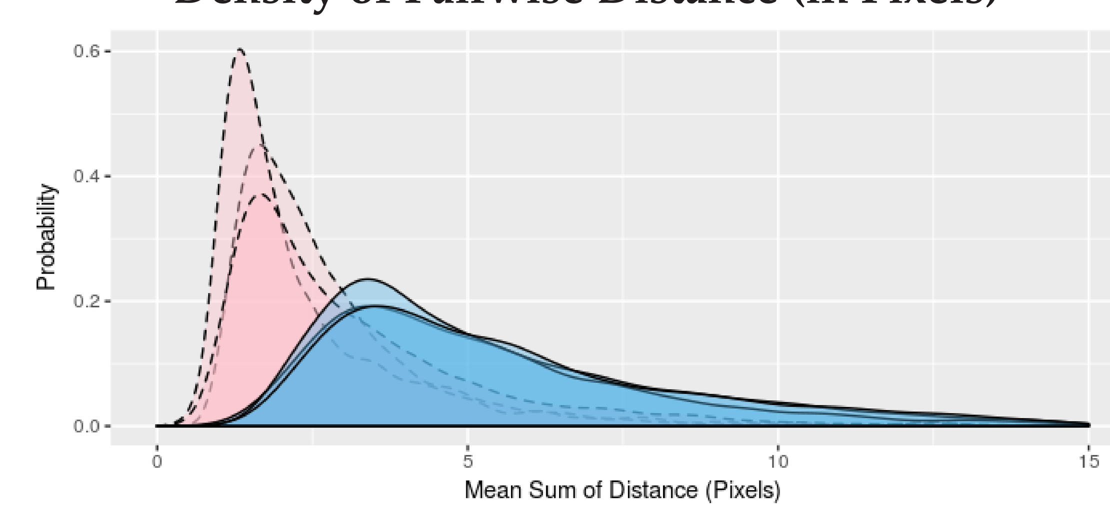
	MT	A	B	C
MT	0	5.81	5.22	5.99
A	5.81	0	2.33	2.83
B	5.22	2.33	0	3.21
C	5.99	2.83	3.21	0

Average Human-human MSD: 2.82px (0.70mm)
Average Human-MTracker MSD: 5.67px (1.41mm)

Testing: Density of Disagreement

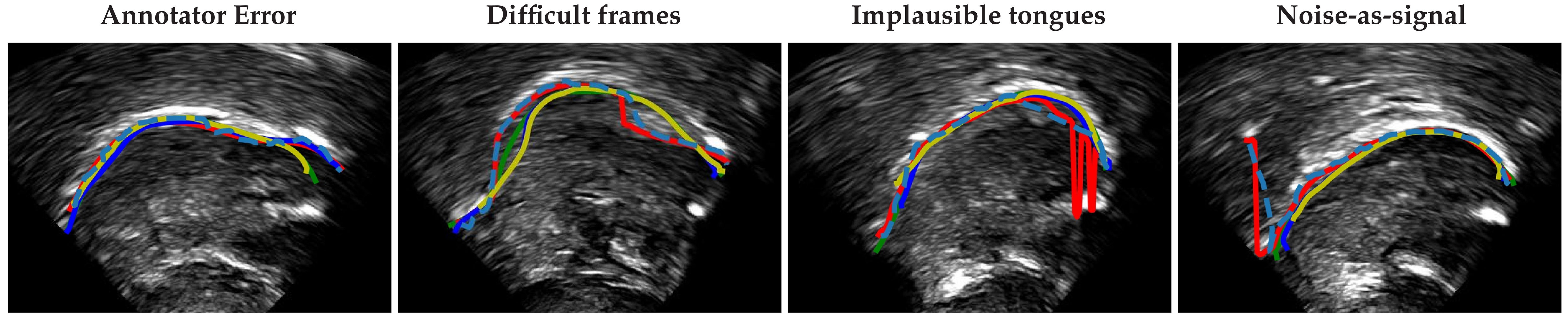
The density of pairwise errors between the three humans (red) and between MTracker and humans (blue) shows wider error distribution for MTracker, and evidence of a consistent offset.

Density of Pairwise Distance (in Pixels)



Problematic Frames and Output

Annotator Error | **Difficult frames** | **Implausible tongues** | **Noise-as-signal**



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
- Implausible Tongue Shape Generation
  - Noise can trigger non-tongue-like shapes

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 splining

Acknowledgements and References

Work supported by NSF Grant BCS-1348150 to Patrice Beddor and Andries Coetzee. Thanks to Mark Tiede, the participants, annotators, and other researchers in ultrasound image processing and deep learning.

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

GetContours by Mark Tiede: https://github.com/mktiede/GetContours