6.189 IAP 2007

Student Project Presentation

Speech Synthesis

Speech Synthesis

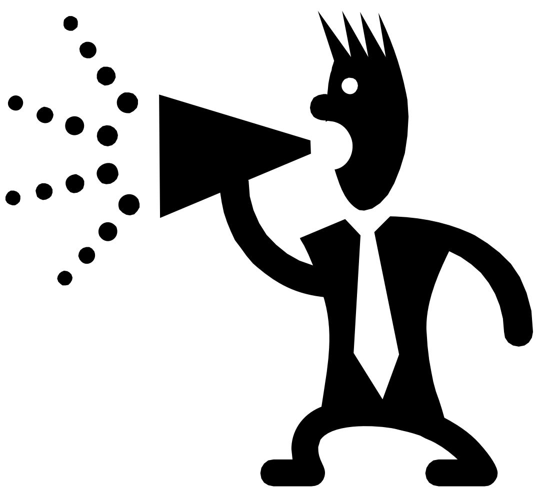
Omari Stephens

Joyce Chen

Eric Eisner

Drew Altschul

Brown Westrick



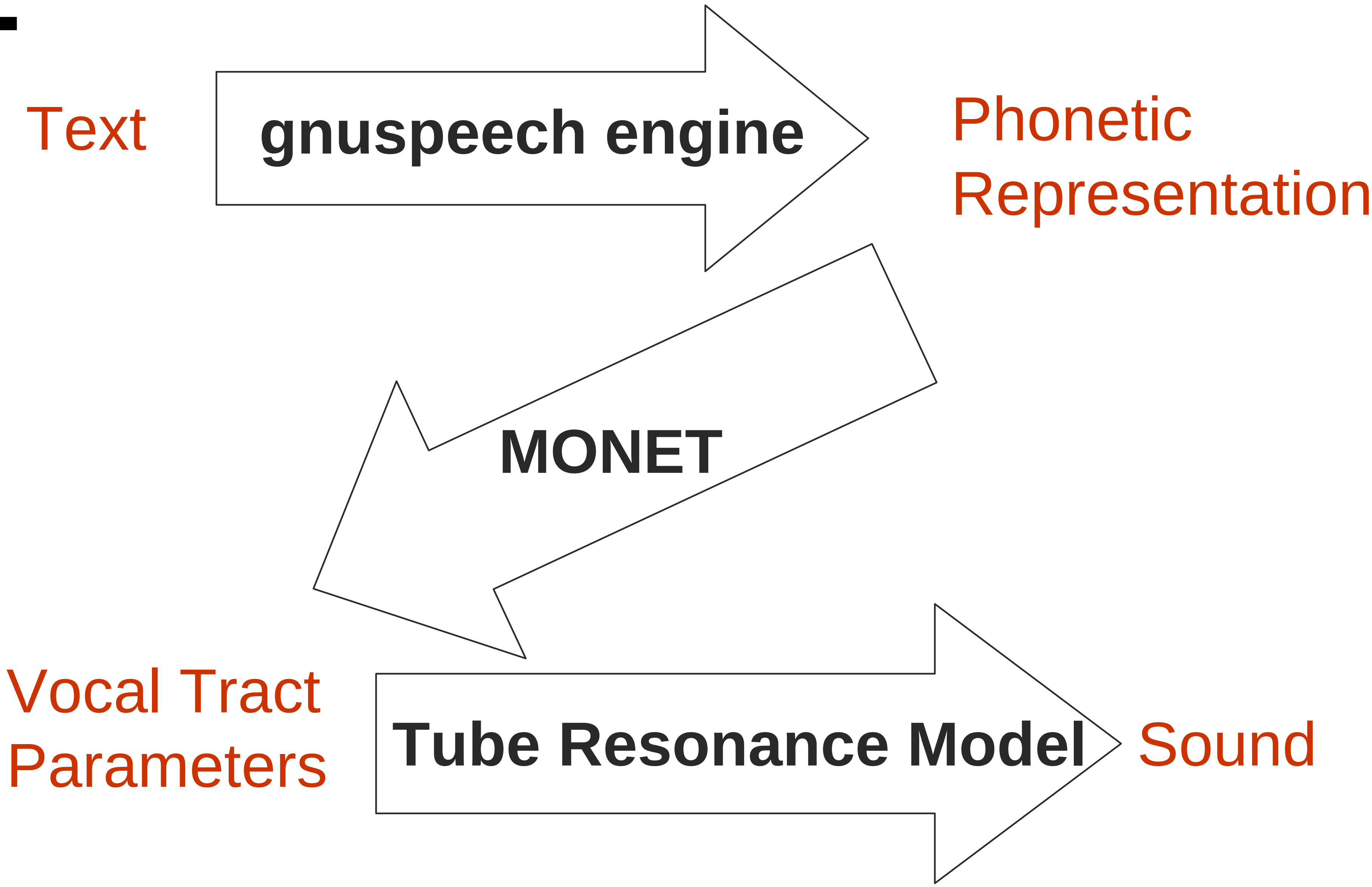
Speech Synthesis Goals

- Produce speech in real time by modeling airflow in the vocal tract
- Modify existing **gnuspeech** software to run on Cell
- Improve speech quality by using additional computational cycles

Why Gnuspeech?

- Gnuspeech available free under the GNU public license
- Models airflow in the vocal tract in real time
  - no prerecorded sounds
- Designed for linguistics research
- Potential to increase in quality with model complexity and computational power

Algorithm Components



Part 1: Gnuspeech Engine

- transform text into purely phonetic form

"all your base are belong to us"
/c // /0 # /w /**_aw_l** /w /**_y_aw_r** /w /**_b_e_i_s** /w **ar_r** /w **b_i./_l_o_ng** /w **t_uu** /w /l /***a_s** # // /c

- dictionary lookup for pronunciation
- ambiguous cases determined by simple linguistic model
- markers for punctuation information
  - word and phrase boundaries
  - basic intonation

Part 2: MONET

- Transform standard phonetic form into vocal tract simulation parameters
- Determine appropriate rhythm and intonation for the given phrase
- Calculate effects neighboring sounds have on each other
- Output seqence of *postures* – snapshots of the shapes the vocal tract takes over time

Part 3:Tube Resonance Model

- Vocal tract divided into 8 main regions, plus nose
  - modeled as coaxial cylinders with variable radius
  - noise source at one end
- Shape of the vocal tract changes over the course of an utterance
- Models propagation of pressure wave
  - constantly changing vocal tract shape
  - physics
- Pressure wave exiting the mouth = speech

Allocating Resources

- **Gnuspeech, MONET**
- Little computation
- Extensive dictionary lookups
- No improvements in quality feasible
- Run on PPE

- **Tube Resonance Model**
- More computation
- Small (constantly updated) data set
- Step size decrease may improve output
- Run on SPEs

TRM Algorithm

- Input: sequence of postures
- Main loop:
  - Update the noise generator (“vocal folds”)
  - Move the shape of the vocal tract one step towards the next posture
  - Update the pressure wave by one timestep inside the new vocal tract shape
  - Record the state of the wave at the mouth aperture

TRM Profile

- Where is the time spent in TRM?
- Task: Percent of Total Time
  - Updating the noise generator: 52%
  - Main loop (except noise gen.): 25%
  - Post-processing sound data: 22%
- Time per main loop: ~15µs
- Decreasing step size won't affect above balance of computation in main loop

Parallelism in the Algorithm

- Very scarce
- Each main loop iteration has true dependences on the previous one
  - state of air flow in vocal tract
  - state of noise generator wave
- Default main loop frequency: 70kHz
- Pipelining possible for post-processing

Challenges

- Objective C and GNUStep
  - difficult to read
  - even harder to debug
  - cannot be compiled for SPE
- Time-consuming conversion attempts
- Dynamic pointer alignment

What **is** working now

- Line-buffered text *to* utterances *to* execution of the TRM
- Monet replacement works minimally
- Tube runs on PPE
- Tube partially runs on SPE

What is **not** working yet

- Obscure GNUStep/Monet dictionary bug
- Monet does not properly execute the tube
- The tube does not successfully receive data
- The driver does not receive data from the post-processor

Conclusions and Future Work

- Extremely difficult to parallelize

- Parallelization can help vocalization quality
  - naturalness
  - speaker identification
  - vowel identification
- Worth the time to rewrite from scratch
  - C and/or C++
  - without the GUI