Visualization tools & demos
and the ICSI Realization group

Dan Ellis
International Computer Science Institute, Berkeley CA
<dpwe@icsi.berkeley.edu>

Outline

1 Visualization: Goals and philosophy
2 Tour of existing tools
3 Using recogviz
4 Future developments & demos
Visualization: Goals and philosophy

Why work on visualization tools?

• Visual representations give insights
  - able to scan large amounts of data
  - a perspective different from other analyses

• Interactive exploration gives insights
  - ‘chasing down’ anomalous phenomena

• Graphical, interactive demonstrations are accessible & appealing
  - for giving demos to visitors
  - for communicating to colleagues

BUT

• Visualization is often non-critical
  → tools must be prefabricated, easy to use
Goals of visualization tool development

- “A demo on every desk”
  - facilitate understanding/insight
  - & diagnose problems/anomalies
  - support presentations & provide figures
  - promote ‘demo culture’
  →minimize barrier?

- Tomorrows demos...
  ...can’t be predicted, so
  - establish framework
  - provide modules

- UI coding is tedious & short-lived
  - prefer high-level, portable solution

- Flexibility to twist to new ends?
# Choosing an infrastructure

- **Special purpose vs. scripting language**
  - need expandability
  - want to be able to modify

- **Choice of scripting language:**

<table>
<thead>
<tr>
<th></th>
<th>sh</th>
<th>perl</th>
<th>Tcl/Tk</th>
<th>java</th>
<th>matlab</th>
</tr>
</thead>
<tbody>
<tr>
<td>graphics</td>
<td></td>
<td>?</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>command shell</td>
<td>√</td>
<td></td>
<td>√</td>
<td></td>
<td>√</td>
</tr>
<tr>
<td>run programs</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sig.proc. support</td>
<td></td>
<td></td>
<td></td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>user extensions</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>‘nice’ syntax</td>
<td>√</td>
<td></td>
<td>√?</td>
<td>√?</td>
<td></td>
</tr>
<tr>
<td>local expertise</td>
<td>√</td>
<td>√</td>
<td>?</td>
<td>?</td>
<td>√</td>
</tr>
</tbody>
</table>

→ **Tcl/Tk + extensions framework**

*xwaves, lyre*
Tour of existing tools

• Early experiments
  - netviz - neural-net weights display
  - hmmviz - HMM model display

• Old but useful
  - sgramImg - on-demand web spectrograms
  - pfview - pfile inspection
  - simpleui - basic interactive recognition

• The current generation
  - berpdemo98 - interactive speech application
  - recogviz - recognizer display & comparison
netviz & hmmviz

- Early experiments

- Advantages of Tcl
  - netviz is 300 lines (+ floatArray extension)
  - clean postscript generated by hmmviz
sgramImg.cgi

- CGI script: spectrogram GIF on-demand

```html
<img src="http://montoya.icsi.berkeley.edu:8080/~dpwe/cgi-bin/sgramImg.cgi?filename=/u/stp/data/ws97hr1/wav/2085-B-0128.wav.gz">
```

- Automatically includes xlabel annotation
- Try it:
  [http://montoya.icsi.berkeley.edu:8080/~dpwe/cgi-bin/soundview.cgi](http://montoya.icsi.berkeley.edu:8080/~dpwe/cgi-bin/soundview.cgi)
pfview

- Display contents of pfiles as images

- Provision for multiple panels, labels, PS output

- Try it:
  > ~dpwe/projects/pfview/pfview.tcl pfile=<pfile>
     [idlist=<utid file>]
simpleui

- Minimal interactive demo of recognizer
- Just connects audio_frontend to a recognizer
  - easy to modify for other grammars, features

Simple structure:

Try it:

> /u/drspeech/share/bin/simpleui
berpdemo98

- Our ‘old’ application plus recogviz modules

Try it:

> /u/drspeech/share/bin/berpdemo98
recogviz

- Motivation: compare recognition techniques
  - at each stage in process (signal, features, probs)

- Try it:
  > /u/dpwe/projects/recogviz/recogviz
Using recogviz

- System structure
Using recogviz (cont.)

- Configuration files: (Tcl syntax)
  - defaults=./defaults.def - common definitions

```tcl
set NUMBERSDIR   "/u/drspeech/data/NUMBERS95"
set listfile     "$/NUMBERSDIR/list/numbers95-cs-dev.utids"
set wavfilecmd   "numbers95_findfile prefix=$NUMBERSDIR/ utid=%u"
set samplerate   8000
set frametime    0.010
# Standard network geometry/type
set mlp3_hidden_size 500
set mlp3_output_size 56
# Other common values
set ftr_start     0
...

- recog1=./recog1.def

set title        "Rasta-plp cepstra"
set DPWE          "/u/dpwe/projects/NUMBERS95"
set ftr_file      "$DPWE/pfiles/n95dev-lras-plp-cep.pf"
set ftr_width     27
set norm_file     "$DPWE/pfiles/n95tr-lras-plp-cep.norm"
set weights       "$DPWE/results/1997sep17/n95tr-embed+iter2.weights"
...

+ recog2=./recog2.def
```

- Use dr_recog params files? (y0/noway)
Future developments & demos

- “A demo on every desk”
  - but demos are more than graphics: message

- Illustrations for research advances
  - feature domains
  - recognizer combinations

- Application-scale demos
  - grandson of BeRP
  - ?
Creating new demos

- **Available pieces:**
  
  - **Hackable top-level scripts**
    - simpleui
    - recogviz
    - berpdemo
  
  - **Embeddable Tcl programs**
    - audio_frontend.tcl
    - probs.tcl
  
  - **[incr Tcl] classes**
    - CanvFA.itcl
    - Recog.itcl
  
  - **Tcl source files**
    - CanvLabels.tcl
    - cgiutils.tcl
  
  - **Tcl/Tk loadable extensions**
    - libaprl
    - libarray_otcl
    - libsound_otcl
    - libpfif_otcl
  
  - **Binary programs**
    - itkwish
    - rasta
    - qnsfwd
    - noway

- **Get bits via Tcl package system**
  
  see also: http://www.tcltk.com, ...
Future work

- **Improving the existing bits**
  - cleaning up modules
  - better example shells
  - documentation

- **Generating new demos**
  - .. within existing tools (recogviz)
  - .. employing existing pieces

- **Collaborative applications**
  - ThisL
  - Daedelus

- **Wider questions:**
  - distributable demos?
  - cross-platform demos?
  - applications vs. basic research?