Testbed for On-Board Signal Processing
(one year with signal processing folks)

Mathieu Kemp - Bluefin Robotics
Li Li, Jonathan Odom, Chris Potts, Jeff Krolik - Duke University
in conclusion ...

After a 12-month study of Signal Processing Folks, I reached the conclusion that they are comfortable with:

\[
P(\alpha, \beta, \theta) = \vec{w}'(\alpha, \beta, \theta)S(\omega)\vec{w}(\alpha, \beta, \theta)
\]

\[
\vec{w}(\alpha, \beta, \theta) = \frac{1}{M} \begin{bmatrix}
\exp[jk_0 \sin(\alpha)x_1(\theta) + j2\pi k_0 \sin(\beta)y_1(\theta)] \\
\vdots \\
\exp[jk_0 \sin(\alpha)x_M(\theta) + j2\pi k_0 \sin(\beta)y_M(\theta)]
\end{bmatrix}
\]

\[
S(\omega) = \text{Cross Spectral Density Matrix at 7 kHz}
\]

Matlab

```matlab
% fourier vector
K = exp(i*bf.OmegaTau*[1:bf.SamplesPerSnapshot]')/(bf.SamplesPerSnapshot);

% csd computation
CSD = zeros(bf.NbElements,bf.NbElements);
for s=1:bf.SnapshotsPerFile
    X = x(:,(s-1)*bf.SamplesPerSnapshot+[1:bf.SamplesPerSnapshot]) * K;
    CSD = CSD + X'*X;
end
CSD = CSD /bf.SnapshotsPerFile;

% beamforming
BFO = zeros(bf.NbAzimuths,bf.NbElevations);
for az = 1:bf.NbAzimuths
    for el = 1:bf.NbElevations
        V = bf.SteeringVector{az,el};
        BFO(az,el) = real(V'*CSD*V);
    end
end
```
but they seem to have a harder time with ...
Krolik's SAM lab

- Jeff Krolik's Sensor Array and Multipath Signal Processing Laboratory (Duke University):
  - sonar/radar array processing
  - multipath
  - tracking in clutter
  - left-right disambiguation, etc

- Traditional approach
  - get a data set from a cruise
  - process in Matlab

- Got tougher lately
  - proprietary & classified data

- New paradigm
  - buy hardware to generate own data
  - ex: "floor" robot + 64-element microphone array.
• **Videre robot**
  – independent speed/turn rate control
  – pan-tilt actuators
  – 8 sonar ranges
  – odometry
  – lidar (Hokuyo URG, 5m range)
  – Linux PC
  – sound card
  – WiFi

• **8 x 8 microphone array**
  NIST,
  re-configurable: forward facing, towed, 2D, variable geometry, etc.

• **command&control**
  Matlab-based, joystick
No software framework?

Working theory 1: provide interfaces to the hardware + a software framework
### CRobot()
- **Videre ERA**
- bidirectional
- thread reads data stream from vehicle and sends external commands to it
- **Start()**: start/stop the thread (thread trampoline to get around `pthread_create()`'s C binding)
- **Get()**: nav, sonar.
- **Set()**: actuators.
- **Send()**: commands
- **SetDeadZone()**: zero axes if too small

### iVehicle.cpp
- **OnStartUp();**
  - instantiate CMoosVehicle
    - `cRobot* m_Vehicle`
  - **Start()**
- **OnConnectToServer();**
  - register to QUIT msg
  - register to COMMAND msg
  - **SetQuitOnFailedIterate(true);**
- **OnNewMail();**
  - on QUIT, set global flag to false
  - else store COMMAND parameters
- **Iterate();**
  - if global flag is set, stop/delete Robots and return false
  - else set and send variables for thread
## JOYSTICK

### CJoystick()
- **Logitech Extreme3D**
- data out only
- `<linux/joystick.h>`
- thread reads data stream from joystick
- `Start/Shutdown()`: start/stop the thread (trampoline)
- `GetAxes()`: return the most recent axes.
- `GetButtons`: return the most recent buttons.
- `SetDeadZone()`: zero axes if too small

### iJoystick.cpp
- **OnStartUp();**
  - instantiate `CMoosJoystick`
    - `cJoystick`* `m_Joystick`
  - `SetDeadZone()`
  - `Start()`
- **OnConnectToServer();**
  - register to QUIT msg
  - `SetQuitOnFailedIterate(true);`
- **OnNewMail();**
  - on QUIT, set global flag to false
- **Iterate();**
  - if global flag is set, stop/delete joystick and return false
  - else read, process, post `SEVO_COMMAND` or `SPEED_COMMAND` or QUIT
CNistArray()

- **NIST mk3 array**
- 64 channels, 24 bits, UDP socket, 22050 hz
- bidirectional
- unit of data = file
- thread reads data stream from array and sends external commands to it
- Start/Stop(): start/stop the polling thread.
- WriteDataToFile(Filename,NbSamples): write most recent samples to a file

iArray.cpp

- **OnStartUp();**
  - instantiate CMoosArray
  - CNistArray* m_Array;
  - Start()
- **OnConnectToServer();**
  - register for QUIT msg
  - register for SEND_ARRAY_DATA
  - SetQuitOnFailedIterate(true);
- **OnNewMail();**
  - on QUIT, set global flag to false
  - on SEND_ARRAY_DATA, set flag
- **Iterate();**
  - if QUIT flag is set, stop/delete array and return false
  - if DATA flag is set, write data to file.
No go: students loved RC-ing the vehicle (so did the ONR sponsor) + the "live" data aspect, but didn't write new code

???

Working theory 2: that was too much hardware to handle at once.
Virtual microphone (J. Odom)

Distinguishing features:
- higher amplification than a conventional beamformer
- real-time adaptation (MVDR)

\[ y(t_n) = \sum_{\text{microphone } m} w_m \ast x_m(t_n) \]

\[ w_m(t_n) = \sum_{\text{frequency } f} \exp(i2\pi ft_n) \frac{(R^{-1}v)^m}{v^+ R^{-1}v}(f) \]

\[ R_{mm'}(f) = H_m(f) H_{m'}^*(f) \]

\[ v_m(f) = \left\{ \exp(-i(2\pi fd / c) \sin(\theta_{\text{target}})m) \right\} \]
CommandAndControl (matlab)

MOOSDB

iArray (acoustic array)

data file

array h/w

iSpeaker (projector)

sound card / ALSA

Software Version 2
Outcome

• Worked well - 15dB rejection
• Interesting problem:
  – mismatch between array clock and sound card clock changed tone frequency
  – location of beamformer peak moved a few degrees
  – array gain decreased
• Q: why did you use it?
• A: because you put Matlab in there
• in other words, it was not the s/w architecture, it was not too much h/w, it was the presence of Matlab that made it useful!
• Test this: multiple projects by separate students with the full hardware suite.
Was Matlab the enabler?

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.** Matlab/Java is what they teach nowadays.** Not C. Not C++.</td>
<td></td>
</tr>
<tr>
<td>2. <strong>In comparison to Matlab, C++ (gsl, blas) is a nightmare to use:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>– each matrix add/multiply is a function call</td>
</tr>
<tr>
<td></td>
<td>– real/complex support is not always present</td>
</tr>
<tr>
<td></td>
<td>– function names are a mile long</td>
</tr>
<tr>
<td></td>
<td>– each time you change something you need to recompile and redeploy</td>
</tr>
<tr>
<td>3. <strong>What if we ran the on-board algorithms directly in Matlab?</strong></td>
<td></td>
</tr>
</tbody>
</table>
Software Version N

- Topside
  - mCommandAndControl (matlab)

- Vehicle
  - iArray (acoustic array)
  - iVehicle (motion/position/servo)
  - iJoystick (remote)
  - iLidar (lidar)
  - videre h/w

- MOOSDB
  - data file
  - array h/w
  - data file
CLidar()

- Hokuyo URG
- serial, 5m, 10hz, 240° FOV, 0.4 ° res
- bidirectional
- unit of data = file
- thread reads data stream from lidar and sends external commands to it
- Start/Stop(): start/stop the polling thread.
- WriteDataToFile(Filename, NbSamples): write most recent samples to a file

iLidar.cpp

- OnStartUp();
  - instantiate CMoosLidar
    - CNLidar* m_Lidar;
  - Start()
- OnConnectToServer();
  - register for QUIT msg
  - register for SEND_LIDAR_DATA
  - SetQuitOnFailedIterate(true);
- OnNewMail();
  - on QUIT, set global flag to false
  - on SEND_LIDAR_DATA, set flag
- Iterate();
  - if QUIT flag is set, stop/delete lidar and return false
  - if DATA flag is set, write data to file.
2D beamformer (J. Odom)
Deformable array (L. Li)
DP-SLAM and Beamforming (C. Potts)
• I'm not advocating running Matlab on-board a UUV but it sure streamlines the integration/testing of signal processing algorithms.
• Would be interesting to look at alternatives that could streamline things - for example Vermeij's pOctaver, or Python's SciPy library with the Python Real-Time Engine.