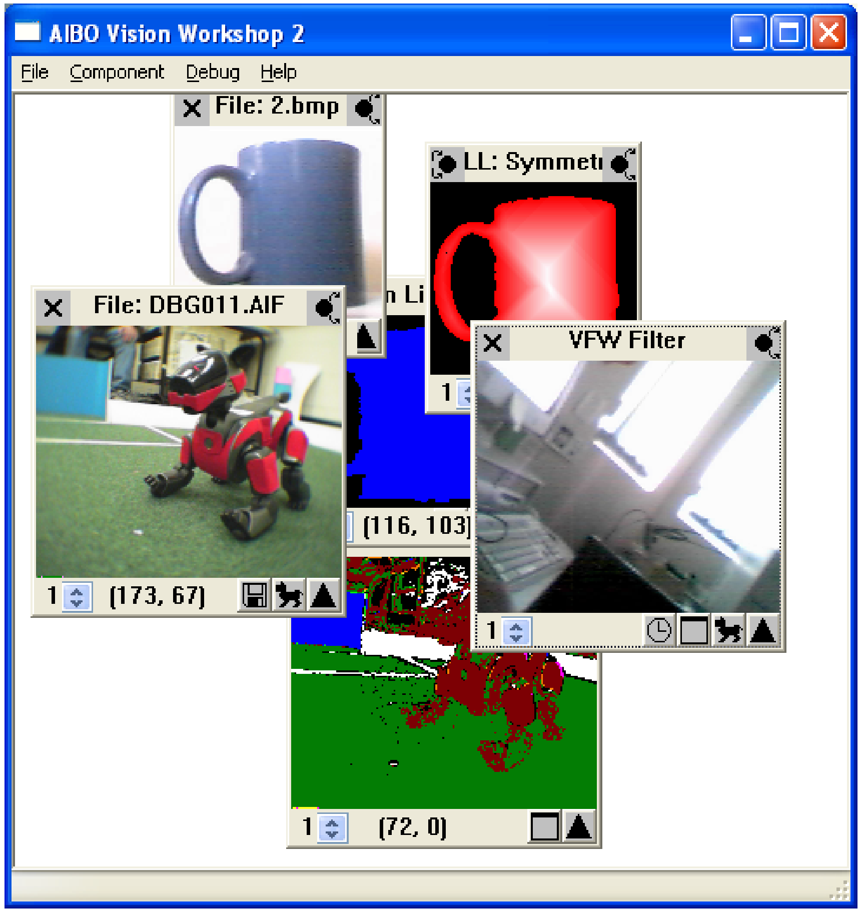
Real-Time Embedded Vision System Development using AIBO Vision Workshop 2

Nathan Lovell
Griffith University, Australia

AIBO Vision Workshop 2 (AVW2)

- Designed to assist development of vision systems
  - Divides the process of vision processing into a number of stages
  - Each processing stage is encapsulated by a user-created “filter” object

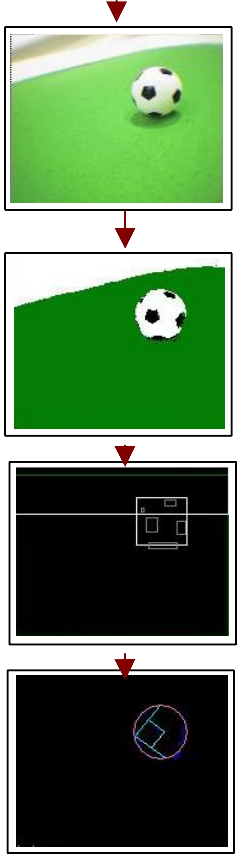


The result of each filter can be viewed and assessed in relation to all the other filters in the processing chain

AVW2 Conceptually

Image Data Source (Video Camera, Robot, Image Repository)

AVW2



1. Data is sourced from an external device

2. Each filter performs an individual processing step using the cumulated data available to each previous filter

3. The last filter in the pipeline produces the desired output

AVW2 Input Filters

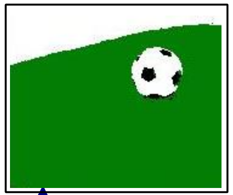
- There are several input filters that act as data sources for the vision pipeline
- AVW2 supports input:
  - From files (BMP, JPeg, Giff, and it's own native format AI2)
  - Over network connections (we use this to receive images from AIBOs over wireless network)
  - From any standard windows video source (supporting Microsoft's Video For Windows (VFW) and DirectShow standards)

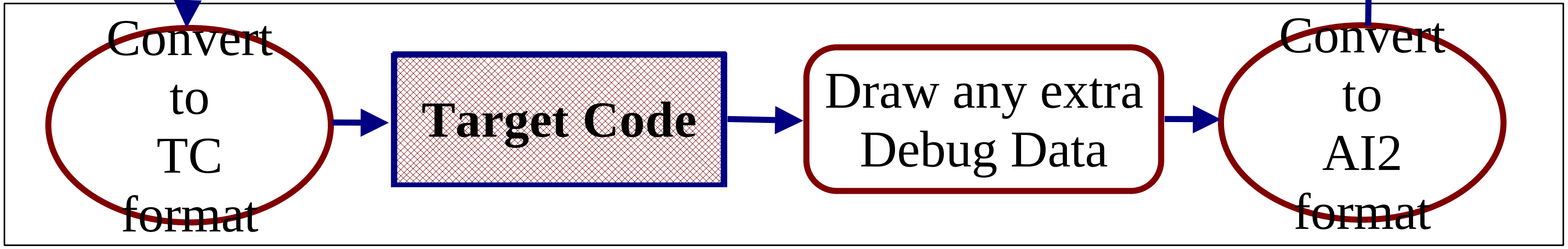
AVW2 Processing Filters

- We want to run code that is under development in AVW2
  - Not on target device
  - We will see why later
- AVW2 allows the user to run unchanged, target platform code as either:
  - A pre-compiled library (DLL)
  - A scripted component interpreted at runtime
  - A scripted component compiled at runtime

Filter Mechanics

- The user creates a filter object which wraps the target code
  - The filter must convert incoming data to the desired format and re-convert outgoing data back to AVW2's format
  - The functionality for displaying images, interacting with AVW2 and the user (and many other features discussed later) is inherited from the base class making the filter very “light” and easy to implement

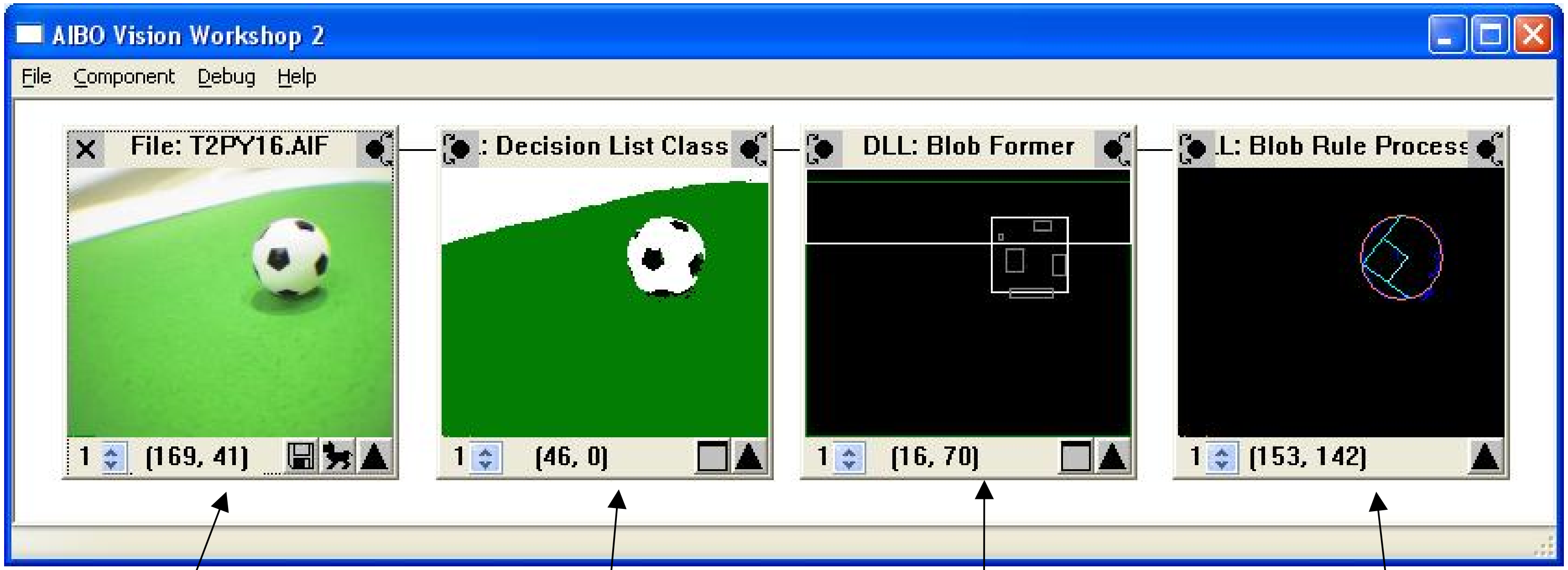




Processing Chains

- Filters are inserted, deleted and connected into processing chains at runtime
- The user creates a chain that emulates the order of processing on the target device enabling the entire image processing
- Data is passed down the chain starting from the input filters
- AVW2 manages dependencies between filters in the chain
  - Even when it knows nothing of the data types of the filter

A Processing Chain Example



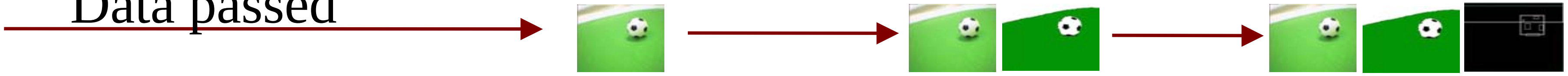
Input from a file

Colour segmentation

Connected regions of colour

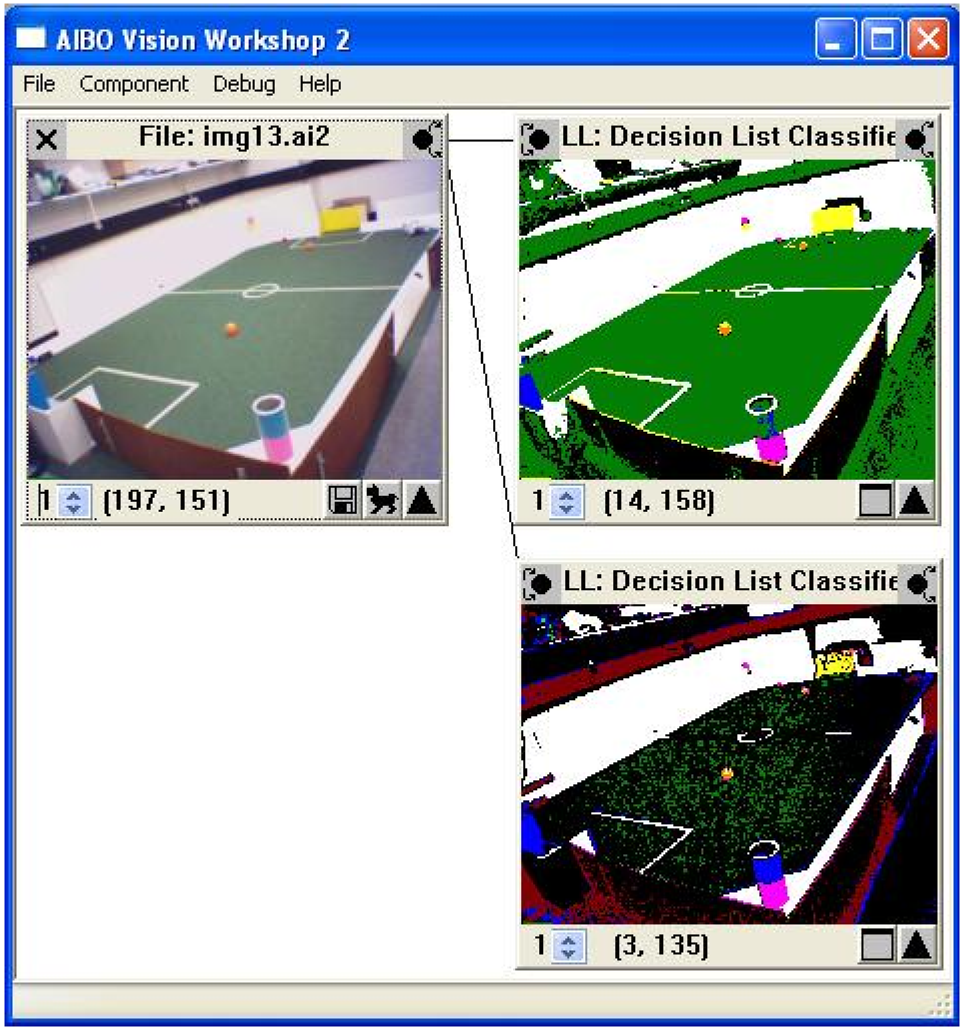
Ball recognition

Data passed



Why Chains, not Pipelines?

- We can use the branching structure of a chain to compare the output of two different filters that perform the same task


AIBO Vision Workshop 2
File Component Debug Help
File: img13.ai2
LL: Decision List Classifie
LL: Decision List Classifie
1 (197, 151)
1 (14, 158)
1 (3, 135)

Why is AVW2 Necessary?

- Development of vision systems is very difficult:
  - Complex, often slow algorithms
  - Large amounts of source data
  - Difficult to debug code on an incoming vision stream
- If you are developing for an embedded device (such as a mobile robot) then the difficulties compound:
  - Often only text I/O debugging facilities provided
  - Lack of profiling and runtime analysis support
  - Lack of processing power
  - Hardware dependencies

How Does AVW2 Help?

- Can be used as a development platform
- Can be used as a debug/test environment
- Can be used as a code profiler and performance evaluation tool
- Immerses the developer into the vision processing pipeline where each step can be evaluated in the context of the whole pipeline and in a visual manner
- No longer just a support tool for AIBO development!

AVW2 as a Development Platform

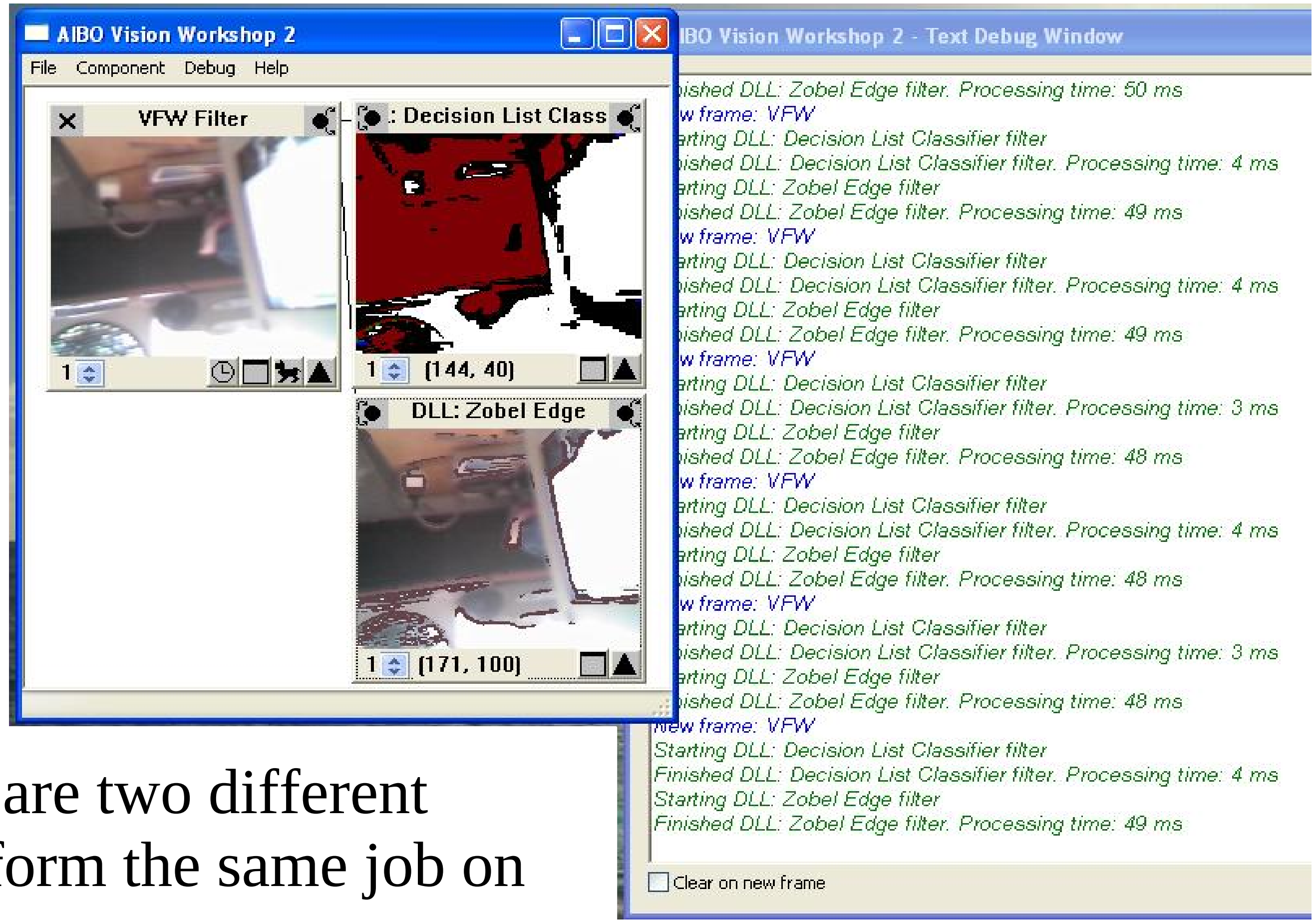
- AVW2 runs the CINT C/C++ interpreter core
  - http://root.cern.ch/root/Cint.html
- Code intended for the target device can be run and evaluated, without compilation, in a filter that reads C++ scripts
- It's easy to later compile it into a DLL to run without the overhead of the scripting engine – simply add “#pragma compile” at the start of the script!

AVW2 as a Debug/Test Environment

- It is possible to set up a real-time stream of images from the target device to AVW2 and to evaluate the output of each filter on a per-image basis
- You can even save the stream for later image-by-image analysis
- AVW2 supports code breakpoints, variable inspection and alteration, step-by-step debugging and most other modern debugging facilities
  - From both scripts and compiled filters

AVW2 as a Code Profiler and Performance Evaluation Tool

- Each filter is automatically profiled every time it is executed
- Filters execute strictly linearly so it is possible to compare profile data between two filters
  - Thus you can compare two different algorithms that perform the same job on the same data



```
ished DLL: Zobel Edge filter. Processing time: 50 ms
w frame: VFW
arting DLL: Decision List Classifier filter
ished DLL: Decision List Classifier filter. Processing time: 4 ms
arting DLL: Zobel Edge filter
ished DLL: Zobel Edge filter. Processing time: 49 ms
w frame: VFW
arting DLL: Decision List Classifier filter
ished DLL: Decision List Classifier filter. Processing time: 4 ms
arting DLL: Zobel Edge filter
ished DLL: Zobel Edge filter. Processing time: 49 ms
w frame: VFW
arting DLL: Decision List Classifier filter
ished DLL: Decision List Classifier filter. Processing time: 3 ms
arting DLL: Zobel Edge filter
ished DLL: Zobel Edge filter. Processing time: 48 ms
w frame: VFW
arting DLL: Decision List Classifier filter
ished DLL: Decision List Classifier filter. Processing time: 4 ms
arting DLL: Zobel Edge filter
ished DLL: Zobel Edge filter. Processing time: 48 ms
w frame: VFW
arting DLL: Decision List Classifier filter
ished DLL: Decision List Classifier filter. Processing time: 3 ms
arting DLL: Zobel Edge filter
ished DLL: Zobel Edge filter. Processing time: 48 ms
New frame: VFW
Starting DLL: Decision List Classifier filter
Finished DLL: Decision List Classifier filter. Processing time: 4 ms
Starting DLL: Zobel Edge filter
Finished DLL: Zobel Edge filter. Processing time: 49 ms
```

Demo

Questions? Comments?