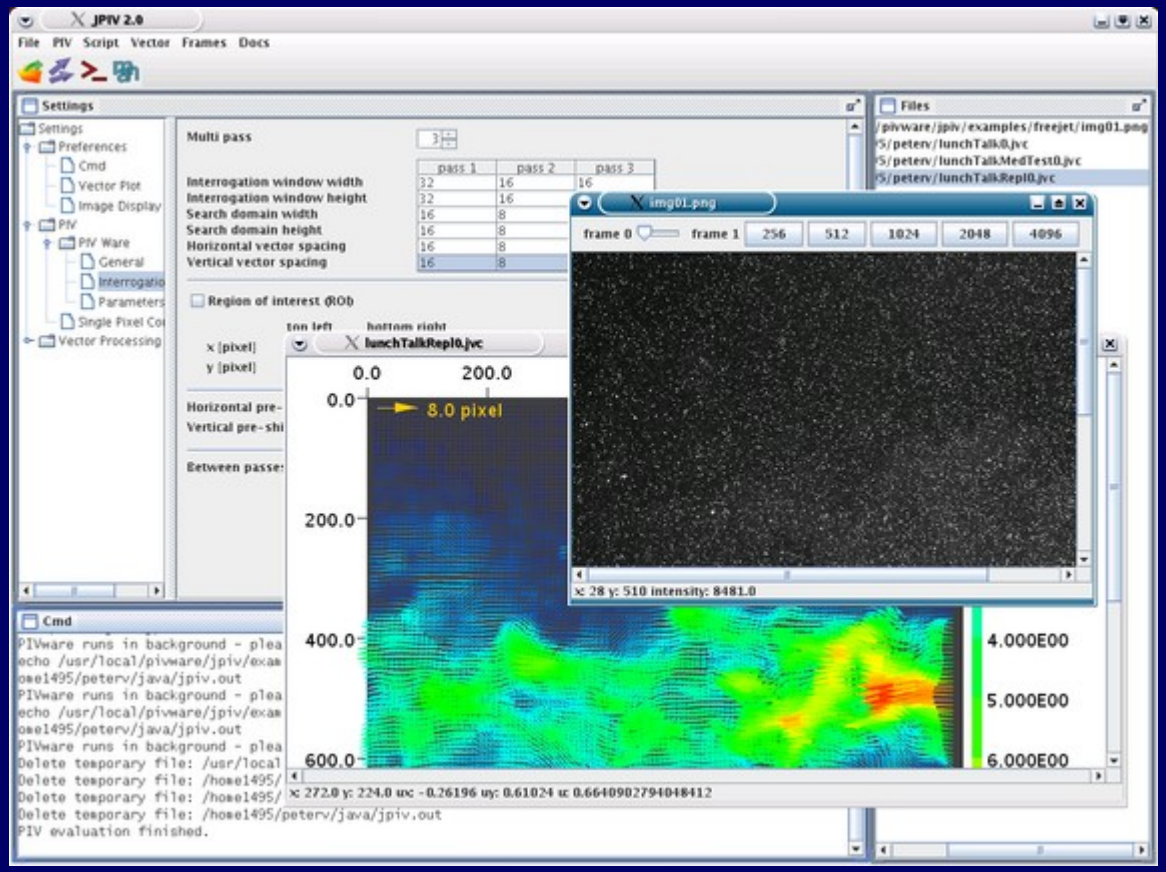
Introduction to Jpiv

PIV evaluation and post-processing



Lunch Talk

Peter Vennemann

April 19, 2006

Laboratory for Aero- and Hydrodynamics

TU Delft

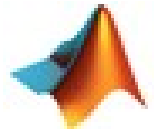
Delft University of Technology

Outline

- Purpose of Jpiv
  - Problem
  - Ambition
- Why Java?
- Jpiv graphical user interface
- General procedure of data processing
- Implementing own software components
  - Executing external software within jpiv
  - Configuring graphical user dialogs
- Documentation
- Questions

Purpose of Jpiv - Problem

- Many lossy pieces of PIV evaluation- and post-processing software are scattered at the laboratory

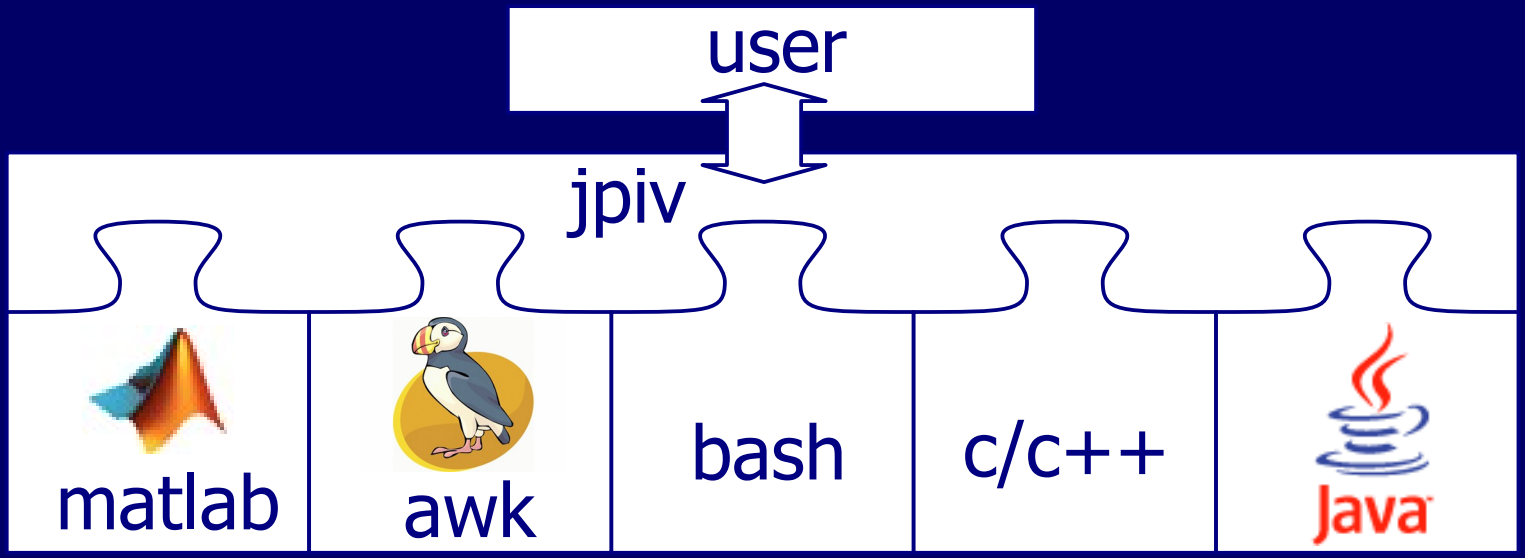
 matlab	 awk	bash	c/c++	 Java

➔ Programming is often done twice or more often

➔ Data processing potential is not used because of unawareness

Purpose of Jpiv - Ambition

- A single, self-explaining, graphical surface for existing software components
- A standardized interface that enables everyone to integrate his own code



TU Delft

Why Java?

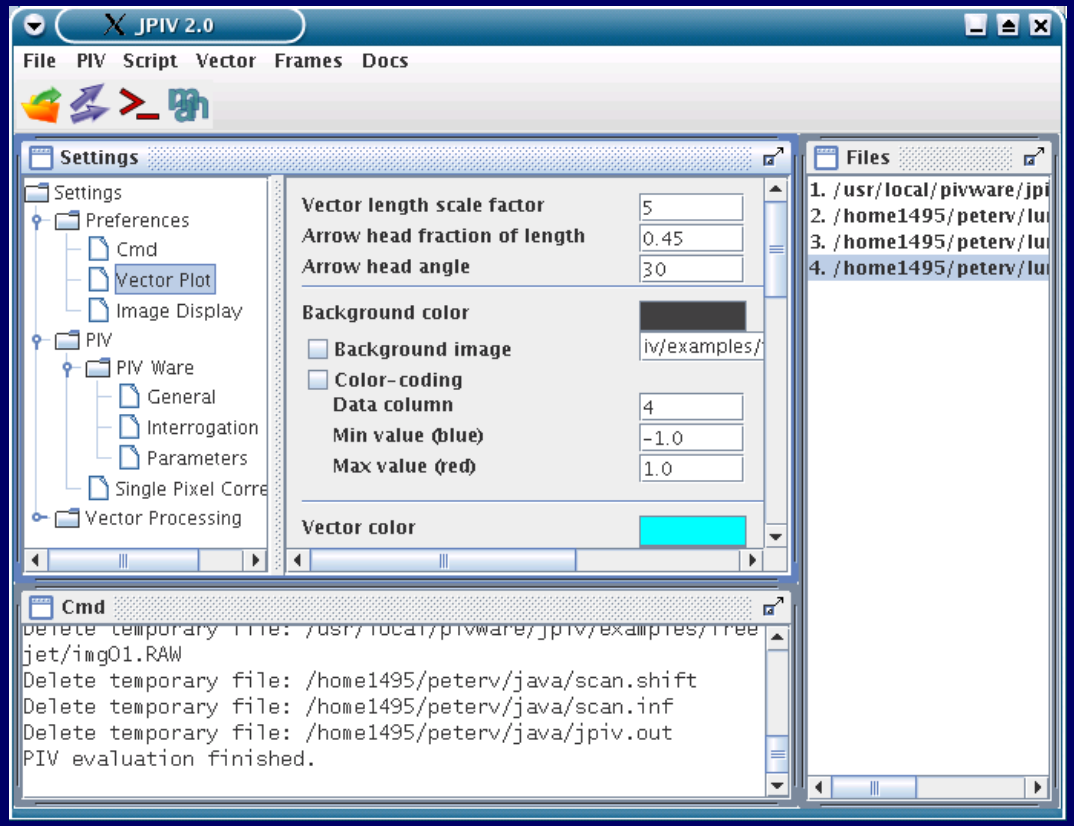
- strictly object oriented:
  extensible without knowing details of existing code
- syntax similar to c++:
  well known, no exotic syntax to learn
- platform independent:
  also runs on the measurement PC's
- extensive class library:
  sophisticated graphical user interface (Swing)
  advanced image processing (JAI)

remark: Java (Sun) and Java-Script (Netscape) have nothing in common!

Jpiv graphical user interface

settings organized in a tree-structure

text output and scripting



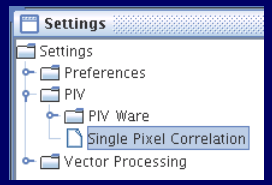
list holding interactive links to in- and output files

TU Delft

General procedure of data processing

1. 

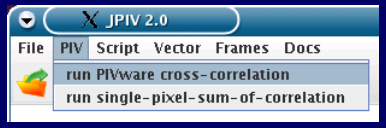
Load links to your image files

2. 

Modify the relevant settings

3. 

Select input files in the list frame

4. 

Do the processing

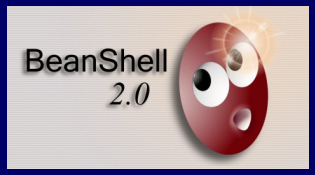
5. 

Display the result

TU Delft

Executing external software

- Bean Shell scripting for java implemented



- Jpiv class "CmdInterpreter" executes external applications and shell commands
- example:

```
ci = jpiv.getCmdInterpreter();
String[] cmd = {"date"};
ci.execute(cmd, ci.TYPE_SHELLCOMMAND);
```

output:

```
Wed Apr 19 12:47:14 CEST 2006
```

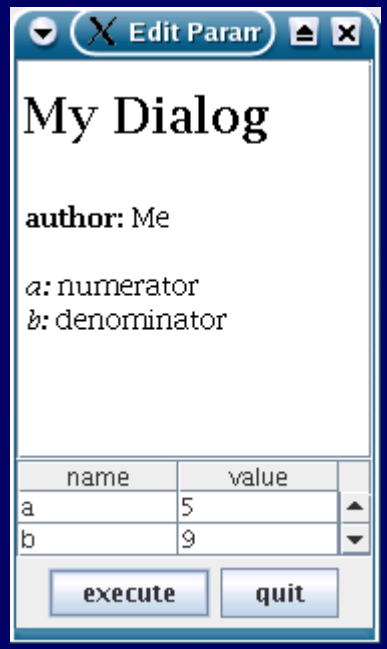
Configurable, graphical user dialog

- Class "ScriptDlg" brings up a configurable dialog
- example:

```
myDlg = new jpiv2.ScriptDlg(jpiv, "My Dialog", "Me");
myDlg.addVariable("a", "5", "numerator");
myDlg.addVariable("b", "9", "denominator");
proceed = myDlg.display();
if (proceed == myDlg.EXECUTE_OPTION) {
    a = myDlg.getAsDouble("a");
    b = myDlg.getAsDouble("b");
    System.out.println("a/b= " + a/b);
}
else {
    System.out.println("cancelled");
}
```

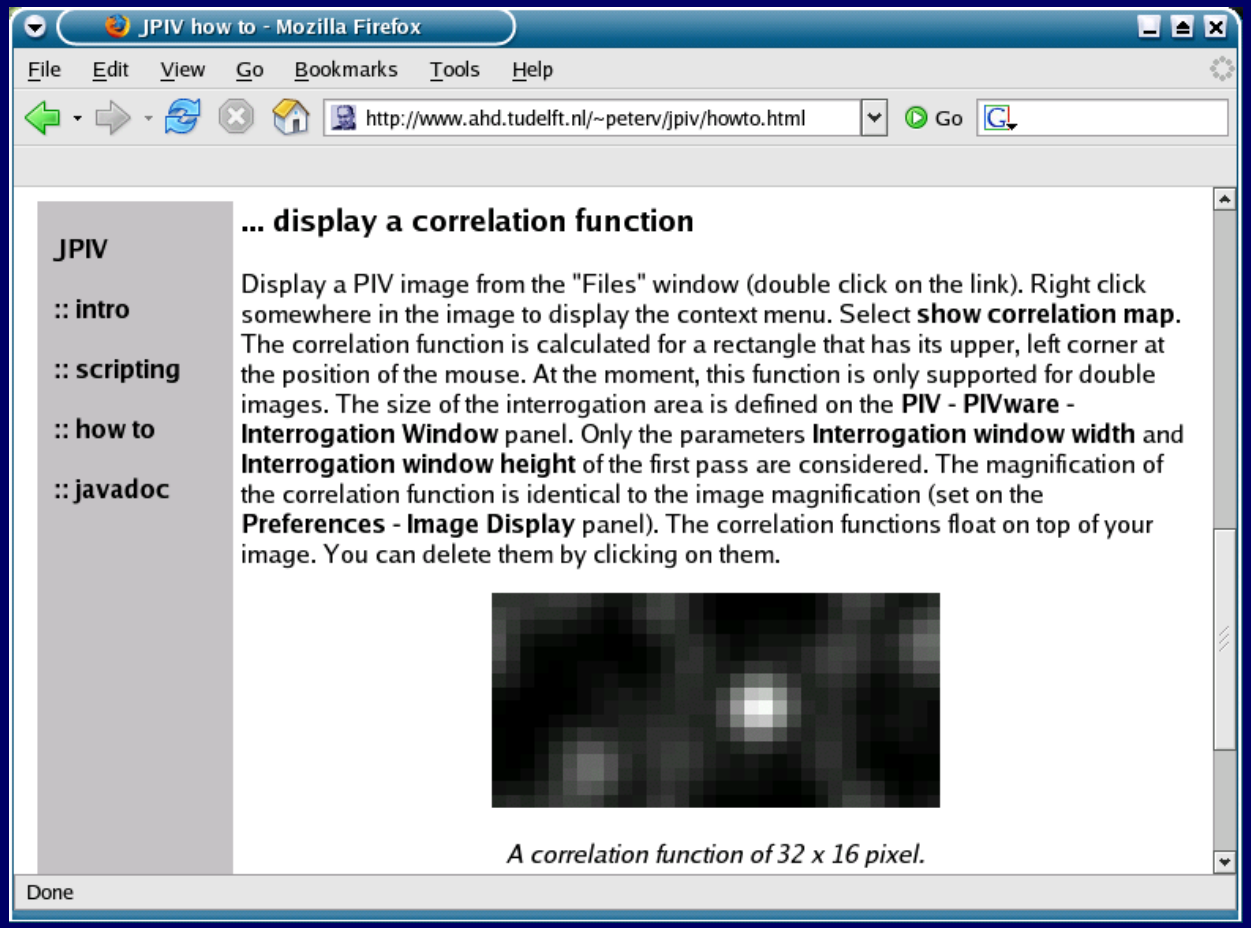
output:

```
a/b= 0.5555555555555556
```



TU Delft

Documentation



... visit the ahd web pages

TU Delft

... don't ask me!