Pan-STARRS Project Management System

IfA/MHPCC Image Processing Pipeline Pilot Project Software Issues and Code Evaluation

Paul Price (PoC)
Pan-STARRS Algorithms Group
Institute for Astronomy
10 December 2003
<table>
<thead>
<tr>
<th>Version/Revision</th>
<th>Date</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>3 Dec 2003</td>
<td></td>
</tr>
<tr>
<td>02</td>
<td>10 Dec 2003</td>
<td>Revised comments on some issues; removed redundancies with PSDC-400-001-01; Robert Lupton removed his use of the term “baroque”…</td>
</tr>
</tbody>
</table>
Table of Contents

1. Introduction .................................................................................................................. 4

2. Specific Issues, Examples, and Comments ............................................................... 5
  2.1 Positive Items ........................................................................................................... 5
  2.2 Compilation practice (Makefiles and such) ......................................................... 6
  2.3 Compilation warnings/errors .................................................................................... 10
  2.4 Unit test failures ....................................................................................................... 13
  2.5 Failed Requirements Following Testing .............................................................. 14
  2.6 Coding standard (Sun Java, C89/Posix 1003.2 infractions etc) ......................... 15
  2.7 Coding practice ........................................................................................................ 16
  2.8 Architecture ............................................................................................................ 24
  2.9 Documentation ......................................................................................................... 25
  2.10 Algorithms and Implementation ........................................................................... 27
1. Introduction

Below follows comments and issues raised at the IfA regarding the delivered version of the Image Processing Pipeline Pilot Project by MHPCC. These comments are intended to serve as a guide for MHPCC as to the expectations of the IfA Algorithms team, as well as to provide feedback that will assist in reaching our common goal of excellent quality code.

Some of the comments reflect the common view of the IfA Algorithms team, while some reflect more individual opinions; we have tried to indicate this where appropriate. There are really only a few different families of comments that manifest themselves in the hundred or so individual comments that can be summarized as follows:

A. Does not adhere to requirements/standards
Requirements and standards were agreed upon when starting the pilot project, but these were not met/followed. Examples of this include violations of the Sun Java coding standard and the C89 standard.

B. Programming standards
MHPCC should hold itself to agreed-upon standards of programming skill and practice, but this level was not apparent in some instances in the delivered code. Examples of this include failure to check return values and delivering code that fails Unit Tests.

C. Implementation of algorithms
IfA expects MHPCC to produce code that is both robust and efficient, but some implementations of algorithms were either incorrect or not optimal. Examples include division by zero, the calculation of mode, median and standard deviation, and the spline interpolation.

D. Compatibility problems
We use a variety of machine architectures at the IfA, ranging from Sun/Solaris to PC/Linux to Mac/OS-X. This variety of machines helps to identify bugs. There were some difficulties in compiling and running the IPP under some of the architectures.

E. User-friendly related issues
Based on detailed examination of the delivered code, together with our experience in scientific programming, we believe there are better ways to present the working software to the user. Consequently, we have several recommendations on messaging output as the software runs, coding efficiency and optimization, and code transparency.
2. Specific Issues, Examples, and Comments (as compiled by PAP, 3 December 2003)

The comments below have been annotated with the initials of the person that provided the comment (so further information may be obtained from the source), and the perceived "family" to which the comment belongs designated by the letter associate with the family as defined in Section 1; admittedly, this is often subjective. In what follows, we have broken out our comments into broad categories (in no particular order or priority):

1. Positive Items,
2. Compilation practice (e.g., Makefiles and such),
3. Compilation warnings/errors,
4. Unit test failures,
5. Failed Requirements following testing,
6. Coding standard (Sun Java, C89/Posix 1003.2 infractions),
7. Coding practice (Less major/stylistic issues),
8. Architecture,
9. Documentation,
10. Algorithms and Implementation.

Key to comment sources:
- RHL: Robert Lupton (rhl@astro.princeton.edu, 609-258-3811)
- PAP: Paul Price (price@ifa.hawaii.edu, 956-9845)
- HA: Herve Aussel (aussel@ifa.hawaii.edu, 956-6796)
- NK: Nick Kaiser (kaiser@ifa.hawaii.edu, 956-6898)
- RJ: Rob Jedicke (jedicke@ifa.hawaii.edu, 956-9841)
- JH: Josh Hoblitt (jhoblitt@ifa.hawaii.edu, 932-2345)
- MW: Mark Waterson (waterson@ifa.hawaii.edu, 876-7600x108)

2.1 Positive Items

2.1.1 Documentation is at a good level for a first release. (JH)
2.1.2 The documentation (esp. Doxygen) is very helpful for writing modules. (HA)

2.1.2 High level of code production. (HA)

2.1.3 Users' Manual makes the architecture clear. (RJ, NK)

2.1.4 It is clear that much thought has gone into the architecture, producing a system that is flexible, powerful and can be used to implement new modules at every camera level. (RJ)

2.1.5 IfA people can successfully understand the code and play around with it. It compiles and runs. (HA)

2.1.6 Module design (3 functions initModule, invokeModule, closeModule) makes the code easy to navigate after some practice (see the initModule to know what’s being worked on, and the invokeModule to see what’s been done). (HA)
2.2 Compilation practice (Makefiles and such)

2.2.1 {B}
Recursive calls to "make" should be "$(MAKE)", so that, e.g. "make -n" works.
[Solution: RHL patch] (RHL)

2.2.2 {B}
"SHELL" isn't specified (RHL)

2.2.3 {B}
Names of compiler and linker are specified in each Makefile instead of globally. (see also 2.2.8) (RHL)

2.2.4 {E}
Echos "Compiling bias.c" etc instead of "gcc -c -g2 -fpic
-L../../module/moduleAPI debias.c -o debias.o", making it difficult
to spot compiler errors. (RHL)

2.2.5 {A}
"-Wall" is not set for all compiles.
[Solution: RHL patch] (RHL)

2.2.6 {E}
Needs at least a summary line indicating:
(1) if make was successful
(2) version which was built
All the lines flowing from make are distracting. (RJ)

2.2.7 {B}
Assumes that "rm" isn't aliased to "rm -i" (should use "$(RM)"). (RHL)

2.2.8 {D}
The Makefiles hard-code the options; there is no support for setting some other default
without overriding the compiler. (RHL)

2.2.9 {B}
The dependencies in the Makefiles require IPP and opt to be checked out (or unpacked) in
the same place in the file system. (RHL)

2.2.10 {B}
$IPPROOT is set using: "IPPROOT=$(shell cd ..;pwd)", but it should be set absolutely, and
in one place. This also assumes that there are no symbolic links. (RHL)
2.2.11 {B}
There is no top-level Makefile to support, e.g., "make install". (RHL)

2.2.12 {E}
There is no "tags" target (while this wasn't in the specifications, it is a good idea):

`.PHONY : tags

tags :
    etags `find . -name \*.ch -o -name \*.cpp \) -print`

(RHL)

2.2.13 {D}
The build fails under:
Darwin Alibaba.local. 6.8 Darwin Kernel Version 6.8
gcc (GCC) 3.3 20030304
This is because "gcc -shared" isn't supported.
[Workaround; see also 2.2.8:
    find . -name Makefile -print | \
    while read f; do perl -pi -e 's/ -shared/ -bundle -undefined suppress -flat_namespace/' $f; done
]

(RHL)

2.2.14 {D}
-fpic doesn't work on Mac OS-X.
[Workaround; see also 2.2.8: -fPIC] (RHL)

2.2.15 {E}
There is a lot of stuff jammed into opt --- it should probably be split up. (RHL)

2.2.16 {E}
There are missing .cvsignore files:
    ? engine/IPP
    ? engine/wrapper
    ? engine/PARSER/lex.c
    ? engine/PARSER/parse.c
    ? engine/PARSER/test/TestParser
    ? module/RemoveBadPixels/test/.gdb_history
    ? module/SimpleTestModules/AddToImage.s
    ? module/debias/test/testimage.o00.T0.fits
    ? module/divideByFlat/debias.d
    ? module/divideByFlat/divideByFlat.d
    ? module/divideByFlat/overscanMeanScaling.d

(RHL)
2.2.17 {E}
Repeating a toplevel "make" command rebuilds engine/IPP and engine/wrapper unnecessarily. Some dependency is written incorrectly. (RHL)

2.2.18 {B}
There's no make target to run the tests. (RHL)

2.2.19 {B}
The UnitTest routines have to be run from their directories since they use relative paths. (RHL)

2.2.20 {E}
Files named "test" are not a good idea on Unix, since it's a shell built-in. (RHL)

2.2.21 {E}
The source file is readFITS.c but the module is readFits.so. We need a consistent convention for module names. (RHL)

2.2.22 {B}
module/debias wasn't added to the targets in the toplevel Makefile. Its Makefile doesn't have a distclean target. Its UnitTest is, however, run. (RHL)

2.2.23 {B}
There exists the potential for infinite recursion in Makefiles with the dataAPI target in src/Makefile, which reads:

```
dataAPI:
    cd $(IPPROOT)/src/engine/dataAPI ; make
```

but that directory doesn't exist (if I didn't "cvs update -d"), and the way that that rule is written, the Makefile doesn't test for success, causing the "cd" to fail, and the make is run again in the toplevel directory. The correct rule is:

```
dataAPI:
    cd $(IPPROOT)/src/engine/dataAPI && $(MAKE)
```

The "&&" means that the make will only be run if the cd succeeds. (RHL)

2.2.24 {D}
Supports "CC = cc", but the flags are gcc/Linux specific. (see also 2.2.8) (RHL)

2.2.25 {E}
src/Makefile doesn't simply recurse into modules, it goes into each submodule explicitly. (RHL)

2.2.26 {B}
src/Makefile doesn't test for the existence of ../docs (built by make docs) (RHL)
2.2.27 {E}
The -O2 builds don't also specify -g
(RHL)

2.2.28 {D}
Specifies "gcc -g2"; there is no "2" option to "-g", at least on gcc 2.96 or 3.3. (RHL)

2.2.29 {B}
There exists tests for debias, but none for the spline/median code. More generally, there are no tests for utilities used in the modules. (RHL)

2.2.30 {B}
The rel-1-0 release tarball doesn't have $Name$ expanded. The name rel-1-0 is a Branch, not a version; a branch tag does not identify a unique delivered version. (RHL)

2.2.31 {E}
module/divideByFlat/test/testimage.o00.T0.fits has no -kb sticky option. (RHL)

2.2.32 {B}
"make distclean" doesn't:
find . \( -name \*\*.o -o -name \*\*.so \) -print
./engine/wrapper.o
./module/debias/test/PrintImage.so
./module/debias/test/Test00SetImage.so
./module/divideByFlat/test/printAltImage.so
./module/divideByFlat/test/test00SetAltImage.so
(RHL)

2.2.33 {D}
Makefiles don’t work with the “vanilla” Solaris make; see also 2.2.8. (HA, RHL)

2.2.34 {D}
Does not compile with "icc". (see also 2.2.8) (HA)
2.3 Compilation warnings/errors

2.3.1 {A} Compiling writeFITS.c
writeFITS.c: In function `invokeModule':
writeFITS.c:104: warning: `bImage' might be used uninitialized in this function
writeFITS.c:105: warning: `siImage' might be used uninitialized in this function
writeFITS.c:106: warning: `iImage' might be used uninitialized in this function
writeFITS.c:107: warning: `fImage' might be used uninitialized in this function
writeFITS.c:108: warning: `image' might be used uninitialized in this function
(PAP on RH9/gcc)

2.3.2 {A} Compiling debias.c.
gcc -O2 -fpic -I/home/price/panstarrs/code/IPP/src/module/moduleAPI -c debias.c -o debias.o
debias.c: In function `invokeModule':
debias.c:211: warning: deprecated use of label at end of compound statement
(PAP on RH9)

2.3.3 {A} Compiling overscanMeanScaling.c.
gcc --pipe -Wall -O2 -I../moduleAPI -c overscanMeanScaling.c -o overscanMeanScaling.o
overscanMeanScaling.c: In function `invokeModule':
overscanMeanScaling.c:92: warning: `n' might be used uninitialized in this function
(PAP on RH9)

2.3.4 {A} Test00SetImage.c:118: warning: implicit declaration of function `strcmp'
Test00CheckImage.c:124: warning: implicit declaration of function `strcmp'
Test00CheckImage.c:137: warning: implicit declaration of function `sprintf'
PrintImage.c:43: warning: implicit declaration of function `printf'
Test01CheckParams.c:39: warning: implicit declaration of function `printf'
Test01CheckParams.c:177: warning: implicit declaration of function `strcmp'
IPP.c:372: warning: implicit declaration of function `yyparse'
lex.l:113: warning: implicit declaration of function `panic'
splineInterp.c:118: warning: implicit declaration of function `splineInterp'
splineInterp.c:142: warning: implicit declaration of function `powf'
[Some are missing system header files, some missing forward declarations.]
(RHL on Mac OS/X)
2.3.5 [A]
Test00CheckImage.c:30: warning: unused variable `PassFail'
Test01CheckParams.c:28: warning: unused variable `PassFail'
debias.c:91: warning: unused variable `errorMsg'
median.c:46: warning: unused variable `countClip'
median.c:51: warning: unused variable `s'
median.c:150: warning: unused variable `i'
[Unused variables. Mostly a source of noise.]
(RHL on Mac OS/X)

2.3.6 [A]
nan.h:24: warning: `__nan_union' defined but not used
[A home-brewed NaN system, with a header-static variable. But C99 has everything in
math.h (though not C89).]
(RHL on Mac OS/X)

2.3.7 [A]
debias.c:211: warning: deprecated use of label at end of compound statement
[An empty default case in a switch; not even a break.] (RHL on Mac OS/X)

2.3.8 [A]
median.c:92: warning: suggest parentheses around + or - inside shift
median.c:97: warning: suggest parentheses around + or - inside shift
[ Line is:
    j = (*maskCount)>>1+maskCountFirst;
This statement constains both redundant and missing parens.
This should be written:
    j = (*maskCount >> 1) + maskCountFirst;
As written, it parses as:
    j = *maskCount >> (1 + maskCountFirst);
]
(I don't _think_ that that's what is meant. The surprisingly low precedence of << and >> in C
is an inheritance from B).
(RHL on Mac OS/X)

2.3.9 [A]
splineInterp.c:164: warning: control reaches end of non-void function
[Solution: int piecewisePolyEval()]
(RHL on Mac OS/X)
2.3.10 {A}
  engine.c:73: warning: assignment discards qualifiers from pointer target type
  engine.c:79: warning: assignment discards qualifiers from pointer target type
  [dlError = dlerror(); and dlError isn't declared const char *.]
  (RHL on Mac OS/X)

2.3.11 {A}
  IPP.c:126: warning: assignment discards qualifiers from pointer target type
  IPP.c:369: warning: int format, pointer arg (arg 4)
  [dlError = dlerror(); and dlError isn't declared const char *.
  Using %d to print a FILE *]
  (RHL on Mac OS/X)

2.3.12 {A}
  lex.yy.c:674: warning: label `find_rule' defined but not used
  lex.yy.c:1626: warning: `yy_flex_realloc' defined but not used
  lex.yy.c:1109: warning: `yyunput' defined but not used
  lex.yy.c:1599: warning: `yy_flex_strlen' defined but not used
  [These are flex problems, and need to be filtered out in some way.] (RHL on Mac OS/X)

2.3.13 {A}
  lex.l:143: warning: return type defaults to `int'
  lex.l:160: warning: return type defaults to `int'
  lex.l:165: warning: return type defaults to `int'
  [lex.l indeed fails to pre-declare panic(), and to use pre-C89
default int types for functions syntax_error(), yyerror(), and panic().]
  lex.l:162: warning: control reaches end of non-void function
  [This is OK, the compiler is being dumb about exit(). But it needs to be filtered out
  (probably via a /* NOTREACHED */ comment).] (RHL on Mac OS/X)

2.3.14 {A}
  TestParser.c:20: warning: return type defaults to `int'
  [No declaration for main() (which is written in pre-C89 K&R style);]
  TestParser.c:32: warning: implicit declaration of function `yyparse'
  [no forward declaration for yyparse().]
  TestParser.c:34: warning: control reaches end of non-void function
  [Namely main()]
  (RHL on Mac OS/X)
2.4 Unit test failures

2.4.1 \{E\}
--- Testing: /home/price/panstarrs/code/IPP/src/engine/dataAPI/test
FAILED.
[Workaround: find . -name "UnitTest" -exec chmod a+x \{} \; ]
(PAP on RH9)

2.4.2 \{E\}
--- Testing: /home/price/panstarrs/code/IPP/src/engine/PARSER/test
FAILED.
[Workaround: find . -name "UnitTest" -exec chmod a+x \{} \; ]
(PAP on RH9)

2.4.3 \{E\}
--- Testing: /home/price/panstarrs/code/IPP/src/module/moduleAPI/test
FAILED.
[Workaround: find . -name "UnitTest" -exec chmod a+x \{} \; ]
(PAP on RH9)

2.4.4.1 \{E\}
--- Testing: /home/price/panstarrs/code/IPP/src/module/RemoveBadPixels/test
FAILED.
:: Parsing Pipeline Configuration File testConfig
:: Parsing successful!!!
:: Message: Reading the data from disk.
::
::
:: FATAL ERROR: Could not open file `../../../../testData/e00000.o00.T0.fits'. Exiting...
[Workaround: cvs co testData ]
(PAP on RH9)

2.4.4.2 \{B\}
Tests assume that testData is checked out in the same directory structure. (RHL)

2.4.5 \{A\}
IPP/src/module/RemoveBadPixels/test fails with a SEGV on Mac because the image sizes are different.
[Solution: SEGV avoided with a RHL patch that checks image sizes.] (RHL)

2.4.6 \{A\}
IPP/src/module/debias/test testConfig fails as it uses sqrtf and powf, which isn't in the C89 standard (it is in C99). In a pure C89 world, these modules fail to link. (RHL)
2.4.7 {D}
The IPP/src/engine/dataAPI/test tests fail as they don't see the psFree/psAlloc symbols (i.e. no init function is called to set the function pointers?). This is easily fixed, and probably linker specific. (RHL)

2.5 Failed Requirements Following Testing

2.5.1 {A}
SEGV in removeBadPixels. (RHL)

2.5.2 {A}
Sigma clipping does not divide by (n-1). (RHL)

2.5.3 {A}
SEGV produced on failed malloc(). (HA)

2.5.4 {A}
Endianness problems. (RHL)
2.6 Coding standard (Sun Java, C89/Posix 1003.2 infractions etc)

2.6.1 {A}
Assumes that dlopen() exists on machine. See also 2.2.8. (RHL)

2.6.2 {A}
Assumes that LD_LIBRARY_PATH is supported (not required by X/Open standard). See also 2.2.8. (RHL)

2.6.3 {A}
malloc.h isn't in the C89 standard.
[Solution: RHL patch --> stdlib.h] (RHL)

2.6.4 {A}
Braces {} around single-line blocks are missing (Sun standard). (RHL)

2.6.5 {A}
Each variable should be on a single line of its own and initialized at its declaration (Sun standard). (PAP)

2.6.6 {A}
dlfcn.h isn't in Posix (its an X/Open System Interfaces Extension to the Posix 1003.1-2001); hence it is not supported on the Mac OS/X 10.2.
[Workaround: Implementations exist, but need to be referenced with -L and -I in the makefiles, or installed in the Proper Places. See also 2.2.8.] (RHL)

2.6.7 {A}
Aren't variables starting with "_ " partially in the system namespace the C89 standard? E.g. _initModule. (RHL)

2.6.8 {E}
Isn't the usual convention that typedefs are UPPERCASE? (RHL)

2.6.9 {A}
Lots of K&R declarations, e.g.:
struct parse_tree *mktoken(), *mkpt(); (RHL)
2.7 Coding practice

2.7.1 {B}
Some files (e.g. PP/src/engine/dataAPI/engine.c) include stdlib.h twice. (RHL)

2.7.2 {B}
MAX_MODULES is defined in both parse.y and IPP.c
(RHL)

2.7.3 {B}
Use of sizeof(char) without realising that sizeof(char) == 1 by definition in C. (RHL)

2.7.4 {B}
Some loops that are iterating are written as "while" loops instead of "for" loops. This is partially a matter of personal style, but as written, I had to figure out why the natural idiom was not employed. (RHL)

2.7.5 {B}
Comments were cut and pasted without editing, leaving inappropriate comments. e.g. "fill in the image with the val const" is above a loop that *checks* the values. (RHL)

2.7.6 {B}
AddToImage.c:
```
    for(x=0;x<img->size.x;x++)
        for(y=0;y<img->size.y;y++)
            (img->image.f)[y][x] = (img->image.f)[y][x] + float_val;
```
* No {} around blocks (contrary to Sun coding standard)
* No dealiasing of pointers (optimisation problems)
* Redundant parentheses ("->" binds harder than "[]")
* Should be written with "+="
(RHL)

2.7.7 {B}
TestParser.c:
* No type for main
* No decent argument parsing (getopt)
* No check for fopen success
* No return value
(RHL)

2.7.8 {B}
No typedefs to encapsulate types --- uses "short int", but really mean an unsigned 16-bit int.
(RHL)
2.7.9 {B}
Lack of consistency throughout between "short" and "unsigned short". (RHL)

2.7.10 {B}
"const" is underused. (RHL)

2.7.11 {B}
In writeFITS.c, name creation: don't complain if the name is truncated when a buffer is overrun. (RHL)

2.7.12 {B}
Comments are very sparse. Variables aren't described. e.g. What are all the variables? What's going on in the program? Many variables aren't described at all (e.g. yy_buffer_state, parse_tree). Some structures have useless descriptions (e.g. MajorFrame - "The Major Frame Structure"). (PAP, RJ)

2.7.13 {E}
Debugging (i.e. InfoLevel=1) output is minimal --- contains only which modules start and stop. (PAP)

2.7.14 {A}
Metadata generated by the IPP is non-existent. The only metadata stored in image->metadata are the FITS headers that have been read in. Even these FITS headers are not complete, but are lacking the COMMENT fields. Metadata detailing software version and exit status of readFits are missing. Debias does not output software version, bias value or fit statistics. Furthermore, metadata is not outputted or saved. Result is failure to meet requirements regarding metadata (see 0.3). (HA)

2.7.15.1 {E}
Some consistent and documented system of variable and function naming should be adopted. (RJ)

2.7.15.3 {E}
What are the namespace conventions? (RHL)

2.7.15.4 {E}
Variable names should be standardized between different modules. (HA)

2.7.16 {B}
Since malloc and free are forbidden, all checked-in code should be grep-ed to ensure they don't include these calls. Quick look through code found many examples where the rules are not obeyed. Alternatively,
```c
#define calloc(N) extern
#define malloc(N) extern
#define realloc(P,N) extern
#define free(N) extern
```
in some suitable header file would allow this to be caught at compile time. (RJ, RHL)
2.7.17 {B}
When "malloc" is used instead of "psAlloc", the return value often isn't checked. (RHL)

2.7.18 {E}
Does MHPCC hold the copyright, as claimed? (RHL)

2.7.19 {B}
Assumes that there are no problems with _ and dlsym(), but there can be. dlsym() has NO standard definition of e.g. leading _, and the version bundled with, e.g., python appears to require an explicit specification. This needs to be addressed in loadModule; e.g.

```c
_initModule moduleInit = NULL;
_invokeModule moduleInvoke = NULL;
...
if (moduleDll) {
    moduleInit = (_initModule)dlsym(moduleDll,"initModule");
    if(!moduleInit) {
        moduleInit = (_initModule)dlsym(moduleDll,"_initModule");
    }
    if(moduleInit) {
        moduleInvoke = (_invokeModule)dlsym(moduleDll,"invokeModule");
        if(!moduleInvoke) {
            moduleInvoke = (_invokeModule)dlsym(moduleDll,"_invokeModule");
        }
    }
}(RHL)
```

2.7.20 {E}
The engineFcns stuff is odd. It allows the user to use different list iterators and memory wrappers for each module, despite the fact that they share data structures. This is flexibility that we will probably never employ, and makes the code hard to follow or to examine with a debugger. I suggest that psAlloc() and friends be made real functions, not function pointers. The IPP design makes it very hard to SWIG the modules without extra layers of code. It can be done, but it isn’t straightforward. The wrapper.c example is very incomplete. It doesn’t handle the issues related to passing data from other modules (the dataSpace stuff in IPP.c) (RHL)

2.7.21 {E}
The arguments to invokeModule() are pointers to lists. In some cases this is because the module needs to append to the lists (which may not exist). A cleaner implementation is to have data structure representing the head of the list that always exists, but may have no entries. This would remove the need for an extra level of indirection, and make more clear what is being done. (RHL)

2.7.22 {B}
The IPP parser silently skips # at the start of lines. (RHL)
2.7.23 {B}
psCreateCell() doesn't initialise Cell.type
(RHL)

2.7.24 {B}
Missing:
   extern int yylineno;
in lex.l
(RHL)

2.7.25 {E}
Why use the parse_tree stuff? Wouldn't it be just fine to use string-valued tokens in parse.y?  
(RHL)

2.7.26 {B}
There is no checking for extraneous parameters, e.g.,
   writeFits varName="image" filename="!testout.fits" foo="bar";
(RHL)

2.7.27 {B}
psDiGetDataItemAsInt assumes that the only special types are INT and FLOAT; if it enumerated all types then it would be protected against adding new types. (RHL)

2.7.28 {B}
In wrapper.c, the module to load is hard coded to
   /home/gusciora/dragon/PAN/CVS/IPP/bin/PrintImage.so
so it is hard to test this example. (RHL)

2.7.29 {B}
In wrapper.c:
   MyCell = (Cell *) malloc(sizeof(Cell));
   psCreateCellData(MyCell, INT_TYPE, CELL_WIDTH, CELL_HEIGHT);
Why not call a (C) constructor? Also, there's no check for malloc succeeding. (RHL)

2.7.30 {E}
Error messages from lexer are uninformative. (RHL)

2.7.31 {B}
There is no memory cleanup when parsing fails. This is made extra difficult by allocating memory for every token. (RHL)
2.7.32 \{B\}

IPP/src/FullUnitTest:

* the core module File::Find should have been used instead of a recursive subroutine
* inconsistent use of curly braces
* warnings are not enable (there are errors that cause warnings)
* strict pragma is not used
* line 5: environment variable assigned to $start_dir, an undeclared scalar that is used only once
* line 6: scalar $indent is initialized but not declared and is never used
* line 7: scalar $retval is initialized but not declared
* line 7: $retval should not be used as a global variable and instead should have the return value of &worm assigned to it
* line 8: return value of &worm is not checked
* line 20: incorrect use of local
* line 20: scalar $base_dir is undeclared but assigned to, this only works because $base_dir is surrounded by ()'s due to the incorrect use of local
* line 21: incorrect use of local
* line 21: @files, $i, and $pwd are not declared - $j is never used
* line 21: $retval should be declared as a lexical scalar within this scope
* line 25: $pwd should be declared as a lexical scalar within this scope
* line 28: back-ticks are used without '2>&1' so the standard error is not trapped
* line 29: logic error, $_ can be undefined when $? is non-zero
* line 29: short circuiting of regexes is unnecessary, use a single regex
* line 45: back-ticks are used without '2>&1' so the standard error is not trapped
* line 46: logic error, $_ can be undefined when $? is non-zero
* line 46: short circuiting of regexes is unnecessary, use a single regex
* line 49: chomp is called but a newline is appended on line 51
* line 57: return value of chdir is not checked, triggerable race condition
* line 58: return value of &worm is not checked
* line 59: return value of chdir is not checked, possible race condition
* line 63: $retval should be returned from &worm here

(JH)

2.7.33 \{E\}

IPP/src/module/fits/readFITS.c produces an error "not enough cells" when given too many:
"FATAL ERROR: FITS file doesn't contain enough cells (Expect 1 cells, found 64)) Exiting..."

(JH)
2.7.34 {B}
Error conditions should generate an error message, at least.
e.g.: overscanMeanScaling.c:
/* if no image data, then don't need to go further */
if(!psIsCellDataValid(cellin->data))
    return NULL;
If there is no image data, the program continues without uttering a word. Not clear what
should happen if this condition is met (documentation). (MW)

2.7.35 {B}
When fed an image 266x256 with image[1:256,1:256]=2 and image[257:166,1:256]=1, the
output contains NaN values (first line of the image, and some pixels in the second line).
[Bug report, issue #88] (HA)

2.7.36 {B}
readnoise parameter specified in configuration file as:
debias image="image" overscans=50 readnoise=10;
when printf-ed from inside, gives readNoise=0. Specifying it as:
debias image="image" overscans=50 readnoise=10.0;
gives the correct result, readNoise=10.0.
(PAP)

2.7.37 {B}
In wrapper.c:
#define IPP_NAME "$Name: "$
#define IPP_VERSION "$Revision: 1.1 "$
These are not referenced. (RHL)

2.7.38 {B}
Memory leak:
PsError* loadModule(Module** module, char* name,DataItemList* parameters) {
    ...
    *module = NULL;
Throws away possible existing value for *module. (RHL)

2.7.39 {B}
In loadModule(), if moduleInit fails, then there's a memory leak. (RHL)

2.7.40 {E}
In wrapper.c, the code checks for loadModule()'s non-NULL status (good), but then also
checks if the module has been returned. This isn't possible; and if it were, why check
dlerror() at the user level??? The user doesn't know that dllopen() was called. (RHL)
2.7.41 {B}
psCreateError() doesn't check that it's given a valid error level. Why are NORMAL and WARNING included but (apparently) never going to be used? (RHL)

2.7.42 {E}
In psCreateError(), what are errorID, and errorMessage? More accurately, where is the dictionary for errorID? (RHL)

2.7.43 {B}
moduleInit = (_initModule)dlsym(moduleDll,"initModule");
moduleInvoke = (_invokeModule)dlsym(moduleDll,"invokeModule");
if (moduleInvoke && moduleInit) {
    ...
} else {
    dlError = dleerror();
}
Note that this call to dleerror() returns the error related to invokeModule; we have no idea if initModule was found; and I suspect that if it is not found but invokeModule is found that dleerror()'s value is undefined. (RHL)

2.7.44 {B}
The config file:
Message   text="Reading";
ReadFits    filename="testout.fits";
Message     text="Done";
crashes IPP:
Program received signal EXC_BAD_ACCESS, Could not access memory.
0x00002334 in main (argc=2, argv=0xbffff908) at IPP.c:213
213     globalParametersEnd->next = params[cmd_ptr];
(RHL)

2.7.45 {B}
In at least readFITS.c, invokeModule() assumes that none of its input DataItemList** pointers are NULL. It would be nice to allow this; otherwise, I have to say:
    Void *null = NULL;
    ... &null, …
(RHL)

2.7.46 {B}
The readFits() code assumes that the input OTA has the proper type for the data in the file, and the correct number of cells, i.e., the engine has to create the data structure before it has seen the file; it should wait until it sees what sort of data is present, e.g., a Cell (NAXIS=2) or an OTA (NAXIS=3). (RHL)
enum DataUnit { /* these values match the FITS values */
    UNKNOWN_TYPE    =0, /**< the data unit is not defined */
    BYTE_TYPE       =8, /**< pixels are 8-bit unsigned chars */
    SHORT_INT_TYPE  =20, /**< pixels are 16-bit unsigned integers */
    INT_TYPE        =40, /**< pixels are 32-bit unsigned integers */
    FLOAT_TYPE      =-32 /**< pixels are 32-bit IEEE floating point */
};

What do these comments mean? They mostly aren't the BITPIX values. (RHL)
2.8 Architecture

2.8.1.1 {E}
Overall architecture is clever, but too clever. It makes the program difficult to debug, set breakpoints etc. Should rely on "traditional" API which allows compiler checking instead of the linked-list approach of passing parameters. (RHL)

2.8.1.2
The organizational structure provides flexibility for dynamically creating new steps in the analysis process at every camera level, which is clever and powerful. The UserManual makes it clear how the system is organized. (RJ)

2.8.3.1 {A}
SWIG interface is missing. (PAP)

2.8.3.2 {B}
API is essentially un-SWIGable without an extra layer of indirection. (RHL)

2.8.4 {E}
Does the engine check to ensure that modules do not produce the same output name? How do modules know the names of available input from other modules? Is this standardized somewhere? Seems like there should be some standard names for common I/O requirements. I see in the User's manual that "bad things will happen" in this situation. (RJ)

2.8.5 {E}
Shouldn't all modules be easily identified as such by name, e.g. modDebias? (RJ)
2.9 Documentation

2.9.1 {B}
In IPP Module User's Manual, under "Rules", it is stated that malloc() and free() are forbidden; psAlloc() and psFree() are to be used instead. Nevertheless, mixed signals are sent, since the very next line states that the user must explicitly psFree() any memory explicitly allocated with malloc(); see also under InvokeModule, where "a module programmer is responsible for freeing any data that was malloc'ed inside". These are likely typographical errors. (PAP)

2.9.2 {A}
No man pages. (NK)

2.9.3 {B}
There is no doxygen mainpage. (RJ)

2.9.4 {E}
HAVEDOT = YES in Doxyfile? (requires the dot tool), (RJ)

2.9.5 {B}
File descriptions are not illuminating, e.g. many state "Test module for the engine". Testing what? (RJ)

2.9.6 {B}
Some enumeration values are not documented, e.g. none of the values in DataType. (RJ)

2.9.7 {B}
Many functions and global variables lack doxygenation. (RJ)

2.9.8 {B}
Very few *.ch files have anything but the brief description line. (RJ)

2.9.9 {B}
Users_Manual links don't work (the first three at least). (RJ)

2.9.10 {A}
Copyright notices are in the source files but there is no inline licensing information or LICENSE file. (JH)

2.9.11 {A}
There are no inlined changes or change log file as required by the GPL. (JH)
2.9.12 {E}
The Doxygenfile specifies:
   PROJECT_NUMBER = Pilot Project
Should be $\text{Name}$
(RHL)

2.9.13 {E}
   TAB_SIZE = 4
Doesn't this in the Java coding standard require 8? (RHL)

2.9.14 {B}
   PAPER_TYPE = a4wide
USA doesn't use A4 paper. (RHL)

2.9.15 {E}
   PDF_HYPERLINKS = NO
Should probably be YES; but does this work with pdflatex? (RHL)

2.9.16 {E}
   USE_PDFLATEX = NO
Should be YES to get a higher quality PDF file. (RHL)
2.10 Algorithms and Implementation

2.10.1 {C}
RemoveBadPixels causes a SEGV if the images aren't the same size.
[Solution: RHL patch] (RHL)

2.10.2 {C}
Sigma clipping is naive:
* uses aliased pointers
* uses float not double;
* uses the $<x^2> - <x>^2$ without subtracting off the mean
* not dividing by (n - 1)
* not asserting *maskCount > 0;
* fails to calculate mode of integer data with many ties.
(RHL)

2.10.3 {C}
Implemented own quicksort, using recursive calls (!).
Previously used qsort(), but did not declare compar() static. (RHL)

2.10.4 {C}
Median code is being applied to integer data. In this case, the correct thing to do is to build a histogram and analyse this; in fact, all the image statistics should be done this way. (RHL)

2.10.5 {C}
splineInterp.c has:
#define TINY 1.0e-20
(RHL)

2.10.6 {C}
splineInterp.c:
    al = (float**)malloc((size_t)((sizeIn+1)*sizeof(float*)));
    al[1] = (float *)malloc((size_t)((sizeIn*3+1)*sizeof(float*)));
    if(!al){
        printf("Allocation Error\n");
        exit(1);
    }
Note SEGV before the test on !al
(RHL)
2.10.7 {C}

splineInterp.c:
    for (i=0;i<sizeIn-1;i++) {
        dx[i] = xIn[i+1]-xIn[i];
        divDiff[i] = (yIn[i+1]-yIn[i])/dx[i];
    }

Note potential division by 0. (RHL)

2.10.8 {C}
Solves the "natural spline problem", but doesn't say so. (RHL)

2.10.9 {C}
Divide by flat:
    * no check is performed against division by zero.
    * If the flat is masked (NaN), then the result is Inf (not IEEE compliant)
    * Input integer object frame, float flat frame --> integer output.

(HA)

2.10.10 {C}
Get the following warning when doing debias:
    WARNING debias:Bad readNoise - All bias RMS values are bad
This occurs even with the correct readNoise value. Two causes:
    * the standard deviation was being calculated incorrectly (mean of the
      square without subtracting the square of the mean)
    * threshold is set: if (stdDev > readNoise) --> bad, whereas a higher
      threshold should be used because the stdDev can easily be 10.05 if the
      readNoise is 10.0.

(PAP)