Cython tricks

Imports

```
from libcpp.vector cimport vector
from libcpp cimport bool
from libc.time cimport time_t, ctime
from libcpp.string cimport string

cimport numpy as np
import numpy as np
```

Useful ctypedefs

```
cdef extern from "<stdint.h>" nogil:
    ctypedef   signed char  int8_t
    ctypedef   signed short int16_t
    ctypedef   signed int   int32_t
    ctypedef   signed long  int64_t
    ctypedef unsigned char  uint8_t
    ctypedef unsigned short uint16_t
    ctypedef unsigned int   uint32_t
    ctypedef unsigned long  uint64_t

ctypedef unsigned shape_t
ctypedef uint16_t mask_t
ctypedef uint16_t extrim_t
ctypedef uint16_t pixstatus_t
ctypedef uint32_t conmap_t

#------------------------------
ctypedef fused nptype2d :
    np.ndarray[np.float64_t, ndim=2, mode="c"]
    np.ndarray[np.float32_t, ndim=2, mode="c"]
    np.ndarray[np.int16_t,   ndim=2, mode="c"]
    np.ndarray[np.int32_t,   ndim=2, mode="c"]
    np.ndarray[np.int64_t,   ndim=2, mode="c"]
    np.ndarray[np.uint16_t,  ndim=2, mode="c"]
    np.ndarray[np.uint32_t,  ndim=2, mode="c"]
    np.ndarray[np.uint64_t,  ndim=2, mode="c"]
```

Cython class attribute declaration and usage

```
cdef class image_algo :
    cdef uint16_t rows, cols

    def image_shape(self) :
        shape = (self.rows, self.cols)
```

C++ wrapper for python

Example of regular C++ class wrapper

```
cdef extern from "psalgos/PeakFinderAlgos.h" namespace "psalgos":
    cdef cppclass PeakFinderAlgos:
         #float  m_r0
         #float  m_dr
         #size_t m_rank
         #size_t m_pixgrp_max_size
         #size_t m_img_size
         #float  m_nsigm
         PeakFinderAlgos(const size_t& seg, const unsigned& pbits) except +
         void setPeakSelectionPars(const float& npix_min
                                   ,const float& npix_max
                                   ,const float& amax_thr
                                   ,const float& atot_thr
                                   ,const float& son_min)
         void peakFinderV3r3[T](const T *data
                               ,const mask_t *mask
                               ,const size_t& rows
                               ,const size_t& cols
                               ,const size_t& rank
                           ,const double& r0
                           ,const double& dr
                           ,const double& nsigm)
         void printParameters();


#------------------------------
cdef class peak_finder_algos :
    """ Python wrapper for C++ class.
    """
    cdef PeakFinderAlgos* cptr  # holds a C++ pointer to instance
    cdef uint16_t rows, cols

    def __cinit__(self, seg=0, pbits=0):
        #print "In peak_finder_algos.__cinit__"
        self.cptr = new PeakFinderAlgos(seg, pbits)

    def __dealloc__(self):
        #print "In peak_finder_algos.__dealloc__"
        del self.cptr

    def set_peak_selection_parameters(self\
                                      ,const float& npix_min\
                                      ,const float& npix_max\
                                      ,const float& amax_thr\
                                      ,const float& atot_thr\
                                      ,const float& son_min) :
        self.cptr.setPeakSelectionPars(npix_min, npix_max, amax_thr, atot_thr, son_min)


    def print_attributes(self) : self.cptr.printParameters()

    def peak_finder_v3r3_d2(self\
                           ,nptype2d data\
                           ,np.ndarray[mask_t, ndim=2, mode="c"] mask\
                           ,const size_t& rank\
                           ,const double& r0\
                           ,const double& dr\
                           ,const double& nsigm) :
        self.cptr.peakFinderV3r3(&data[0,0], &mask[0,0], data.shape[0], data.shape[1], rank, r0, dr, nsigm)
        self.rows = data.shape[0]
        self.cols = data.shape[1]
        return self.list_of_peaks_selected()

    def list_of_peaks_selected(self) :
        cdef vector[Peak] peaks = self.cptr.vectorOfPeaksSelected()
        return [py_peak.factory(p) for p in peaks]

    def list_of_peaks(self) :
        cdef vector[Peak] peaks = self.cptr.vectorOfPeaks()
        return [py_peak.factory(p) for p in peaks]
```

Wrapping struct

file.h:

```
#include <iostream> // for cout, ostream

struct Peak{
  float seg;
  float row;
  float col;

  Peak(){} // do not fill out member by default

  Peak(const float& _seg=0,
       const float& _row=0,
       const float& _col=0) :
       seg(_seg),
       row(_row),
       col(_col){ }

  //copy constructor
  Peak(const Peak& o)
    : seg(o.seg)
    , row(o.row)
    , col(o.col){}

  Peak& operator=(const Peak& rhs) {
    seg         = rhs.seg;
    row         = rhs.row;
    col         = rhs.col;
    return *this;
  }
};
/// Stream insertion operator,
std::ostream&
operator<<(std::ostream& os, const Peak& p); // needs to be implemented in file.cpp
```

file.cpp:

```
#include <sstream>   // for stringstream
#include <iomanip>   // for std::typedef

 std::ostream&
  operator<<(std::ostream& os, const Peak& p)
  {
    os << fixed
       << "Seg:"      << std::setw(3) << std::setprecision(0) << p.seg
       << " Row:"     << std::setw(4) << std::setprecision(0) << p.row
       << " Col:"     << std::setw(4) << std::setprecision(0) << p.col
    return os;
  }
```

file.pyx:

```
cdef extern from "psalgos/PeakFinderAlgos.h" namespace "psalgos":
    cdef cppclass Peak :
        Peak() except +
        Peak(const Peak& o) except +
        Peak operator=(const Peak& rhs) except +
        float seg
        float row
        float col

#------------------------------

cdef class py_peak :
    cdef Peak* cptr  # holds a C++ pointer to instance
    def __cinit__(self, _make_obj=True):
        if _make_obj:
            self.cptr = new Peak()

    def __dealloc__(self):
        if self.cptr is not NULL :
            del self.cptr
            self.cptr = NULL

    # https://groups.google.com/forum/#!topic/cython-users/39Nwqsksdto
    @staticmethod
    cdef factory(Peak cpp_obj):
        py_obj = py_peak.__new__(py_peak, _make_obj=False)
        (<py_peak> py_obj).cptr = new Peak(cpp_obj) # C++ copy constructor
        return py_obj

    def parameters(self) :
        p = self.cptr
        return (p.seg, p.row, p.col)

    @property
    def seg      (self) : return self.cptr.seg

    @property
    def row      (self) : return self.cptr.row

    @property
    def col      (self) : return self.cptr.col
```

Usage in other class, file.pyx:

```
    def list_of_peaks(self) :
        cdef vector[Peak] peaks = self.cptr.vectorOfPeaks()
        return [py_peak.factory(p) for p in peaks]
```

Recommended wrappers for numpy

```
cdef extern from "psalgos/LocalExtrema.h" namespace "localextrema":
    void mapOfLocalMinimums[T](const T *data
                              ,const mask_t *mask
                              ,const size_t& rows
                              ,const size_t& cols
                              ,const size_t& rank
                              ,extrim_t *arr2d
                              )

def local_minimums(nptype2d data,\
                   np.ndarray[mask_t, ndim=2, mode="c"] mask,\
                   int32_t rank,\
                   np.ndarray[extrim_t, ndim=2, mode="c"] arr2d\
                  ): mapOfLocalMinimums(&data[0,0], &mask[0,0], data.shape[0], data.shape[1], rank, &arr2d[0,0])
```