170317 smBIO meeting

서정훈, 허준석

ORIGINAL ARTICLE | OPEN ACCESS

A high-density 3D localization algorithm for stochastic optical reconstruction microscopy

Hazen Babcock ✉, Yaron M Sigal and Xiaowei Zhuang ✉

Optical Nanoscopy 2012 **1**:6 | **DOI:** 10.1186/2192-2853-1-6 | © Babcock et al.; licensee Springer. 2012
Received: 23 April 2012 | **Accepted:** 11 June 2012 | **Published:** 16 July 2012

Abstract

Background

Stochastic optical reconstruction microscopy (STORM) and related methods achieves sub-diffraction-limit image resolution through sequential activation and localization of individual fluorophores. The analysis of image data from these methods has typically been confined to the sparse activation regime where the density of activated fluorophores is sufficiently low such that there is minimal overlap between the images of adjacent emitters. Recently several methods have been reported for analyzing higher density data, allowing partial overlap between adjacent emitters. However, these methods have so far been limited to two-dimensional imaging, in which the point spread function (PSF) of each emitter is assumed to be identical.

Methods

In this work, we present a method to analyze high-density super-resolution data in three dimensions, where the

https://optnano.springeropen.com/articles/10.1186/2192-2853-1-6

https://github.com/ZhuangLab/storm-analysis

mikigom committed on **GitHub** Update fitting.py		Latest commit `36a13ad` 6 days ago
.eggs	설치 환경 및 동작 확인	7 days ago
build/lib/storm_analysis	ignore .pyc	7 days ago
dist	설치 환경 및 동작 확인	7 days ago
docker	init	7 days ago
storm_analysis.egg-info	설치 환경 및 동작 확인	7 days ago
storm_analysis	Update fitting.py	6 days ago
.gitignore	*.pyc ignore	7 days ago
.sconsign.dblite	설치 환경 및 동작 확인	7 days ago
MANIFEST.in	init	7 days ago
README.md	Update README.md	7 days ago
SConstruct	init	7 days ago
appveyor.yml	init	7 days ago
setup.cfg	init	7 days ago
setup.py	init	7 days ago

L1H	ignore .pyc	7 days ago
c_libraries	설치 환경 및 동작 확인	7 days ago
daostorm_3d	오타 수정	7 days ago
dbscan	ignore .pyc	7 days ago
decon_storm	init	7 days ago
fista	ignore .pyc	7 days ago
frc	ignore .pyc	7 days ago
rcc	ignore .pyc	7 days ago
rolling_ball_bgr	ignore .pyc	7 days ago
sCMOS	ignore .pyc	7 days ago
sa_library	Update fitting.py	6 days ago
sa_utilities	ignore .pyc	7 days ago
simulator	ignore .pyc	7 days ago
spliner	ignore .pyc	7 days ago
test	ignore .pyc	7 days ago
visualizer	init	7 days ago
voronoi	ignore .pyc	7 days ago
wavelet_bgr	ignore .pyc	7 days ago

L1H	ignore .pyc	7 days ago
c_libraries	설치 환경 및 동작 확인	7 days ago
daostorm_3d	오타 수정	7 days ago
dbscan	ignore .pyc	7 days ago
decon_storm	init	7 days ago
fista	ignore .pyc	7 days ago
frc	ignore .pyc	7 days ago
rcc	ignore .pyc	7 days ago
rolling_ball_bgr	ignore .pyc	7 days ago
sCMOS	ignore .pyc	7 days ago
sa_library	Update fitting.py	6 days ago
sa_utilities	ignore .pyc	7 days ago
simulator	ignore .pyc	7 days ago
spliner	ignore .pyc	7 days ago
test	ignore .pyc	7 days ago
visualizer	init	7 days ago
voronoi	ignore .pyc	7 days ago
wavelet_bgr	ignore .pyc	7 days ago

/storm_analysis/daostorm_3d/

```
mikigom@mikigom-desktop:~/github/smBIO-storm-analysis/storm_analysis/daostorm_3d
$ ls
batch_analysis.py           find_peaks.py       mufit_analysis.py
dual_cam_batch_analysis.py  __init__.py         README.md
example.xml                 LOW_SNR_README.txt  sample_data
```

/storm_analysis/daostorm_3d/

(2) .tif format data

```
$ python
>>> from storm_analysis.daostorm_3d.mufit_analysis import analyze
>>> analyze("comp.tif", "comp_mlist.bin", "3d_zfit.xml")
Peak finding
('Frame:', 0, 430, 430)
('Frame:', 1, 426, 856)

('Added', 856)

Tracking
Molecules: 856 (comp_mlist.bin)
Descriptor: 1
Processing molecule 0 in frame 0 (tracker)
Finished processing
Found 856 tracks
Analysis complete
```

/storm_analysis/daostorm_3d/

(2) .tif format data

```
$ python
>>> from storm_analysis.daostorm_3d.mufit_analysis import analyze
>>> analyze("comp.tif", "comp_mlist.bin", "3d_zfit.xml")
Peak finding
('Frame:', 0, 430, 430)
('Frame:', 1, 426, 856)

('Added', 856)

Tracking
Molecules: 856 (comp_mlist.bin)
Descriptor: 1
Processing molecule 0 in frame 0 (tracker)
Finished processing
Found 856 tracks
Analysis complete
```

Input output config

Note.

Input :

comp.dax is the STORM movie in .dax format

Or

comp.tif is the TIFF format file

/storm_analysis/daostorm_3d/

(2) .tif format data

```
$ python
>>> from storm_analysis.daostorm_3d.mufit_analysis import analyze
>>> analyze("comp.tif", "comp_mlist.bin", "3d_zfit.xml")
Peak finding
('Frame:', 0, 430, 430)
('Frame:', 1, 426, 856)

('Added', 856)

Tracking
Molecules: 856 (comp_mlist.bin)
Descriptor: 1
Processing molecule 0 in frame 0 (tracker)
Finished processing
Found 856 tracks
Analysis complete
```

Input output config

Note.

Output : Insight3 format (자체 포맷)

sa_utilities/bin_to_tagged_spot_file.py
: Insight3 -> Micro-Manager Format

Micro-Manager :
https://micro-manager.org/wiki/Download_Micro-Manager_Latest_Release

/storm_analysis/daostorm_3d/mufit_analysis.py

```python
import storm_analysis.daostorm_3d.find_peaks as find_peaks
import storm_analysis.sa_library.parameters as params
import storm_analysis.sa_utilities.std_analysis as std_analysis


def analyze(movie_name, mlist_name, settings_name):
    # parameters : setting_name file에서 parameters를 읽어들여 warpping하는 class
    #   parametrs.getAttr()로 구체적인 값에 접근
    parameters = params.ParametersDAO().initFromFile(settings_name)
    # find_peaks.py 내부에서는 sa_library/fitting.py 쪽으로부터 모든 메소드를 받고 있음
    # fitting 메소드마다의 방식을 정의해주는 것은 finder
    finder = find_peaks.initFindAndFit(parameters)
    std_analysis.standardAnalysis(finder,
                                  movie_name,
                                  mlist_name,
                                  parameters)
```

가장 상위 함수. finder는 finder와 fitter를 self로 함께 넘긴다.

/storm_analysis/daostorm_3d/find_peaks.py

```python
class DaostormFinderFitter(fitting.PeakFinderFitter):
    """
    Base class to encapsulate 3d-daostorm peak finding and fitting.
    """
    def __init__(self, parameters, peak_finder, peak_fitter):
        fitting.PeakFinderFitter.__init__(self, parameters)
        self.peak_finder = peak_finder
        self.peak_fitter = peak_fitter


def initFindAndFit(parameters):
    """
    Return the appropriate type of finder and fitter.
    """

    # Initialize finder.
    if (parameters.getAttr("filter_sigma", -1.0)  > 0.0):
        print("Using matched filter for peak finding.")
        finder = DaostormPeakFinder(parameters)
    else:
        finder = fitting.PeakFinder(parameters)

    # Initialize fitter.
    fitters = {'2dfixed' : Daostorm2DFixedFitter,
               '2d' : Daostorm2DFitter,
               '3d' : Daostorm3DFitter,
               'Z' : DaostormZFitter}
    fitter = fitters[parameters.getAttr("model")](parameters)

    return DaostormFinderFitter(parameters, finder, fitter)
```

```python
import storm_analysis.sa_library.fitting as fitting
import storm_analysis.sa_library.ia_utilities_c as utilC
import storm_analysis.sa_library.matched_filter_c as matchedFilterC
import storm_analysis.sa_library.dao_fit_c as daoFitC
import storm_analysis.simulator.draw_gaussians_c as dg
```

필요한 거의 모든 모듈을 다 import.

.xml에 따른 fitter 설정

L1H	ignore .pyc	7 days ago
c_libraries	설치 환경 및 동작 확인	7 days ago
daostorm_3d	오타 수정	7 days ago
dbscan	ignore .pyc	7 days ago
decon_storm	init	7 days ago
fista	ignore .pyc	7 days ago
frc	ignore .pyc	7 days ago
rcc	ignore .pyc	7 days ago
rolling_ball_bgr	ignore .pyc	7 days ago
sCMOS	ignore .pyc	7 days ago
sa_library	Update fitting.py	6 days ago
sa_utilities	ignore .pyc	7 days ago
simulator	ignore .pyc	7 days ago
spliner	ignore .pyc	7 days ago
test	ignore .pyc	7 days ago
visualizer	init	7 days ago
voronoi	ignore .pyc	7 days ago
wavelet_bgr	ignore .pyc	7 days ago

/storm_analysis/sa_library/

```
mikigom@mikigom-desktop:~/github/smBIO-storm-analysis/storm_analysis/sa_library$
 ls
arraytoimage.py     i3dtype.py            multi_fit.h
dao_fit.c           i3togrid.py           multi_fit.os
dao_fit_c.py        ia_utilities.c        parameters.py
dao_fit.os          ia_utilities_c.py     readhres.py
datareader.py       ia_utilities.os       readinsight3.py
daxwriter.py        imagecorrelation.py   README.txt
driftutilities.py   __init__.py           rebin.py
fitting.py          loadclib.py           recenter_psf.py
gaussfit.py         matched_filter.c      regfilereader.py
grid.c              matched_filter_c.py   static_background.py
grid_c.py           matched_filter.os     writeinsight3.py
grid.os             multi_fit.c
```

/storm_analysis/sa_library/fitting.py

```
    def analyzeImage(self, new_image, bg_estimate = None, save_residual = False, verbose = False):
        """
        image - The image to analyze.
        bg_estimate - (Optional) An estimate of the background.
        save_residual - (Optional) Save the residual image after peak fitting, default is False.

        return - [Found peaks, Image residual]
        """

        #
        # Pad out arrays so that we can better analyze localizations
        # near the edge of the original image.
        #
        image = padArray(new_image, self.margin)
        residual = padArray(new_image, self.margin)
        if bg_estimate is not None:
            bg_estimate = padArray(bg_estimate, self.margin)

        self.peak_finder.newImage(image)
        self.peak_fitter.newImage(image)
```

/storm_analysis/sa_utilites/std_analysis.py

```
def standardAnalysis(find_peaks, data_file, mlist_file, parameters):
    """
    Perform standard analysis.
    """

    # peak finding
    print("Peak finding")
    if(not peakFinding(find_peaks, data_file, mlist_file, parameters)):
        print("")
```

/storm_analysis/sa_library/fitting.py의 *analyzeImage*는

/storm_analysis/daostorm_3d/mufit_analysis.py를 통해

/storm_analysis/sa_utilities/std_analysis.py에서 호출

```
def peakFinding(find_peaks, movie_file, mlist_file, parameters):
    """
    Does the peak finding.
    """

    # open files for input & output
    movie_data = datareader.inferReader(movie_file)
    [movie_x,movie_y,movie_l] = movie_data.filmSize()

    # if the i3 file already exists, read it in,
    # write it out & start the analysis from the
    # end.
    total_peaks = 0
    if(os.path.exists(mlist_file)):
        print("Found", mlist_file)
        i3data_in = readinsight3.loadI3File(mlist_file)
        try:
```

```
# Find and fit the peaks.
if static_bg_estimator is not None:
    bg_estimate = static_bg_estimator.estimateBG(curf) - parameters.getAttr("baseline")
    [peaks, residual] = find_peaks.analyzeImage(image,
                                                bg_estimate = bg_estimate)
else:
    [peaks, residual] = find_peaks.analyzeImage(image)
```

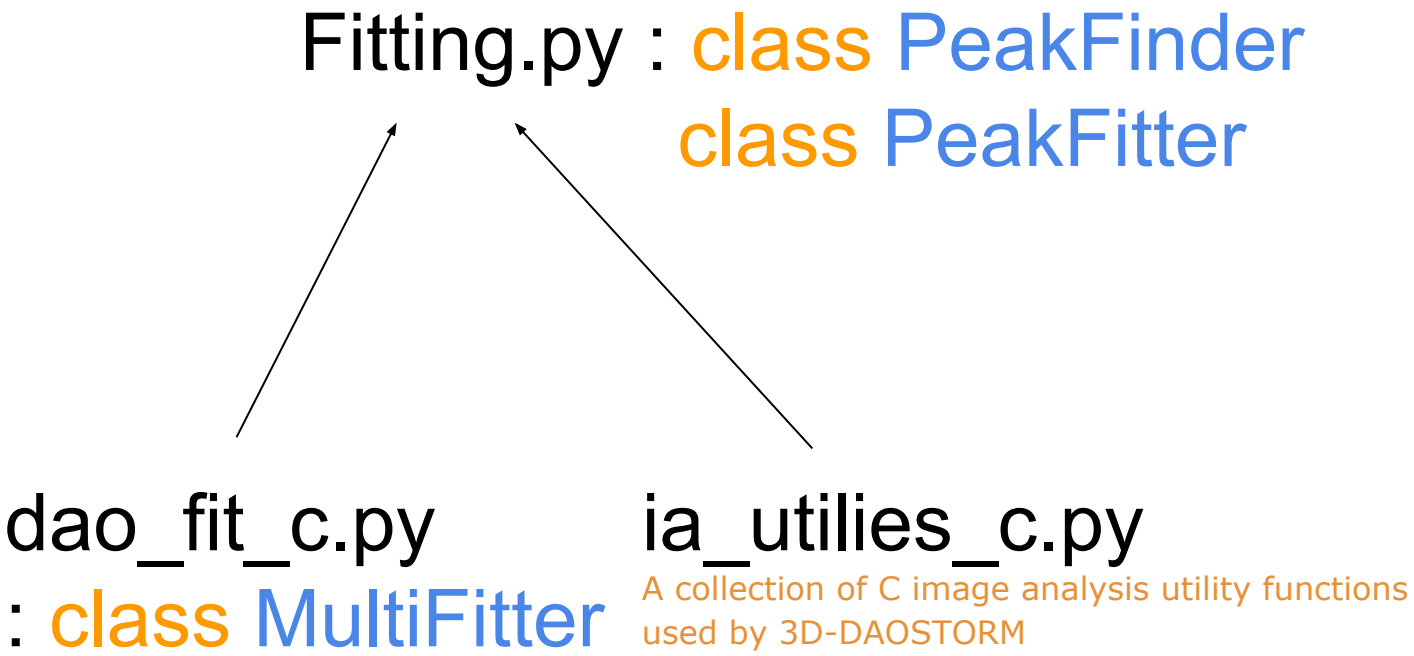
/storm_analysis/sa_library/fitting.py

```
    def analyzeImage(self, new_image, bg_estimate = None, save_residual = False, verbose = False):
        """
        image - The image to analyze.
        bg_estimate - (Optional) An estimate of the background.
        save_residual - (Optional) Save the residual image after peak fitting, default is False.

        return - [Found peaks, Image residual]
        """

        #
        # Pad out arrays so that we can better analyze localizations
        # near the edge of the original image.
        #
        image = padArray(new_image, self.margin)
        residual = padArray(new_image, self.margin)
        if bg_estimate is not None:
            bg_estimate = padArray(bg_estimate, self.margin)

        self.peak_finder.newImage(image)
        self.peak_fitter.newImage(image)
```


Fitting.py : class PeakFinder
class PeakFitter
dao_fit_c.py
: class MultiFitter
ia_utilies_c.py
A collection of C image analysis utility functions
used by 3D-DAOSTORM

0. Initialize the algorithm

0.1. 첫 threshold value를
Minimum peak height의 4배로 설정

```
<!-- threshold -->
<!-- This is basically the same as the minimum height
     parameter for peak finding in Insight3. -->
<threshold type="float">200.0</threshold>
```

/storm_analysis/daostorm_3d/example.xml

```
self.wx_params = None
self.wy_params = None

self.sigma = parameters.getAttr("sigma")                    # Peak sigma (in pixels).
self.threshold = parameters.getAttr("threshold")            # Peak minimum threshold (height, in camera units).

self.neighborhood = self.sigma*PeakFinder.unconverged_dist  # Radius for marking neighbors as unconverged.
```

/storm_analysis/sa_library/fitting.py 304 in class PeakFitter/__init__

```
# Initialize new peak minimum threshold.
if(self.iterations>4):
    self.cur_threshold = 4.0 * self.threshold
else:
    self.cur_threshold = float(self.iterations) * self.threshold
```

/storm_analysis/sa_library/fitting.py 176 in class PeakFinder/findPeaks

0. Initialize the algorithm

0.1. Residual image를 원본 사진으로 설정

```
#
# Pad out arrays so that we can better analyze localizations
# near the edge of the original image.
#
image = padArray(new_image, self.margin)
residual = padArray(new_image, self.margin)
if bg_estimate is not None:
    bg_estimate = padArray(bg_estimate, self.margin)
```

/storm_analysis/sa_library/fitting.py 391 in
class PeakFinderFitter/analyzeImage

1. New localization identification

1.1. Local Maximum

```
def peakFinder(self, no_bg_image):
    """
    This method does the actual peak finding.

    Override this if you want to change the peak finding behaviour.
    """
    # Mask the image so that peaks are only found in the AOI.
    masked_image = no_bg_image * self.peak_mask

    # Identify local maxima in the masked image.
    [new_peaks, self.taken] = utilC.findLocalMaxima(masked_image,
                                                    self.taken,
                                                    self.cur_threshold,
                                                    self.find_max_radius,
                                                    self.margin)
```

/storm_analysis/sa_library/fitting.py 231 in class PeakFinder/peakFinder

```
int findLocalMaxima(double *image, int *taken, double *peaks, double thresho
ld, double radius, int image_size_x, int image_size_y, int margin, int peak_
size)
{
  int cnts,i,j,k,l,m,max,r,t;
  double tmp;

  cnts = 0;
  r = (int)(radius+0.5);
  radius = radius*radius;
  // For all pixels
  for(i=margin;i<(image_size_y-margin);i++){
    for(j=margin;j<(image_size_x-margin);j++){
      m = i*image_size_x+j;
      tmp = image[m];
      // only if pixel is larger then threshold
      if(tmp>threshold){
    max = 1;
    k = -r;
    // For around all pixels
    while((k<=r)&&max){
      t = m+k*image_size_x;
```

/storm_analysis/sa_library/ia_utilities.c 72 in findLocalMaxima

1. New localization identification

1.2. 만약 새로운 localization이 없고, localization height threshold가 h_0라면 현재 localization 리스트를 반환하고 알고리즘을 멈춘다.

```
# Use pre-specified peak locations if available, e.g. bead cal
if self.peak_locations is not None:
    new_peaks = self.peak_locations

# Otherwise, identify local maxima in the image and initialize
else:
    new_peaks = self.peakFinder(no_bg_image)

# Update new peak identification threshold (if necessary).
# Also, while threshold is greater than min_threshold we
# are automatically not done.
found_new_peaks = False
if (self.cur_threshold > self.threshold):
    self.cur_threshold -= self.threshold
    found_new_peaks = True

# If we did not find any new peaks then we may be done.
if (new_peaks.shape[0] == 0):
    return [found_new_peaks, peaks]

# Add new peaks to the current list of peaks if it exists,
# otherwise these peaks become the current list.
if isinstance(peaks, numpy.ndarray):
    merged_peaks = utilC.mergeNewPeaks(peaks,
                                       new_peaks,
                                       self.new_peak_radius,
                                       self.neighborhood)

    # If none of the new peaks are valid then we may be done.
    if (merged_peaks.shape[0] == peaks.shape[0]):
        return [found_new_peaks, merged_peaks]
    else:
        return [True, merged_peaks]
else:
    return [True, new_peaks]
```

```
no_bg_image = self.peak_finder.subtractBackground(residual, bg_estimate)
[found_new_peaks, peaks] = self.peak_finder.findPeaks(no_bg_image, peaks)
if isinstance(peaks, numpy.ndarray):
    [peaks, residual] = self.peak_fitter.fitPeaks(peaks)

if verbose:
    if isinstance(peaks, numpy.ndarray):
        print(" peaks:", i, found_new_peaks, peaks.shape[0])
    else:
        print(" peaks:", i, found_new_peaks, "NA")

if not found_new_peaks:
    break
```

/storm_analysis/sa_library/fitting.py 405 in
class PeakFinderFitter/analyzeImage

1.3. threshold > h_0라면, threshold를 h_0만큼 줄인다.
1.4. Next Page

/storm_analysis/sa_library/fitting.py 154 in
class PeakFinder/findPeaks

1. New localization identification

1.4. 새로 발견된 localization 중, 현재의 모든 localization에서 적어도 1 pixel 이상 떨어져있는 localization만을 리스트에 추가시키고, “Running” flag를 붙인다.

```
for(i=0;i<num_new_peaks;i++){
  x = new_peaks[i*NPEAKPAR+XCENTER];
  y = new_peaks[i*NPEAKPAR+YCENTER];
  bad = 0;
  j = 0;
  while((j<num_in_peaks)&&(!bad)){
    dx = x - in_peaks[j*NPEAKPAR+XCENTER];
    dy = y - in_peaks[j*NPEAKPAR+YCENTER];
    rad = dx*dx+dy*dy;
    if(rad<radius){
  bad = 1;
    }
    // FIXME: This could mark as running peaks that are
    //   close to a bad peak which won't get added anyway.
    else if(rad<neighborhood){
  out_peaks[j*NPEAKPAR+ZCENTER] = 0.0;
  out_peaks[j*NPEAKPAR+STATUS] = RUNNING;
    }
    j++;
  }
  if(!bad){
    for(j=0;j<NPEAKPAR;j++){
  out_peaks[k*NPEAKPAR+j] = new_peaks[i*NPEAKPAR+j];
    }
    k++;
  }
}

k -= num_in_peaks;
return k;
```

/storm_analysis/sa_library/ia_utilities.py 287
in mergeNewPeaks

1. New localization identification

1.5. Parameter-dependent clamp values 초기화

```
# Default clamp parameters.
#
# These set the (initial) scale for how much these parameters
# can change in a single fitting iteration.
#
self.clamp = numpy.array([1000.0,   # Height
                          1.0,      # x position
                          0.3,      # width in x
                          1.0,      # y position
                          0.3,      # width in y
                          100.0,    # background
                          0.1])     # z position
```

/storm_analysis/sa_library/dao_fit_c.py 166
in class MultiFitter/__init__

2. Refining localization parameters

2.2. Elliptical Gaussian에 의해 fit image f를 계산한다. 오직 2.1에서 계산된 fitting neighborhood로만 계산된다.

```
def newImage(self, new_image):
    fitting.PeakFinder.newImage(self, new_image)

    # If does not already exist, create a gaussian filter object.
    if self.mfilter is None:
        psf = dg.drawGaussiansXY(new_image.shape,
                                 numpy.array([0.5*new_image.shape[0]]),
                                 numpy.array([0.5*new_image.shape[1]]),
                                 sa = self.filter_sigma)
        psf = psf/numpy.sum(psf)
        self.mfilter = matchedFilterC.MatchedFilter(psf)
```

2. Refining localization parameters

2.2. Elliptical Gaussian에 의해 fit image f를 계산한다. 오직 2.1에서 계산된 fitting neighborhood로만 계산된다.

```
class MatchedFilter(object):

    def __init__(self, psf, estimate_fft_plan = False):
        """
        If you are only going to use this object on a few images using
        estimate_fft_plan = True is a good idea as the initialization
        will go a lot faster, particularly for large images.
        """
        self.psf_shape = psf.shape

        rc_psf = recenterPSF.recenterPSF(psf)

        self.mfilter = m_filter.initialize(rc_psf, rc_psf.shape[0], rc_psf.shape[1], int(estimate_fft_plan))
```

/storm_analysis/sa_library/matched_filter_c.py 35 in class MatchedFilter/__init__

/sa_library/matched_filter.c에서는 fftw3.h(FFT 라이브러리) 가지고 뭘 계산을 하는데…
모르겠네요.

2. Refining localization Parameters

2.3. 모든 ‘Running’ flag Localization에 대해 fit error를 계산한다.

절대 오차가 tolerance보다 작으면 ‘Converged’로 표시한다.

```
if(peak->status == RUNNING){
  l = peak->yi * fit_data->image_size_x + peak->xi;
  err = 0.0;
  for(j=0;j<peak->size_y;j++){
    for(k=0;k<peak->size_x;k++){
  m = (j * fit_data->image_size_x) + k + l;
  fi = fit_data->f_data[m] + fit_data->bg_data[m] / ((double)fit_data->bg_counts[m]);
  if(fi <= 0.0){
    if(TESTING){
      printf(" Negative f detected! %.3f %d\n", fi, peak->index);
    }
    peak->status = ERROR;
    fit_data->n_neg_fi++;
    j = peak->size_y + 1;
    k = peak->size_x + 1;
  }
  xi = fit_data->x_data[m];
  /*
   * This should not happen as the expectation is that negative image
   * values are eliminated upstream of this step.
   */
  if(TESTING){
    if(xi <= 0.0){
      printf(" Negative x detected! %.3f %d\n", xi, m);
    }
  }
  err += 2*(fi-xi)-2*xi*log(fi/xi);
    }
  }
  peak->error_old = peak->error;
  peak->error = err;
  if (VERBOSE){
    printf("error: %d %f %f %f\n", peak->index, peak->error_old, peak->error, fit_data->tolerance);
  }
  if(((fabs(err - peak->error_old)/err) < fit_data->tolerance) && (peak->status != ERROR)){
    peak->status = CONVERGED;
  }
```

/sa_library/multi_fit.c 30 in mFitCalcErr

2. Refining localization
Parameters

2.4.1~2.4.3. Parameter update를 위해 각종 LA 연산을 함.

모두 /sa_library/dao_fit.c에서 이루어짐.
Jacobian, Hessian 연산은 native implementation(무지무지 길다),
U 업데이트를 위한 Solver는 dposv_라는 extern 함수로 LAPACK 사용

2. Refining localization Parameters

2.4.5 Parameter update.

대신, Oscillation을 방지하기 위해 몇 가지 연산을 걸어준다.

/sa_library/multi_fit.c 269 in mFitUpdateParams

```
void mFitUpdateParams(peakData *peak, double *delta)
{
  int i;

  if(VERBOSE){
    printf("%d : ", peak->index);
  }
  for(i=0;i<NFITTING;i++){

    if(VERBOSE){
      printf("%.3e %.3f | ", delta[i], peak->clamp[i]);
    }

    if (delta[i] != 0.0){

      // update sign & clamp if the solution appears to be oscillating.
      if (peak->sign[i] != 0){
    if ((peak->sign[i] == 1) && (delta[i] < 0.0)){
      peak->clamp[i] *= 0.5;
    }
    else if ((peak->sign[i] == -1) && (delta[i] > 0.0)){
      peak->clamp[i] *= 0.5;
    }
      }
      if (delta[i] > 0.0){
    peak->sign[i] = 1;
      }
      else {
    peak->sign[i] = -1;
      }

      peak->params[i] -= delta[i]/(1.0 + fabs(delta[i])/peak->clamp[i]);
    }
  }
  if(VERBOSE){
    printf("\n");
  }
}
```

2. Refining localization Parameters

2.4.6 Negative 값이 나온 localization들을 “BAD” 표시 해준다. 이 값들은 다음엔 무시된다.

/sa_library/dao_fit.c 324 in fitDataUpdate

```
  /*
   * Check that the peak hasn't moved to close to the
   * edge of the image. Flag the peak as bad if it has.
   */
  margin = fit_data->margin;
  xc = dao_peak->xc;
  yc = dao_peak->yc;
  if((xc <= margin)||(xc >= (fit_data->image_size_x - margin - 1))||(yc <=
margin)||(yc >= (fit_data->image_size_y - margin - 1))){
    peak->status = BADPEAK;
    fit_data->n_margin++;
    if(TESTING){
      printf("object outside margins, %.3f, %.3f\n", peak->params[XCENTER],
 peak->params[YCENTER]);
    }
  }

  /*
   * Check for negative height.
   */
  if(peak->params[HEIGHT]<0.0){
    peak->status = BADPEAK;
    fit_data->n_neg_height++;
    if(TESTING){
      printf("negative height, %.3f, %.3f (%.3f, %.3f)\n", peak->params[BAC
KGROUND], peak->params[HEIGHT], peak->params[XCENTER], peak->params[YCENTER
]);
    }
  }

  /*
   * Check for negative widths
   */
  if((peak->params[XWIDTH]<0.0)||(peak->params[YWIDTH]<0.0)){
    peak->status = BADPEAK;
    fit_data->n_neg_width++;
    if(TESTING){
```

3. Localization cleanup

3.1 'converged'나 'running'을 포함한 localization list를 만든다. 이때, 새로운 localization은 0.9*h_0 이상의 hight를, 0.5*sigma 이상의 width를 가져야한다.

3.2 주변 localization과 비교하여 sigma보다 작은 localization은 localization list에서 제외된다. 이때, 제거된 localization 주변에 있는 localization은 'running'으로 표시된다.

3.3 step 2를 다시 실행한다.

```
def fitPeaks(self, peaks):
    """
    Performs a single iteration of peak fitting.

    peaks - A numpy array of peaks to fit.

    return - [updated peaks, updated residual]
    """

    # Fit to update peak locations.
    [fit_peaks, residual] = self.peakFitter(peaks)
    fit_peaks = self.mfitter.getGoodPeaks(fit_peaks,
                                          0.9*self.threshold,
                                          0.5*self.sigma)

    # Remove peaks that are too close to each other & refit.
    fit_peaks = utilC.removeClosePeaks(fit_peaks, self.sigma, self.neighborhood)
    [fit_peaks, residual] = self.peakFitter(fit_peaks)

    fit_peaks = self.mfitter.getGoodPeaks(fit_peaks,
                                          0.9 * self.threshold,
                                          0.5 * self.sigma)

    return [fit_peaks, residual]
```

/storm_analysis/sa_library/fitting.py 316 in class PeakFitter/fitPeaks

4. Update the residual image

4.1. 원본 image에서 fit image를 뺀 background image를 추정한다. 그리고 background image에 8 픽셀의 시그마로 gaussian filter를 건다.

논문과 코드가 다름!

실제로는 gaussian filter만 걸고 있음.

```
def estimateBackground(image, size = 8):
    """
    A simple background estimator.

    image - A 2D numpy array containing the image.
    size - (Optional) The size of the filter to use, default is 8.

    returns - A 2D numpy array containing the image background estimate.
    """

    # scipy.ndimage.filters.gaussian_filter(input, sigma, order=0, output=No
=4.0)
    # See https://docs.scipy.org/doc/scipy-0.16.1/reference/generated/scipy.
cipy-ndimage-filters-gaussian-filter also.
    background = scipy.ndimage.filters.gaussian_filter(image, (size, size))
    return background
```

/storm_analysis/sa_library/fitting.py 25 in estimateBackground

```
jw@jw-Lenovo-ideapad-110-15ISK:~/github/smBIO-storm-analysis/storm_analysis$ grep -nr "backgroundEstimator" .
./sa_library/fitting.py:74:    class and then override backgroundEstimator() and peakFinder().
./sa_library/fitting.py:143:    def backgroundEstimator(self, image):
./sa_library/fitting.py:274:            self.background = self.backgroundEstimator(image)
./sCMOS/find_peaks.py:78:          self.background = self.backgroundEstimator(image)
```

backgroundEstimator()나 self.background를 override하는 다른 클래스나 메소드를 찾을 수 없음

4. Update the residual image

4.2.1. 원본 image에서 fit image를 뺀 것을 residual image로 설정한다.

```c
/*
 * mFitGetResidual(residual).
 *
 * Returns image - fit.
 *
 * fit_data - Pointer to a fitData structure.
 * residual - Pre-allocated space to store the residual values.
 *            This should be square & the same size as the image.
 */
void mFitGetResidual(fitData *fit_data, double *residual)
{
  int i;

  //calcFit(fit_data);
  for(i=0;i<(fit_data->image_size_x * fit_data->image_size_y);i++){
    residual[i] = fit_data->x_data[i] - fit_data->f_data[i];
  }
}
```

/storm_analysis/sa_library/multi_fit.c 85 in mFitGetResidual

```python
def peakFitter(self, peaks):
    """
    This method does the actual peak fitting.
    """
    fit_peaks = self.mfitter.doFit(peaks)
    residual = self.mfitter.getResidual()
    return [fit_peaks, residual]
```

/storm_analysis/sa_library/fitting.py 347 in class PeakFitter/peakFitter

4. Update the residual image

4.2.2. residual image의 평균값을 계산한다.
4.2.4. 4.2.2에서 계산한 평균값을 residual image에 더한다.

```
void mFitCalcErr(fitData *fit_data, peakData *peak)
{
  int j,k,l,m;
  double err,fi,xi;

  if(peak->status == RUNNING){
    l = peak->yi * fit_data->image_size_x + peak->xi;
    err = 0.0;
    for(j=0;j<peak->size_y;j++){
      for(k=0;k<peak->size_x;k++){
    m = (j * fit_data->image_size_x) + k + l;
    fi = fit_data->f_data[m] + fit_data->bg_data[m] / ((double)fit_data->bg_counts[m]);
    if(fi <= 0.0){
      if(TESTING){
        printf(" Negative f detected! %.3f %d\n", fi, peak->index);
      }
      peak->status = ERROR;
      fit_data->n_neg_fi++;
      j = peak->size_y + 1;
      k = peak->size_x + 1;
    }
    xi = fit_data->x_data[m];
```

/sa_library/multi_fit.c 41 in mFitCalcErr

4. Update the residual image

4.2.3. residual image에서 추정된 background image를 뺀다.

```
no_bg_image = self.peak_finder.subtractBackground(residual, bg_estimate)
[found_new_peaks, peaks] = self.peak_finder.findPeaks(no_bg_image, peaks)
if isinstance(peaks, numpy.ndarray):
    [peaks, residual] = self.peak_fitter.fitPeaks(peaks)
```

/storm_analysis/sa_library/fitting.py 409 in class PeakFitter/fitPeaks

```
def subtractBackground(self, image, bg_estimate):
    """
    This subtracts a smoothed background out of an image. As a side
    effect it also updates self.background.

    image - The image to subtract the background from.
    bg_estimate - An estimate of the background.

    return - The image with the background subtracted.
    """

    # If we are provided with an estimate of the background
    # then just use it.
    if bg_estimate is not None:
        self.background = bg_estimate

    # Otherwise make our own estimate.
    else:
        self.background = self.backgroundEstimator(image)

    return image - self.background
```

/storm_analysis/sa_library/fitting.py 316 in class PeakFitter/fitPeaks

5. Termination of the algorithm

Step 1로 돌아가서 다시 반복한다. Iteration 설정 횟수만큼 반복한다. 1.2에 따라, 새로운 localization이 없다면 바로 loop를 종료한다.

```
for i in range(self.peak_finder.iterations):
    if save_residual:
        resid_dax.addFrame(residual)

    no_bg_image = self.peak_finder.subtractBackground(residual, bg_estimate)
    [found_new_peaks, peaks] = self.peak_finder.findPeaks(no_bg_image, peaks)
    if isinstance(peaks, numpy.ndarray):
        [peaks, residual] = self.peak_fitter.fitPeaks(peaks)

    if verbose:
        if isinstance(peaks, numpy.ndarray):
            print(" peaks:", i, found_new_peaks, peaks.shape[0])
        else:
            print(" peaks:", i, found_new_peaks, "NA")

    if not found_new_peaks:
        break
```

/storm_analysis/sa_library/fitting.py 154 in
class PeakFinder/findPeaks

L1H	ignore .pyc	7 days ago
c_libraries	설치 환경 및 동작 확인	7 days ago
daostorm_3d	오타 수정	7 days ago
dbscan	ignore .pyc	7 days ago
decon_storm	init	7 days ago
fista	ignore .pyc	7 days ago
frc	ignore .pyc	7 days ago
rcc	ignore .pyc	7 days ago
rolling_ball_bgr	ignore .pyc	7 days ago
sCMOS	ignore .pyc	7 days ago
sa_library	Update fitting.py	6 days ago
sa_utilities	ignore .pyc	7 days ago
simulator	ignore .pyc	7 days ago
spliner	ignore .pyc	7 days ago
test	ignore .pyc	7 days ago
visualizer	init	7 days ago
voronoi	ignore .pyc	7 days ago
wavelet_bgr	ignore .pyc	7 days ago

/storm_analysis/visualizer/

```
jw@jw-Lenovo-ideapad-110-15ISK:~/github/smBIO-storm-analysis/storm_analysis/visualizer$ ls
data          qtRangeSlider.py  test_low_snr_list.bin  visualizer.ui
__pycache__   README.txt        visualizer.py          visualizer_ui.py
```

visualizer.py : PyQt5 기반 GUI Python 프로그램

1) .dax, .spe, .tif format의 movie 파일
2) .bin format의 Insight3 파일
3) .bin format의 storm 파일

등을 읽을 수 있게 설계.

근데 버그 있어요