CPT Project

Generated by Doxygen 1.8.6

Fri Jul 28 2017 18:06:11

Contents

1 Main Page

This project provides library, test, and examples of several topic of microarray process include :

- Chip Image Process
- genotyping

Each of these topic has a well description, API reference in this document.

2 Chip Image Process

To extract the probe intensities from the image. There are several issues of the image we need to resolve:

1. The images are slanted
2. Noise
3. Grid recognition and segmentation
4. A chip sample is captured into multiple images, which need to be stitched.

5. The region of interest detection, remove the regions we don't need. etc.

To solve these issues, we develop a pipeline with following steps:

- Image Rotation
- Image Gridding
- Image Segmentation
- Image Min CV Segmentation
- Image Region of Interest Detection
- Image Background Fix
- Image Stitching

2.1 Image Rotation

Algorithm main input :

- image data (matrix).
- grid image (optional).

Algorithm workflow:

1. Select a source for rotation estimation. If grid image is provided, then grid image will be selected.
2. Strengthen the grid edge (If no grid image provided)
   (a) Blur the image
   (b) Discrete Fourier transform
   (c) Apply north filter (shadow filter)
   (d) Inverse discrete Fourier transform
   (e) Apply south filter (shadow filter)
   (f) Apply north and south filter 3 times
   (g) Normalize and binarize
3. Strengthen the grid edge (If grid image provided)
   (a) Normalize
   (b) binarize
4. Image Hough transform
5. Estimate the entropy of all theta in histogram, and select minimum angle
6. Output the selected angle

detail desciption of Hough Transform can see here Hough transform

class reference cpt::improc::RotationEstimation cpt::improc::RotationCalibration

2.1.1 Hough transform

Hough transform is a generatic soultion for finding a line on image.

A line in Cartiesian coordinate system can be descibed as this form :

$y = ax + b$

In this form, some special case like the parameter (a, b) of a vertical line is infinite, which is hard to search and implement in program.

Relatively, Polar coordinate system provieds a limited parameter space to describe a 2 dimension space, witch has good property for quantile searching and implementation.

The formula of mapping Cartiesian coordinate system to Polar coordinate system is :

$x = rcos(\theta)$

$y = rsin(\theta)$

By the formula, every point of Cartiesian coordinate system can be mapping to a curve on Polar coordinate system, and every line can be map to a point.

Therefore, if there are multiple points in Cartiesian coordinate system can be connected into a line, then there will be multiple curves in Polar coordinate system overlapped at a point.

By these description, a line searching algorithm can be described :

1. Find the points' intensity higher than some threshold on the image
2. Mapping the points to Polar coordinate system and accumulate every (r, θ)'s count
3. Return the max count point on Polar coordinate system.

The (r, θ) of the point is the line we need.

For image rotation, the θ is the rotation angle of image.

2.2 Image Gridding

Algorithm main input:

- Image source
- The upper bound of the grid line interval

Algorithm workflow:

1. Fit sine wave with 2 directions (x, y)
   (a) Projection the image to 1 dimension with given direction
   (b) Discrete Fourier transform, extract the frequency.
   (c) Fit the sine wave by using linear regression and extract the phase.
   $D = Asin(\omega t + \phi) + C$
   (d) Generate the grid line.
2. Collect the grid lines and create tiles of grid
3. Return tiles and column, row number.

class reference cpt::improc::Gridding

2.3 Image Segmentation

Algorithm main input:

- Image source
- Margin information

Algorithm workflow:

This algorithm shrinks down all cells of the grid.

The reason we do this adjustment is because the gridding step may not perfectly segment the cells, it may have some signal overflow to the neighboring cells. This process sampling the center of the cell to avoid the probe intensity cross talked.

class reference cpt::improc::Segmentation

2.4 Image Min CV Segmentation

Algotithm main input:

- raw image
- image grid tiles
- new tile width and height after margin

Algorithm workflow:

1. For each tiles, scan the pixels in tile by sliding windows with new tile width and height and compute the CV for every windows.
2. Select the window which has minimun CV to be the new tile.

Note that, $CV = stddev/mean$

2.5 Image Region of Interest Detection

Algorithm main input :

- marker information
- intensities grid
- grid coordinate system

Algorithm workflow:

1. Detect the marker by pattern match
2. Bound the exact intensities region of the intensities grid by the marker pattern.

   For example:

   If the marker size is 10$*$10, the interval of marker 116 and there are 2$*$2 markers ver on intensities grid then the algorithm will try to bound a $126 * 126$ region on the grid.

3. Fix the coordinate system, transform the grid to make sure the probe position ordered start from the left top.

   To do this step is because that the image's probe coordinate system may not the same matrix, and the origin position of the matrix in OpenCV library is starting from the left top and row major. To make sure coordinate and probe intensities is matched, this step is necessary.

4. Extract the bounded intensities.

5. Write back to the Intensities grid.

   class reference cpt::improc::ROIDetection

2.6 Image Background Fix

Algotithm main input:

- probe grid (grid after ROI)
- raw image (image after ROI)
- marker information, include height, width, x and y direction interval
- the local segment number in x and y direction, which denote as sxn, syn.
- local background percentage, which denote as p

Algorithm workflow:

1. Use marker information to filter the marker probe on probe grid.
2. Split the raw image and probe grid into sxn $*$ syn segment. For example, let sxn=4, syn=4 and probe gird width=16, height=20, then the probe grid will be segment into 16 pieces and every pieces has width=4 and height=5.
3. For each grid segment, select the lowest p percentage probes' intensities in probe grid, and compute the mean of these probe intensities, the result is the local background.
4. For each pixel v of related segment in raw image, fix the pixel v by: $v = (v - localbackground)/localbackground$
5. The local background will then used by global background process.
   (a) Compute the mean of all locol backgrounds, which represent the global mean.
   (b) For each pixel v, multiply the global mean and the result is the final background fixed image of this algorithm.

2.7 Image Stitching

Algorithm main input:

- intensities grids
- every image most left top marker's probe absolute position.

  For example :

  If the marker size is 10$*$10, the interval of marker 116 and there are 2$*$2 markers cover on intensities grid.

  Then the image 0_0 usually related to (2,2), 1_0 related to (2,118).

Algorithm workflow:

Stitch image by every images' marker position.

3 Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

4 File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

5 Class Documentation

5.1 cpt::improc::Gridding< FLOAT > Struct Template Reference

Recognize the grid border of the image.

```
#include <gridding.hpp>
```

Public Member Functions

- auto operator() (cv::Mat &in_src, double max_intvl, int16_t v_final, const std::string &img_path, bool verbose=true)

  Recognize the grid border of the image.

5.1.1 Detailed Description

template<class FLOAT>struct cpt::improc::Gridding< FLOAT >

Recognize the grid border of the image.

Detail information can see here Image Gridding

5.1.2 Member Function Documentation

5.1.2.1 template< class FLOAT > auto cpt::improc::Gridding< FLOAT >::operator() (cv::Mat & *in_src*, double *max_intvl*, int16_t *v_final*, const std::string & *img_path*, bool *verbose* = `true`) `[inline]`

Recognize the grid border of the image.

Parameters

in_src	Input image
max_intvl	Max interval of grid line
v_final	Show grid result
img_path	The filesystem path of raw image.
verbose	Set to false if no image shown are needed (will override other "v_" prefix variable), else set to true.

Returns

grid rows, grid cols, grid tiles (a rectangle set)

The documentation for this struct was generated from the following file:

- include/CPT/improc/gridding.hpp

5.2 cpt::improc::MinCVAutoMargin Struct Reference

Adjustment the pixel padding and margin size of grid by searching the minimum coefficient of variation section.

`#include <min_cv_auto_margin.hpp>`

Public Member Functions

- auto operator() (const cv::Mat &src, std::vector< cv::Rect > &tiles, int32_t windows_width, int32_t windows-_height)

  Adjustment the pixel padding and margin size of grid by searching the minimum coefficient of variation section.

5.2.1 Detailed Description

Adjustment the pixel padding and margin size of grid by searching the minimum coefficient of variation section.

Search the minimum coefficient of variation section in the grid cell, use the selected section to represent the cell value. The section size is defined by input parameter. See Image Min CV Segmentation for more detail.

5.2.2 Member Function Documentation

5.2.2.1 auto cpt::improc::MinCVAutoMargin::operator() (const cv::Mat & *src*, std::vector< cv::Rect > & *tiles*, int32_t *windows_width*, int32_t *windows_height*) [inline]

Adjustment the pixel padding and margin size of grid by searching the minimum coefficient of variation section.

Parameters

src	Image data.
tiles	The grid cell generate by gridding step.
windows_width	The result section width in grid cell after auto margin.
windows_height	The result section height in grid cell after auto margin.

The documentation for this struct was generated from the following file:

- include/CPT/improc/min_cv_auto_margin.hpp

5.3 cpt::improc::background_fix::PartialProbeGridSubAndDivisionBase01 Struct Reference

`#include <sub_and_division01.hpp>`

Inherits cpt::improc::background_fix::SegmentMean.

Public Member Functions

- auto operator() (cv::Mat_< float > &grid, cv::Mat &image, std::size_t marker_width, std::size_t marker_-height, std::size_t marker_x_interval, std::size_t marker_y_interval, std::size_t segment_x_num, std::size_t segment_y_num, float background_trimmed_percent) const

  Local background process of image.

5.3.1 Detailed Description

The detail information can see here Image Background Fix. See Image Background Fix for detail.

5.3.2 Member Function Documentation

5.3.2.1 auto cpt::improc::background_fix::PartialProbeGridSubAndDivisionBase01::operator() (cv::Mat_$<$ float $>$ & *grid*, cv::Mat & *image*, std::size_t *marker_width*, std::size_t *marker_height*, std::size_t *marker_x_interval*, std::size_t *marker_y_interval*, std::size_t *segment_x_num*, std::size_t *segment_y_num*, float *background_trimmed_percent*) const [inline]

Local background process of image.

Parameters

grid	The probe grid after ROI (probe domain image)
image	The raw image after ROI (pixel domain image)
marker_width	The width of marker
marker_height	The height of marker
marker_x_-interval	The interval between markers in x direction and is in probe domain
marker_y_-interval	The interval between markers in y direction and is in probe domain
segment_x_num	The x direction number of local segmentation
segment_y_num	The y direction number of local segmentation
background_-trimmed_percent	The percentages of the probes in a single segments, which use to compute the local background

The documentation for this struct was generated from the following file:

- include/CPT/improc/background_fix/sub_and_division01.hpp

5.4 cpt::improc::ROIDetection Struct Reference

The Regioin of Interest detection algorithm implementation.

`#include <r_o_i_detection.hpp>`

Public Member Functions

- auto operator() (cv::Mat_< float > &mean, cv::Mat_< float > &stddev, cv::Mat_< uint16_t > &pixels, cv::Mat_< float > &cv_mat, std::vector< cv::Mat_< int >> &detail_raw_values, bool &roi_qc_fail, const uint32_t &x_marker_num, const uint32_t &y_marker_num, const uint32_t &marker_x_interval, const uint32_t &marker_y_interval, const std::vector< cv::Mat_< uint8_t >> &markers, const std::string &img_path, const bool &enable, const int16_t &v_mean_trimmed, const int16_t &v_mean_binarized, const int16_t &v_mean_score, const int16_t &v_layout_score, const int16_t &v_marker_check, const int16_t &v_mean, const int16_t &v_std, const int16_t &v_cv, const bool &verbose=true)

  The Regioin of Interest detection algorithm implementation.

5.4.1 Detailed Description

The Regioin of Interest detection algorithm implementation.

The detail information can see here Image Region of Interest Detection

5.4.2 Member Function Documentation

5.4.2.1 auto cpt::improc::ROIDetection::operator() (cv::Mat_< float > & *mean*, cv::Mat_< float > & *stddev*, cv::Mat_< uint16_t > & *pixels*, cv::Mat_< float > & *cv_mat*, std::vector< cv::Mat_< int >> & *detail_raw_values*, bool & *roi_qc_fail*, const uint32_t & *x_marker_num*, const uint32_t & *y_marker_num*, const uint32_t & *marker_x_interval*, const uint32_t & *marker_y_interval*, const std::vector< cv::Mat_< uint8_t >> & *markers*, const std::string & *img_path*, const bool & *enable*, const int16_t & *v_mean_trimmed*, const int16_t & *v_mean_binarized*, const int16_t & *v_mean_score*, const int16_t & *v_layout_score*, const int16_t & *v_marker_check*, const int16_t & *v_mean*, const int16_t & *v_std*, const int16_t & *v_cv*, const bool & *verbose* = true) [inline]

The Regioin of Interest detection algorithm implementation.

Search the marker position by the given marker pattern "markers", bound the region coverred by all markers, and extract the intensities.

Parameters

mean	The mean value of intensities grid.
stddev	The standard deviation of the intensities grid.
pixels	The pixel of raw image
cv_mat	The matrix of cv relate to every probe. (can be empty matrix)
detail_raw_-values	The detail pixel values in every probe. (can be empty vector)
roi_qc_fail	The output QC state of ROI process.
x_marker_num	The horizontal direction of marker numbers of the image (mean intensity grid).
y_marker_num	The vertical direction of marker numbers of the image (mean intensity grid).
marker_x_-interval	The grid distance (cell numbers) between markers of the horizontal direction.
marker_y_-interval	The grid distance (cell numbers) between markers of the vertical direction.
markers	The candidate marker patterns. The algorithm will search the marker pattern from the first, and if the match quality is too low, then it will try to match the next candidate of the markers. The marder pattern is specified by opencv matrix with uint8_t integer element, for example : . . . . X X . . X X . . . . X X . . X X . . . . . . X X . . . . X X . . . . . . X X . . X X . . . . X X . . X X . . . . The 'X' is 255. The '.' is 0.
img_path	The filesystem path of raw image.
enable	False then this function will do nothing, otherwise work normally
v_mean_-trimmed	Show trimmed mean image
v_mean_-binarized	Show binarized mean image
v_mean_score	Show scored mean image
v_layout_score	Show layout score image
v_marker_check	Show marker check image
v_mean	Show mean image
v_std	Show standard deviation image
v_cv	Show cv image
verbose	Set to false if no image show process are need (will override other "v_" prefix variable), else set to true.

The documentation for this struct was generated from the following file:

- include/CPT/improc/r_o_i_detection.hpp

5.5 cpt::improc::RotationCalibration Struct Reference

Rotate the image by given angle.

```
#include <rotation_calibration.hpp>
```

Public Member Functions

- template<class FLOAT >
  auto operator() (cv::Mat &in_src, FLOAT theta, int16_t v_final, bool verbose=true)

  Rotate the image by given angle.

5.5.1 Detailed Description

Rotate the image by given angle.

Input the angle and image, the function rotate the image in-place.

Detail information can see here Image Rotation

5.5.2 Member Function Documentation

5.5.2.1 template<class FLOAT > auto cpt::improc::RotationCalibration::operator() (cv::Mat & *in_src*, FLOAT *theta*, int16_t *v_final*, bool *verbose* = `true`) `[inline]`

Rotate the image by given angle.

Parameters

in_src	The input image.
theta	The input rotate angle.
v_final	Show the rotate result.
verbose	Set to false if no image show process are need (will override other "v_" prefix variable), else set to true.

The documentation for this struct was generated from the following file:

- include/CPT/improc/rotation_calibration.hpp

5.6 cpt::improc::RotationEstimation< FLOAT > Class Template Reference

Estimate the rotation angle of image need to be corrected.

`#include <rotation_estimation.hpp>`

Public Member Functions

- auto operator() (cv::Mat &in_src, const bool &has_grid_img, cv::Mat &grid_img, const FLOAT &min_theta, const FLOAT &max_theta, const FLOAT &steps, const int16_t &v_edges, const int16_t &v_hough, bool verbose=true)

  Estimate the rotation angle of image need to be corrected.

5.6.1 Detailed Description

template<class FLOAT>class cpt::improc::RotationEstimation< FLOAT >

Estimate the rotation angle of image need to be corrected.

Template Parameters

FLOAT	The float point type used

The detail information can see here Image Rotation

5.6.2 Member Function Documentation

5.6.2.1 template< class FLOAT > auto cpt::improc::RotationEstimation< FLOAT >::operator() (cv::Mat & *in_src*, const bool & *has_grid_img*, cv::Mat & *grid_img*, const FLOAT & *min_theta*, const FLOAT & *max_theta*, const FLOAT & *steps*, const int16_t & *v_edges*, const int16_t & *v_hough*, bool *verbose* = `true`) `[inline]`

Estimate the rotation angle of image need to be corrected.

Estimate the rotation angle of image

Parameters

in_src	Input image
has_grid_img	Flag for grid image provided or not
grid_img	Grid image. Pass a empty matrix, if not provided.
min_theta	The scan angles' lower bound. Given a angle range to be scan, as small as fast
max_theta	The scan angles' upper bound. Given a angle range to be scanned, as small as faster.
steps	The scan angles' step interval as big as faster, but accuracy will lower.
v_edges	Show the edge image.
v_hough	Show the histogram
verbose	Set to false if no image show process are need (will override other "v_" prefix variable), else set to true.

The documentation for this class was generated from the following file:

- include/CPT/improc/rotation_estimation.hpp

5.7 cpt::improc::Segmentation Struct Reference

Adjustment the pixel padding and margin size of grid.

```
#include <segmentation.hpp>
```

Public Member Functions

- auto operator() (const cv::Mat &src, const std::vector< int32_t > &cell_margin, std::vector< cv::Rect > &tiles, const int16_t &v_final, const bool &verbose=true)

    Adjustment the pixel padding and margin size of grid.

5.7.1 Detailed Description

Adjustment the pixel padding and margin size of grid.

The detail information can see here Image Segmentation

5.7.2 Member Function Documentation

5.7.2.1 auto cpt::improc::Segmentation::operator() (const cv::Mat & *src*, const std::vector< int32_t > & *cell_margin*, std::vector< cv::Rect > & *tiles*, const int16_t & *v_final*, const bool & *verbose* = `true`) `[inline]`

Adjustment the pixel padding and margin size of grid.

Parameters

src	The input image
cell_margin	The cell margin information
tiles	The grid cells
v_final	Show the segment result
verbose	Set to false if no image show process are need (will override other "v_" prefix variable), else set to true.

The documentation for this struct was generated from the following file:

- include/CPT/improc/segmentation.hpp

6 File Documentation

6.1 include/CPT/application/affy2hdf5/sharelib.h File Reference

Define Affymetrix chip sample file (.cel) to Centrillion chip sample file (.cen) convert function.

```
#include <string>
```

Functions

- void **cpt::application::affy2hdf5::file_convert_to** (const std::string &affycel, const std::string &cenhdf5)

    Affy CEL file to CEN hdf5 conversion procedure.

6.1.1 Detailed Description

Define Affymetrix chip sample file (.cel) to Centrillion chip sample file (.cen) convert function.

Author

Chia-Hua Chang

6.2 include/CPT/application/cadtool/sharelib.h File Reference

Define the conversion method between binary CAD file and human-readable JSON text format.

```
#include <string>
```

Functions

- void **cpt::application::cadtool::json2cad_file_convert_to** (const std::string &input_file_path, const std::string &output_file_path)

    Do the file conversion from JSON to CAD.
- void **cpt::application::cadtool::cad2json_file_convert_to** (const std::string &input_file_path, const std::string &output_file_path)

    Define the format conversion method from CAD to JSON.
- void **cpt::application::cadtool::file_convert_to** (const std::string &input_file_path, const std::string &output_file_path, const std::string &mode)

    Do the file conversion from JSON to CAD.

6.2.1 Detailed Description

Define the conversion method between binary CAD file and human-readable JSON text format.

Author

Chia-Hua Chang

6.3 include/CPT/application/cdf2cad/sharelib.h File Reference

Define conversion function of CDF (Affymetrix Chip Description File) to CAD format file.

```
#include <string>
```

Functions

- void **cpt::application::cdf2cad::file_convert_to** (const std::string &input, const std::string &output, const std::string &probe_tab, const std::string &annot_csv)

  Conversion CDF format to CAD format.

6.3.1 Detailed Description

Define conversion function of CDF (Affymetrix Chip Description File) to CAD format file.

Author

Chia-Hua Chang

6.4 include/CPT/application/cenfile_builder/sharelib.h File Reference

Define the JSON schema to CEN HDF5 file building function.

```
#include <string>
#include <CPT/application/cenfile_builder/sharelib.h>
```

Functions

- void **cpt::application::cenfile_builder::build** (const std::string &schema_json, const std::string &result_-cenfile_hdf5)

  Build the CEN HDF5 file from the JSON schema file.

6.4.1 Detailed Description

Define the JSON schema to CEN HDF5 file building function.

Author

Chia-Hua Chang

6.5 include/CPT/improc/background_fix/sub_and_division01.hpp File Reference

The background fix algorithm (number 1). First, compute the local background and subtracted. Second, compute the global background and division.

```
#include <CPT/improc/util.hpp>
#include <CPT/improc/background_fix/index.hpp>
#include <CPT/improc/filters/marker.hpp>
#include <CPT/view/tile.hpp>
#include <CPT/improc/filters/helper.hpp>
#include <CPT/utility/logger.hpp>
```

Classes

- struct cpt::improc::background_fix::PartialProbeGridSubAndDivisionBase01

6.5.1 Detailed Description

The background fix algorithm (number 1). First, compute the local background and subtracted. Second, compute the global background and division.

Author

Chia-Hua Chang

6.6 include/CPT/improc/gridding.hpp File Reference

```
#include <opencv2/core/core.hpp>
#include <CPT/improc/util.hpp>
```

Classes

- struct cpt::improc::Gridding< FLOAT >

  Recognize the grid border of the image.

6.7 include/CPT/improc/min_cv_auto_margin.hpp File Reference

Adjustment the pixel padding and margin size of grid by searching the minimum coefficient of variation section.

```
#include <cstdint>
#include <vector>
#include <CPT/improc/util.hpp>
#include <CPT/utility/assert.hpp>
#include <cassert>
```

Classes

- struct cpt::improc::MinCVAutoMargin

  Adjustment the pixel padding and margin size of grid by searching the minimum coefficient of variation section.

6.7.1 Detailed Description

Adjustment the pixel padding and margin size of grid by searching the minimum coefficient of variation section.

Author

Chia-Hua Chang

6.8 include/CPT/improc/r_o_i_detection.hpp File Reference

The Regioin of Interest detection algorithm implementation.

```
#include <CPT/improc/util.hpp>
#include <CCD/stream_var.hpp>
#include <CPT/improc/chip_mark_layout.hpp>
#include <CPT/improc/coordinate_system_normalization.hpp>
#include <CPT/view.hpp>
#include <CPT/improc/r_o_i_qc.hpp>
```

Classes

- struct cpt::improc::ROIDetection

  The Regioin of Interest detection algorithm implementation.

6.8.1 Detailed Description

The Regioin of Interest detection algorithm implementation.

Author

Alex Lee
Chia-Hua Chang

This module do following things :

1. Extract the exact intensities region of the intensities grid.
2. Correct the coordinate system of intensities grid.
3. Write back to the intensities grid.

6.9 include/CPT/improc/rotation_calibration.hpp File Reference

Rotate the image by given angle.

```
#include <cmath>
#include <random>
#include <CPT/improc/util.hpp>
```

Classes

- struct cpt::improc::RotationCalibration

  Rotate the image by given angle.

6.9.1 Detailed Description

Rotate the image by given angle.

Author

Alex Lee

6.10 include/CPT/improc/rotation_estimation.hpp File Reference

Estimate the rotation angle of image need to be corrected.

```
#include <cmath>
#include <random>
#include <CPT/improc/util.hpp>
#include <CCD/para_thread_pool/para_thread_pool.hpp>
#include <CPT/improc/hough_transform.hpp>
#include <atomic>
```

Classes

- class cpt::improc::RotationEstimation< FLOAT >

    Estimate the rotation angle of image need to be corrected.

6.10.1 Detailed Description

Estimate the rotation angle of image need to be corrected.

Author

Alex Lee, Chia-Hua Chang

6.11 include/CPT/improc/segmentation.hpp File Reference

Adjustment the pixel padding and margin size of grid.

```
#include <CPT/improc/util.hpp>
```

Classes

- struct cpt::improc::Segmentation

    Adjustment the pixel padding and margin size of grid.

6.11.1 Detailed Description

Adjustment the pixel padding and margin size of grid.

Author

Alex Lee

Index