
Conime Programmer's Manual

Rephael Wenger

Table of Contents

| | |
|---|---|
| 1. Introduction | 1 |
| 1.1. History | 1 |
| 1.2. System Requirements | 1 |
| 1.3. Hardware Requirements | 2 |
| 1.4. Documentation | 2 |
| 1.5. Open Source License | 2 |
| 1.6. Authors | 2 |
| 2. Software Organization | 3 |
| 2.1. Graphical User Interface | 3 |
| 2.2. Input and Output Routines | 3 |
| 2.3. Data Structures | 3 |
| 2.4. Image Processing Algorithms | 3 |
| 3. Image Processing Algorithms | 4 |
| 3.1. Background Filtering | 4 |
| 3.2. Contour Segmentation | 5 |
| 3.3. Template Segmentation | 6 |
| 3.4. Registration | 6 |
| 3.5. Spot Identification | 8 |
| 3.6. Spot Intensity Normalization | 8 |
| 3.7. Spot Comparison | 9 |
| References | 9 |

1. Introduction

Conime is a C++ program to view, process and analyze two dimensional gel electrophoresis images of DNA sequences, also known as Restriction Landmark Genomic Scanning (RLGS) profiles. This document describes the structure of the conime software and the important individual modules in that software.

1.1. History

Restriction landmark genomic scanning (RLGS) is a two dimensional gel electrophoresis technique used in the laboratory of Dr. Christoff Plass (OSU Human Cancer Genetics) for detecting cancer related changes in DNA. A major cost of RLGS is the time and manpower needed to analyze the images of RLGS gels. In Spring, 1999, Dr. Rephael Wenger (OSU Computer Science and Engineering) and Dr. Plass started a collaborative effort to automate some of that analysis. Seed money for conime was provided by a grant from the OSU Cancer Center. From 2001-2004, conime was funded by a grant from the National Cancer Center.

1.2. System Requirements

Conime is written in C++ and uses X11 and OpenGL for its graphics display and Motif for its user interface. It also uses the libtiff library for reading tiff files. It has been compiled and tested on both Linux and Solaris operating systems.

1.3. Hardware Requirements

The CPU and memory requirements for conime depend on the size of the images under analysis. A 17x14 inch autoradiograph scanned at 300 DPI has 5100x4200 pixels. If pixel intensities are stored in 8 bits (0-255), the image requires approximately 20 Meg of memory. Conime requires about 250 Meg of memory for each 20 Meg image processed.

Conime requires approximately 2.5 minutes of CPU time to filter, segment, register and compare a single 20 Meg image with a master image on a 3 Ghz Intel processor running Linux. Running times will vary depending on the quality of the image and its similarity to the master image. Running time also depends upon user specified settings of the various algorithm parameters. The times given here are for the default parameter settings.

1.4. Documentation

Conime currently has the following documentation:

- Conime man page;
- Conime Programmer's Manual (this document);
- Conime source code documentation (generated from source code comments by doxygen);

1.5. Open Source License

The Ohio State University has granted the following open source license for conime:

COPYRIGHT (c) 2005, THE OHIO STATE UNIVERSITY ALL RIGHTS RESERVED

Permission is granted to use, copy, create derivative works and redistribute this software, associated documentation and such derivative works (the "SOFTWARE") in source and object code form provided that the above copyright notice, this grant of permission, and the disclaimer below appear in all copies and derivatives made; and provided that the SOFTWARE and any derivative works are made available in source code form upon request; and provided that The Ohio State University and authors of the SOFTWARE are acknowledged in any publications reporting its use, and the name of The Ohio State University or any of its officers, employees, students or board members is not used in any advertising or publicity pertaining to the use or distribution of the SOFTWARE without specific, written prior authorization.

THE SOFTWARE IS PROVIDED AS IS, WITHOUT REPRESENTATION FROM THE OHIO STATE UNIVERSITY AS TO ITS FITNESS FOR ANY PURPOSE, AND WITHOUT WARRANTY BY THE OHIO STATE UNIVERSITY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT. The Ohio State University has no obligation to provide maintenance, support, updates, enhancements, or other modifications. The Ohio State University shall not be liable for compensatory or non-compensatory damages, including but not limited to special, indirect, incidental, or consequential damages, with respect to any claim arising out of or in connection with the use of the SOFTWARE, even if it has been or is hereafter advised of the possibility of such damages.

1.6. Authors

The following people worked on the software development of conime: Tahmina Ansari, Jason Knight, Ramakrishnan Kazhiyur-Mannar, Arun Somasundaram, Dr. Rephael Wenger.

The following people provided biomedical expertise, images and testing for the development of conime: Mau-Ting Lin, Dr. Christoff Plass, Dr. Dominic Smiraglia.

2. Software Organization

The software in conime is divided into four independent parts: the user interface, input/output routines, data structures and image processing algorithms. The four parts are independent so that changes and improvements can be made independently to all four. In particular, the image processing algorithms are independent of the image, spot and marker data structures.

2.1. Graphical User Interface

Conime's graphical user interface is based on OpenGL, a widespread, multiplatform graphics standard, and on X11/Motif, an X-windows user interface toolkit. X-windows is the widely used windowing system for Unix/Linux operating systems. Open source versions of both OpenGL and Motif are available.

The user interface is broken into three parts: viewports for viewing the RLGS images before and after the application of image processing algorithms, menus and dialogs for applying the image processing algorithms and modifying the view of the images, and a set of X11 callback routines connecting the Motif menus and dialogs to specific actions. OpenGL is used to draw the viewports, displaying the RLGS images and drawing graphical objects such as spot centers and markers. Motif is used to create the menus and dialogs. The callback routines are processed by the X11 windowing system.

2.2. Input and Output Routines

Conime's input and output routines read the RLGS images from files, read and write spot and marker information in conime (.cnm) files, and read and write conime program settings in conime initialization files. Images can only be read from tiff files. Numerous programs, both free and commercial, are available to convert from other image formats to tiff format. Conime uses the open source library libtiff to read tiff files.

Conime (.cnm) files are an internal format designed for conime, which stores spot and marker information for a given image. Each conime file contains information about a single image. An image can have multiple conime (.cnm) files, allowing different users to perform independent analyses of the image or a user to perform different types of analyses of the images.

Conime initialization files control numerous conime program settings such as filtering, registration and matching parameters, initial window size and initial image scale.

2.3. Data Structures

Conime has three basic data objects, images, spots and markers. Images are the original RLGS gel image, and warped, filtered and compressed versions of the image. Conime also stores for each pixel an identifier telling whether and which spot contains the pixel.

For each spot, conime stores a center location, a bounding box and minimum and maximum gray scale values. Conime also stores an optional spot label and flags telling whether the spot has been identified from a Master image and whether it is hidden (not visible.)

For each marker, conime stores a marker shape (square, cross, X,) color, size and a required label. Markers also have flags telling whether the markers should be used for image registration, and whether they are hidden.

Laboratories using RLGS divide Master images with an 8x7 grid, to aid in identifying and locating spots. Conime has grid data structures which can represent such grid overlays. Conime also uses grid data structures to control image warping.

2.4. Image Processing Algorithms

Conime has the following algorithms:

- Background filtering: Identifies foreground and background pixels based on areas within contour lines;
- Contour segmentation: Segments foreground into individual spots based on areas within contour lines;
- Template segmentation: Resegments foreground using a segmented template image as a guide;
- Image comparison: Compares subregions of two images, measuring the partial Hausdorff distance between foreground pixels in the subregions;
- Image registration: Translates and warps one image to register it with another, using image comparison to identify matching regions in the image;
- Spot identification: Identify and label spots on one image from the identified, labelled spots on a master image;
- Spot intensity normalization: Normalize the spot intensities of an image to a "uniform" scale;
- Spot comparison: Compare spots in two images and report spot differences, missing or added spots or amplified spots.

3. Image Processing Algorithms

3.1. Background Filtering

A primary concern of conime is to avoid missing any spots in the initial image processing steps. Conime divides spot extraction into two parts: background filtering and segmentation. For background filtering, we identify isocontours which enclose regions of a specified area and select those regions as foreground. We apply standard pixel image processing techniques such as erosion and dilation to remove noise from the images. For segmentation, we again use isocontours, this time to identify spot centers. A spot is "grown" from each center.

By using isocontours instead of thresholds, our filtering algorithm is much more sensitive to the slightest variation in intensities. As importantly, good filtering parameters depend upon the spot size and density, not upon the spot intensities. Since spot size and density are consistent between RLGS profiles, the parameters are not modified for each gel. Good sensitivity values in other gel analysis software depend greatly on spot intensity which make them much more gel dependent.

The contour area filtering algorithm takes three parameters: `max_fg`, `min_fg` and `open_size`. Parameters `max_fg` and `min_fg` are the maximum and minimum sizes of foreground connected components, respectively. Decreasing `max_fg` decreases the number of foreground pixels. Decreasing `min_fg` increases the number of foreground pixels and the number of small components. Default value for `max_fg` is 60,000. Default value for `min_fg` is 100. Parameter `open_size` is the pixel size used for the opening operator[Gonzalez92]. Default value for `open_size` is 2 pixels.

Procedure 1. Contour Area Filtering

1. Sort all pixels by decreasing intensity using counting sort [Cormen01];
2. For each pixel p , create a set $\{p\}$ containing the pixel p ;
3. Process pixels in sorted order by decreasing intensity, forming the union of the set $\{p\}$ and the sets containing the adjacent pixels (above, below, left or right) which have intensity greater than or equal to $\{p\}$;
4. After pixel p is processed, store the size $A(p)$ of the current set containing p . Size $A(p)$ is the area of the largest connected component containing p of pixels with intensities greater than p ;

5. Let U be the set of pixels with $A(p)$ less than `max_fg` (default 60,000);
6. Remove some pixels from U by applying the opening operator with size `open_size` (default 2) to the pixels in U (See [Gonzalez92].) Let U' be the remaining set of pixels;
7. Form the connected components of U' ;
8. Remove from U' any pixels in connected components with size less than `min_fg` (default 100);
9. The remaining pixels in U' are the foreground pixels;

Our experimental results on conime's background filtering including a comparison with ImageMaster software are reported in [Kazhiyur05]. Conime found 97-98% of the spots on the human NotI-EcoRV-HinfI and AscI-EcoRV-HinfI master gels. Misses occurred in regions of high spot density or where a faint spot is adjacent to and shadowed by a darker spot. Such regions are problematic for other spot extraction algorithm.

Use of contours and contour properties is described in [Bajaj97][Carr04]. The union-find algorithm used for constructing contours is also similar to the one given by Carr et. al. in [Carr00]. The algorithm is also similar to watershed segmentation.

3.2. Contour Segmentation

The contour area segmentation algorithm takes two parameters: `min_center_size` and `min_center_depth`. Parameter `min_center_size` is the minimum size of connected component of center pixels. The intensity depth of a connected component of center pixels is the difference between the maximum and minimum intensities of the pixels in that component. Parameter `min_center_depth` is the minimum intensity depth of a connected component.

Procedure 2. Contour Area Segmentation

1. Sort all foreground pixels by decreasing intensity using counting sort[Cormen01];
2. For each foreground pixel p , create a set $\{p\}$ containing the pixel p ;
3. Process foreground pixels in sorted order by decreasing intensity, forming the union of the set $\{p\}$ and the sets containing the adjacent foreground pixels (above, below, left or right) which have intensity greater than or equal to $\{p\}$;
4. After pixel p is processed, store the size $A(p)$ of the current set containing p . Store the maximum intensity $M(p)$ of any pixel in the current set containing p . Size $A(p)$ is the area of the largest connected component containing p of pixels with intensities greater than p . Intensity $M(p)$ is the maximum intensity of any pixel in the largest connected component containing p of pixels with intensities greater than p ;
5. Let U be the set of pixels with $A(p)$ less than `max_center_size` or with $M(p)$ minus the intensity of p less than `max_center_depth`;
6. Form the connected components of U ;
7. Create a spot center at the center of each connected component of U ;
8. Simultaneously grow each spot from its center, partitioning the pixels in U into different spots;
9. Store each spot center, the spot's minimum and maximum intensity and a bounding box around the pixels in the spot;

Both contour area filtering and contour area segmentation run in time which is approximately linear in the total number of pixels. Theoretically, the algorithm could take $O(n \log(n))$ time, but would only do so on highly unusual input data. Counting sort takes linear time.

3.3. Template Segmentation

After foreground pixels are segmented into spots, additional segmentation can be performed by matching the image with a template image and using the template image to guide further addition or splitting of spots. Spots can also be identified and labelled by matching them with identified and labelled spots from the template image.

After the initial matching and template segmentation, the segmentation image can be reregistered with the template image based on the spot matches and further segmentation can be attempted. This matching, segmentation and reregistration can be iterated a number of times.

The segmentation and template images must be registered before applying template segmentation. Thus an unfiltered image must first be filtered and initially segmented, then registered with the template and only then can it be segmented with the template.

The segment dialog permits the choice of contour segmentation or template segmentation or both. Contour segmentation reapplies segmentation based on contour area and has the parameters as described above. Template segmentation requires a filtered, segmented template image from which to apply segmentation. Template segmentation parameters are number of iterations, identify spots flag, and tag spots flag. Number of iterations is the number of iterations of the matching, segmentation and reregistration procedure. If the identify spots flag is true, then spots on the segmentation image are identified and labelled using identified and labelled spots on the template image. If tag spots flag is true, then spots on the segmentation image have tags added which indicate whether the spot was added or split and how the spot was matched with a spot in the template image.

3.4. Registration

Conime registers images by selecting a scattered set of rectangular regions in one image and matching these regions against the comparison image. Conime matches images using the pixel matching algorithm described in [7]. The algorithm attempts to minimize the partial Hausdorff distance between the foreground pixels of the two images. It measures the distance between each foreground pixel and the closest foreground pixel in the other image, throws away a certain percentage of the largest distances, and returns the maximum of the remaining values. Because the algorithm throws away the largest distances, noise and spot differences do not hinder it from finding good matches.

The matching in conime depends only on the foreground pixels identified in the background filtering algorithm, not upon the intensities of the pixels or the spot segmentation. It is thus unaffected by differences in overall image intensities or by segmentation errors or differences between images.

Conime places corresponding registration markers at the center of corresponding matched regions. Corresponding registration markers can be placed in multiple images, allowing multiple image registration.

The registration algorithm takes six parameters: `cfactor`, `match_window_width`, `max_num_rmarkers`, `ignore_percent`, `matching_threshold`, `max_angular_distortion`. Parameter `cfactor` is the compression factor. Each image is compressed by `cfactor` x `cfactor` before searching for matching regions. Parameter `match_window_width` is the size of the window width and height used for matching. Parameter `max_num_rmarkers` is the maximum number of registration markers which will be added to the images. Parameter `ignore_percent` is the percentage of pixel distances which will be ignored in computing the partial Hausdorff distance. Parameter `matching_threshold` is a maximum threshold for considering two image windows as forming a match. If the ratio of the partial Hausdorff distance for the two windows over the partial Hausdorff distance for a random set of pixels is less than the `matching_threshold`, then the windows match. Parameter `max_angular_distortion` is the maximum angular distortion created by corresponding registration markers. The difference between the coordinates of a pair of registration markers is a vector. If the angle between a vector for one pair of registration markers and the corresponding vector between the corresponding pair of registration markers is greater than `max_angular_distortion`, the one of the pairs of registration markers is removed. The registration algorithm takes one reference image and a set of other images. It registers each of the images with the reference image. The registration algorithm is as follows.

Procedure 3. Image Registration

1. For each image, compute the approximate distance D between registration markers;
2. For each image, compute the distance of each foreground pixel to the boundary of the foreground region;
3. For each image, generate a set of `max_num_rmarkers` potential registration markers as follows:
 - a. While there is still some foreground pixel at least distance D from the other registration markers and fewer than `max_num_rmarkers` have been generated, create a registration marker on the foreground pixel with greatest distance to the boundary of the foreground region;
 - b. If fewer than `max_num_rmarkers` have been generated at the completion of the previous step, repeat using a distance $D/2$ instead of D ;
4. For each image, compute the distance of each pixel to the closest foreground pixel;
5. Compress each image by `cfactor` x `cfactor`;
6. For every registration marker, compute nine equally spaced locations in a window centered at `r1` of width and height `max_match_width`;
7. For every registration marker `r1` in the reference image A and every registration marker `r2` in another image B, do:
 - a. Let `w` equal `match_window_width`;
 - b. Let (x,y) be the `x` and `y` locations of marker `r2`;
 - c. While `w` is greater than 1 do:
 - i. Compute the partial Hausdorff distance between a window centered at `r1` and windows centered at the nine points $(x-w, y-w)$, $(x, y-w)$, $(x+w,y-w)$, $(x-w, y)$, (x, y) , $(x+w, y)$, $(x-w, y+w)$, $(x,y+w)$, $(x+w, y+w)$;
 - ii. Replace (x,y) with the point from the set of nine with minimum partial Hausdorff distance;
 - iii. Set `w` to equal `w/2`;
 - iv. Compute the partial Hausdorff distance between random sets of foreground pixels centered at `r1` and at (x,y) ;
 - v. If the ratio of the partial Hausdorff distance between `r1` and (x,y) and the partial Hausdorff distance between random sets is less than `ignore_percent`, then add matching registration markers at `r1` in image A and (x,y) in image B;

The partial Hausdorff distance between a window centered at `r1` in image A and a window centered at (x,y) in image B is computed as follows.

Procedure 4. Partial Hausdorff Distance

1. Translate image A over image B so that $r1$ and the location (x,y) are at the same location;
2. For each foreground pixel in image A, compute the distance to the closest foreground pixel in image B. Take all the computed distances, delete the largest ignore_percent percentage and let $d(A,B)$ be the maximum of the remaining distances;
3. For each foreground pixel in image B, compute the distance to the closest foreground pixel in image A. Take all the computed distances, delete the largest ignore_percent percentage and let $d(B,A)$ be the maximum of the remaining distance;
4. The partial Hausdorff distance is $\max(d(A,B), d(B,A))$;

After computing a set of matching registration markers, conime deforms one image to align its registration markers with the registration markers in a corresponding image. The deformation is produced by placing a regular 200x200 grid over the image, using the registration markers to identify locations for nearby grid vertices, interpolating the locations of remaining grid vertices, and then using bilinear interpolation to map each grid square to the target image.

Conime uses warped images only to determine corresponding spot center locations and for visualization purposes. Measurements are always performed on the unwarped images.

Pixel based matching is extremely computation intensive and so is done on compressed version of the image, typically with 25:1 compression. Conime has little problem registering even poor quality RLGS images.

3.5. Spot Identification

Laboratories applying RLGS usually select a master gel for each enzyme-genome combination. The master gels are carefully analyzed and each spot on the master gel is assigned a unique label. Spots on other gels are assigned labels by matching them with spots on the master gel.

Conime duplicates this process, assigning labels to spots on a gel by matching them with spots on other gels. For each enzyme-genome combination, the labels must be manually assigned to spots in one or more initial gels. New gels are then filtered, segmented and registered with the initial gels, and then spots are matched based on image location proximity. Labels are copied from spots in the initial gels to matching spots in the new gel.

Matching is based on the closeness between spot centers, the overlap of pixels assigned to each spot, and the partial Hausdorff distance between a window around each spot. Only the closest pairs of spot centers are matched. Closeness between spot centers is measured as the ratio of the distance to the closest spot to the distance to the second closest spot. If the partial Hausdorff distance is below a certain threshold and the overlap of pixels is greater than a threshold and the closeness is below a threshold, then the spots are matched.

The matching algorithm is extremely conservative to avoid misidentification of any gel spots. Typically, half the spots in the new gel are labeled, although the ratio varies with the correspondence between the new gels and the initial gels. Spot labels can be used to register any two gels from the same enzyme-genome combination.

3.6. Spot Intensity Normalization

Spot intensities in RLGS profiles vary greatly, both between two images and across an image. Researchers typically compare spots with nearby spots to adjust for local variations in intensity or intensity variations between gels. Similarly conime computes the median intensity of nearby spots and the median intensity of surrounding background

pixels and modifies spot pixel intensities based on those values. Normalization errors can appear when faint spots are adjacent to clusters of darker ones.

3.7. Spot Comparison

Spots are measured by normalizing their pixel intensities and extracting various measures from the histogram of normalized spot pixel intensities. Spots from a source image are compared with a target image by registering and normalizing the two images and then comparing measures of corresponding pixels. Measures include median and 90th percentile pixel intensities and percentage of spot pixels from one image which are foreground pixels in the other image. If the difference or ratio of normalized pixel intensities exceeds various thresholds or the foreground percentage is too small, then the spot is marked as a mismatch. No attempt is made to match spots or rely upon accurate spot segmentation.

References

- [Bajaj97] C.L. Bajaj, V. Pascucci, and D.R. Schikore. *The Contour Spectrum*. Proceedings of the 8th Conference on Visualization '97. 1997.
- [Carr00] H. Carr, J. Snoeyink, and U. Axen. *Computing Contour Trees in All Dimensions*. Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms. 2000.
- [Carr04] H. Carr, J. Snoeyink, and M. van de Panne. *Simplifying Flexible Isosurfaces Using Local Geometric Measures*. Proceedings of IEEE Visualization '04. 2004.
- [Cormen01] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. 2001. Second Edition. MIT Press.
- [Gonzalez92] R.C. Gonzalez and R.E. Woods. *Digital Image Processing*. 1992. Addison-Wesley Longman Publishing Co., Inc..
- [Kazhiyur05] Ramakrishnan Kazhiyur-Mannar and Dominic J. Smiraglia. *Contour Area Filtering of 2-Dimensional Electrophoresis Images*. 2005. The Ohio State University. OSU CSE Technical Report: OSU-CIS-RC-2/05-TR10 .