
POINCaRe: A User Guide

*Amir H. Meghdadi,
James F. Peters (Supervisor)
University of Manitoba*

Computational Intelligence Laboratory

ENGR E1-526 Engineering & Information Technology Complex

75A Chancellor's Circle

Winnipeg, Manitoba R3T 5V6

Submitted 3 February 2012

UM CI Laboratory Technical Report
Number TR-CI-17-Jan-2012
3 February 2012

University of Manitoba [poincare](http://wren.ee.umanitoba.ca/)

Computational Intelligence Laboratory

URL: <http://wren.ee.umanitoba.ca/>

Copyright © CI Laboratory, University of Manitoba 2012.

Project no. CI-17-Jan-2012

POINCaRe: A User Guide

Amir H. Meghdadi,
James F. Peters (Supervisor)
University of Manitoba
Computational Intelligence Laboratory
ENGR E1-526 Engineering & Information Technology Complex
75A Chancellor's Circle
Winnipeg, Manitoba R3T 5V6
Submitted 3 February 2012

CI Laboratory TR-CI-17-Jan-2012

3 February 2012

This research has been funded by the Natural Sciences & Engineering Research Council of Canada (NSERC) grant 185986, Manitoba Centre for Excellence Fund (MCEF) and the Canadian Arthritis Network (CAN) Bioengineering grant SRI-BIO-05.

Contents

Abstract	3
Keywords	3
1 Introduction	3
2 POINCaRe Download and Install Instructions	4
3 Using the Program	5
3.1 Selecting Parameters	5
3.2 Choosing the features (represented by probe functions) and methods	7
4 Types of Analysis	7
4.1 Pairwise image comparison	7
4.2 Histogram of features	8
4.3 Finding tolerance neighborhoods and manual selection of a neighborhood	8
4.4 Edge detection	8
4.5 Selecting a region of interest (ROI)	9
4.5.1 Upper and lower approximation of ROI by tolerance neighborhoods	9
4.5.2 Nearness between an ROI and a test image	10
4.6 Content-based image retrieval	10
5 Visual Features	11
5.1 List of Probe Functions	11
5.1.1 Average gray level value	11
5.1.2 Information content (entropy)	11
5.1.3 Color features	14
5.1.4 Texture and statistical features	14
5.1.5 Edge features	15
References	15

Abstract

This research report introduces the POINCaRe image analysis system named after Jules Henri Poincaré (1854 - 1912). This system provides a variety of image analysis approaches and instantiates the perception of nearness between objects in a visual field. The current version of POINCaRe uses neighborhoods of visual elements (small subimages) in determining the degree of nearness of pairs of nonempty sets. With this tool, it is possible to engage in content-based image retrieval (CBIR) from a near set perspective considered in the context of tolerance space, metric spaces, fuzzy sets, fuzzy tolerance spaces and fuzzy valued distances.

Keywords: Perception of nearness, image similarity, nearness measure, content-based image retrieval (CBIR), near sets, neighbourhood of a point, query by content, tolerance spaces, metric spaces, fuzzy sets, fuzzy metric spaces, fuzzy tolerance spaces, fuzzy valued distance.

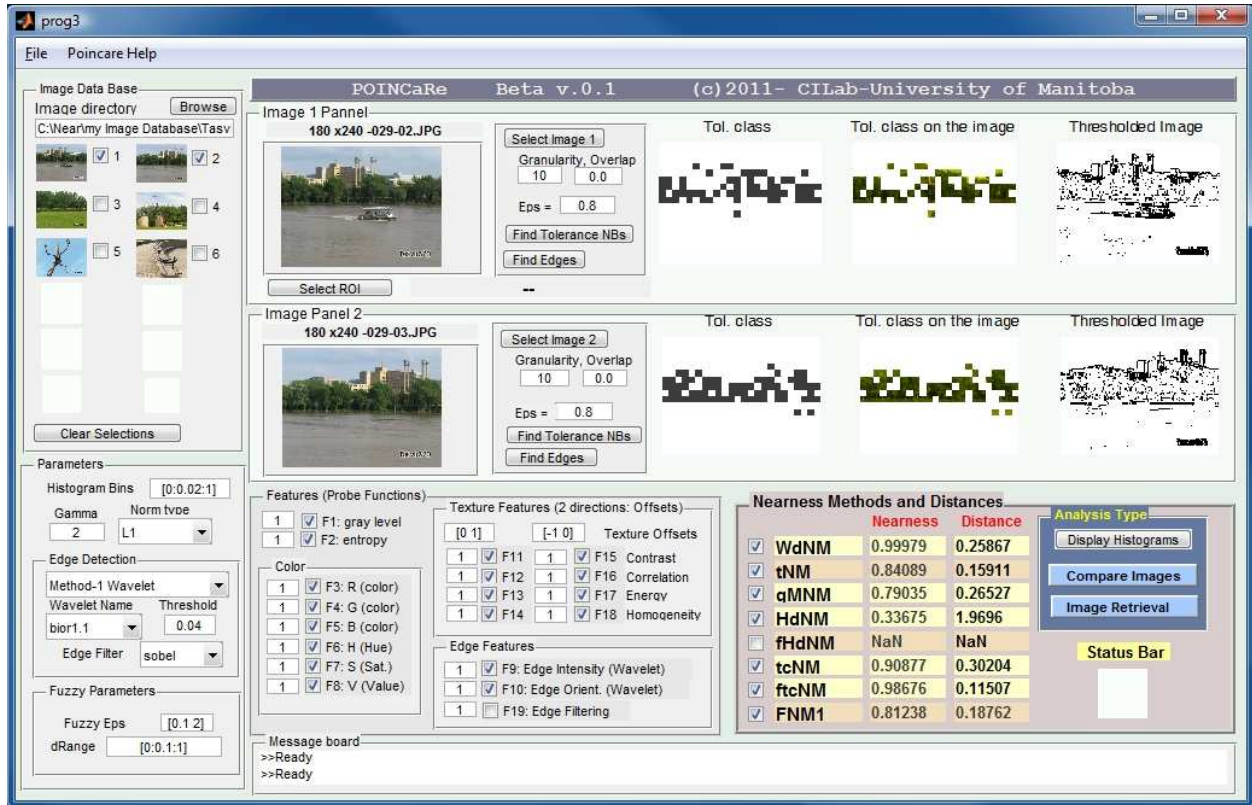


Figure 1: A snapshot of the GUI of POINCaRe beta version 0.1

1 Introduction

POINCaRe (Program for Object and Image Nearness Comparison and Recognition) is a computer application designed for image similarity analysis and content based image retrieval. The program was developed as part of a PhD thesis [8] completed in Computational Intelligence Laboratory at the University of Manitoba. This image analysis system is an outgrowth of several years of intensive research on the image resemblance problem (see, e.g., [11, 9, 14, 10, 13]).

An executable version of POINCaRe can be downloaded from the Computational Intelligence Laboratory web site at the University of Manitoba¹. POINCaRe was originally written in MATLAB but is now available as a standalone executable program. POINCaRe is named after Jules Henri Poincaré (1854 - 1912), whose work on the philosophical aspects of the contrast between the mathematical and physical continua that led to the idea of tolerance space theory. POINCaRe can be also read as the initials for: Program for Object and Image Nearness Comparison and Recognition.

Program Features

The current released version of POINCaRe (beta 0.1) has the following capabilities:

- (**Feature.1**) Calculating 8 different similarity measures between digital images based on their visual features.
- (**Feature.2**) CBIR: Calculating the similarity between a given query image and test images in a selected directory of images as well as sorting the images in an HTML file, based on similarity to the query image.
- (**Feature.3**) Specifying a region of interest (ROI) in a query image and comparing the ROI with the test image(s).
- (**Feature.4**) Individual analysis of images in terms of, for example, edge detection, plotting the histogram of local feature values, and finding tolerance neighborhoods in images.

Figure 1 shows a snapshot of the graphical user interface (GUI) of the program.

2 POINCaRe Download and Install Instructions

POINCaRe is a standalone executable application that has been implemented with MATLAB but does not need MATLAB to run. However, you need to have the proper version of MATLAB Compiler Runtime (MCR) installed on your computer. You can read about MCR from the Mathworks² web site. Follow the following steps to install MCR and POINCaRe. If a recent version of MATLAB is installed on your computer, then you already have MCR installed. You can check the version of your MCR and go directly to step 2 below. If there is an MCR problem, you can always come back to step 1 and install a proper MCR.

Step 1: Download and Install MATLAB Compiler Runtime (MCR):

Skip this step if an updated version of MATLAB or MCR has already been installed on your computer. You need MCR version 7.15 or higher to run version 0.1 (beta) of POINCaRe. If MATLAB is installed on your computer, you can type: `[major, minor] = mcrversion` at your MATLAB command prompt to see what is the version of MCR on your computer. If you don't have MATLAB or an updated version of MCR, you need to install MCR on your computer (only once). `MCRInstaller.exe` will install MCR on your computer. Due to licensing issues, `MCRInstaller.exe` file cannot be uploaded with open access. However, you can obtain this file from any licensed MATLAB distribution that comes with MATLAB compiler. In MATLAB 7.5 (R2007b) and newer, the command `(mcrinstaller)` can be used to determine where the installer is located. You can copy the file into your computer and run it.

Step 2: Download and Install POINCaRe

Currently, there is only a 64 bit version of POINCaRe available for Microsoft windows. Check the web site³ or contact us for updates and new versions. Download the self extracting executable file. Copy the

¹<http://wren.ee.umanitoba.ca/>

²www.mathworks.com/

³<http://wren.ece.umanitoba.ca>

file into a directory in your PC. Extract the file contents into a directory. Double click on the main file Poincare_win64_v1.exe to run the program.

3 Using the Program

To begin using the POINCaRe system, consider following steps. You can browse and select a directory that contains your images using the controls shown in figure ??.

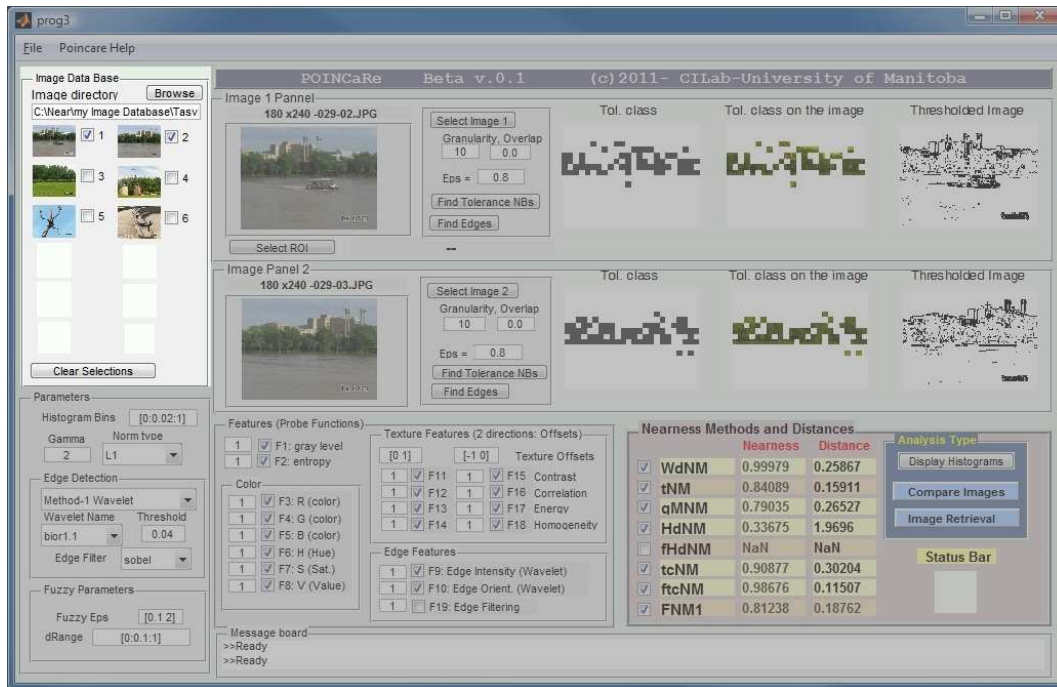


Figure 2: POINCaRe: Image Panels

This directory should contain only images. Most of the common image file formats are supported. The images do not have to be the same size. However, since the granularity parameter (subimage size) will be the same for all images, it is strongly suggested to avoid using images with significantly different sizes so that the ration of the subimage size the image itself has little variations for the sake of consistency. There is a default directory with 30 sample images located in the same path that the program is installed. These images are selected from Tasvir3x70 dataset. There is no limit in the number of images. However, a thumbnail view of the first 12 images in the directory are shown in the Image Data Base panel. Pairs of images can be selected from this panel to be compared.

3.1 Selecting Parameters

All of the methods implemented in POINCaRe are based on dividing images into subimages and calculating the local features at each subimage. **Granularity** (in pixels) is the size of square subimages and **Overlap** is a number between 0 and 1 that represents degree of overlap between subimages. Default value of overlap is zero. Figure 3 shows where to enter the parameters. The current implemented methods require both epsilon values for image 1 and 2 to be the same.

Note that the value of granularity depends on the size of images. As a suggested compromise between accuracy and speed, it is recommended to choose the sub image size such that there are no more than

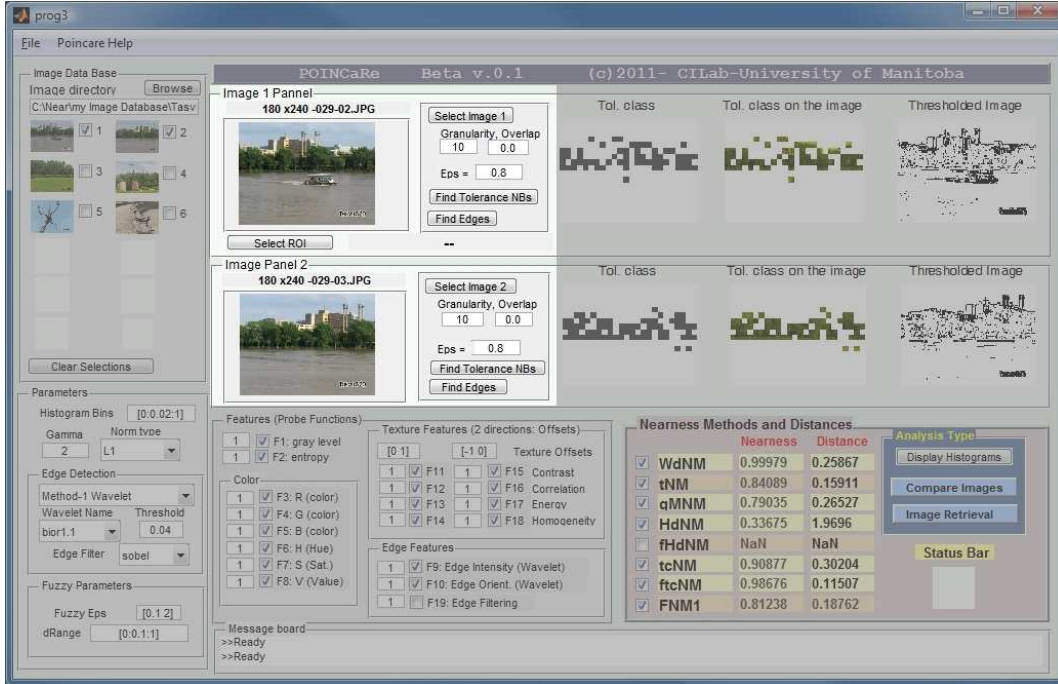


Figure 3: POINCaRe: Pairs of Images and Parameters

approximately 500 subimages in each image. The Epsilon value (ε) for tolerance based methods can be selected as a fixed value. The method for adaptive selection of epsilon value based on the image data has not been implemented in this simplified version of POINCaRe.

Other parameters can also be selected as it can be seen in figure 4 as follows:

- **Histogram Bins:** This is a vector with values between 0 and 1 representing the normalized histogram bins in any method that is based on histogram calculations ($KdNM$ or $WdNM$)
- **Gamma (γ):** This parameters is a scaling parameter in converting distance measure to similarity measure. if D is a distance measure, similarity or nearness measure (NM) is calculated using the mapping $NM = 1 - D^\gamma$ if $D \in [0, 1]$ or $NM = \frac{1}{1+D^\gamma}$ if $D \in [0, +\infty)$.
- **Norm Type:** The norm type is the type of norm used in calculation of the distance between visual elements. Default values is L_1 norm or Manhattan distance.
- **Wavelet Name and Threshold:** Wavelet functions of type **Wavelet Name** will be used for calculating the edge intensity and edge orientations based on the edge detection method in [6]. The threshold value **Threshold** is used to detect an edge if the edge intensity is above the threshold. The parameter **Edge Filter** is the type of filter for another method of edge detection base on the gradient of the image using different operators such as *sobel* or *prewitt* ([2]). This method will be used if the 19th feature (Edge filtering) is selected (see figure 5).
- **Fuzzy Eps and dRange:** The parameter **Fuzzy Eps** is a vector that contains the two corner parameters of the fuzzy tolerance relation ε_1 and ε_2 . **dRange** is a vector of values where the membership function of the fuzzy distance μ_{ftcFDM} is calculated.

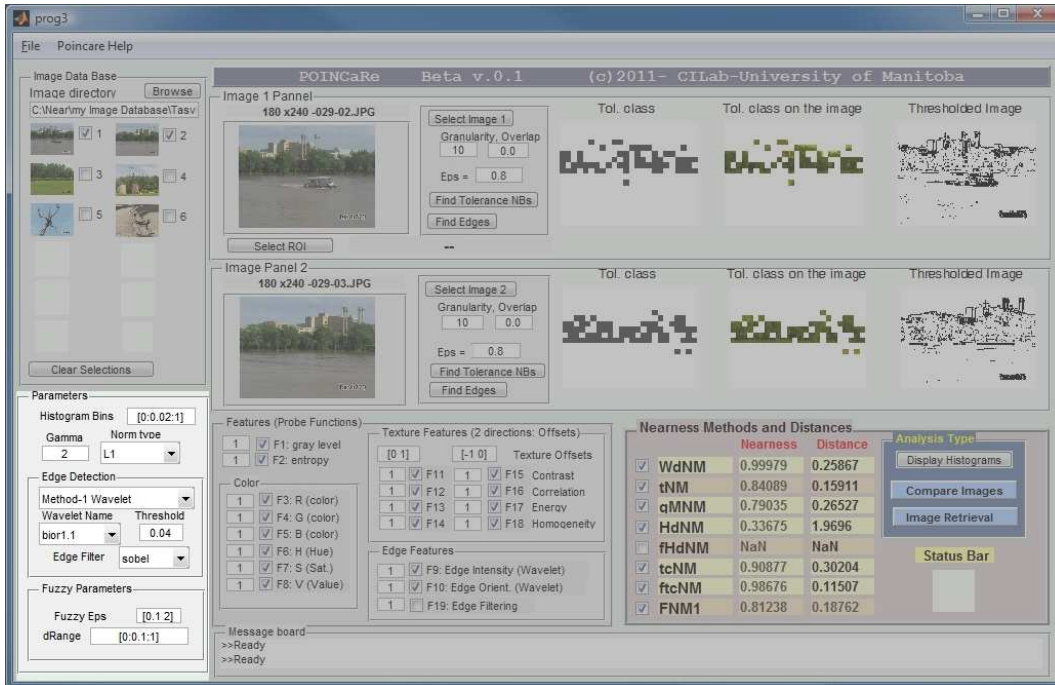


Figure 4: POINCaRe: Parameters Panel

3.2 Choosing the features (represented by probe functions) and methods

The user can select up to 19 different features to be used in the feature vector for each visual element (subimage). Features will be later normalized between 0 and 1. For more information on how each feature is calculated, you can refer to section 5. These features describe average color, texture and edge information in each subimage. Note that POINCaRe ver 0.1 does not have the ability to automatically choose the tolerance threshold (ϵ). Therefore, the user is advised to choose the fixed epsilon value according to the number of selected features in each experiment. Each normalized feature has a range of variation between 0 and 1. Therefore, the range of variation of $d(x, y) = \|\vec{\phi}_B(x) - \vec{\phi}_B(y)\|_1$ (assuming an L_1 norm distance is used) is between 0 and M where M is the number of features.

4 Types of Analysis

There are 3 different type of analysis possible in POINCaRe. The first type is individual analysis on the image content of each single image. The second type is pairwise comparison of a pair of images and calculating the similarity and/or distance between images. The last type of analysis is CBIR image analysis by calculating the similarity/distance between a query image and all the images in a directory and sorting the images based on similarity. The program can perform all the above analysis types as follows:

4.1 Pairwise image comparison

A pair of images loaded into Image Panel 1 and Image Panel 2 can be directly compared to each other. Each image can be selected by using the [Select Image] button in each panel or by choosing the corresponding thumbnail image from the [Image Data Base] panel. Similarity between images is calculated using the given parameters in the GUI after the used clicks on [Compare Images] button in the [Analysis Type] section of the program as shown in Figure 5. Selected nearness and distance

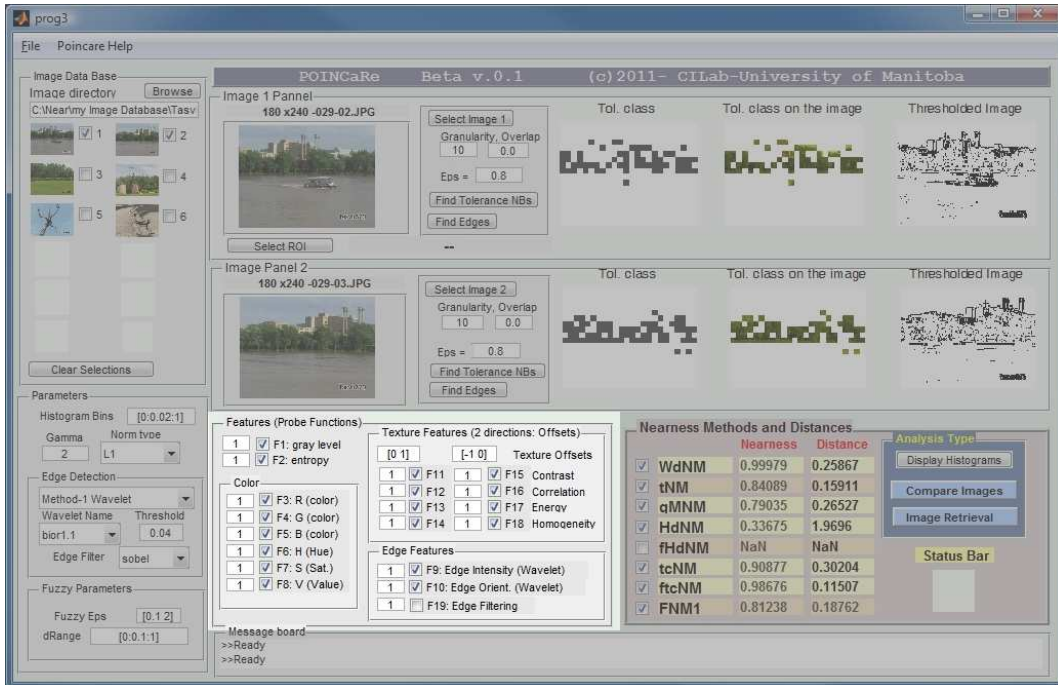


Figure 5: POINCaRe: Features and Methods Panel

measure will be shown in this panel. Nearness is a normalized number between 0 and 1 and distance is a non-normalized positive real number.

4.2 Histogram of features

Clicking on [Display Histograms] will open a new window where the histogram of all the selected feature values for the subimages in each image will be displayed. The histogram bins are used as mentioned earlier. Figure 6 shows an example of histograms generated by the program when only average R, G and B color components are selected as the features.

4.3 Finding tolerance neighborhoods and manual selection of a neighborhood

Clicking on Find Tolerance NBs for each image calculates the tolerance neighborhoods and the number of tolerance neighborhoods and a graphical representation of the size of each neighborhood is shown in each image panel. Moreover, by clicking the mouse on any point on an image in the image panel, the corresponding tolerance neighborhood around the selected subimage will be calculated and displayed on each image panel. Figure 3 shows sample tolerance neighborhoods selected this way after clicking on the centre of each image. Note that in this example, only R, G and B features are selected and the resulting tolerance neighborhoods represent part of the images which have almost the same color.

4.4 Edge detection

After clicking on Find Edges, edge intensity and orientation is calculated at each point and edge intensity is thresholded by the given threshold level to produce a binary image of detected edges shown in the last image window of the image panel.

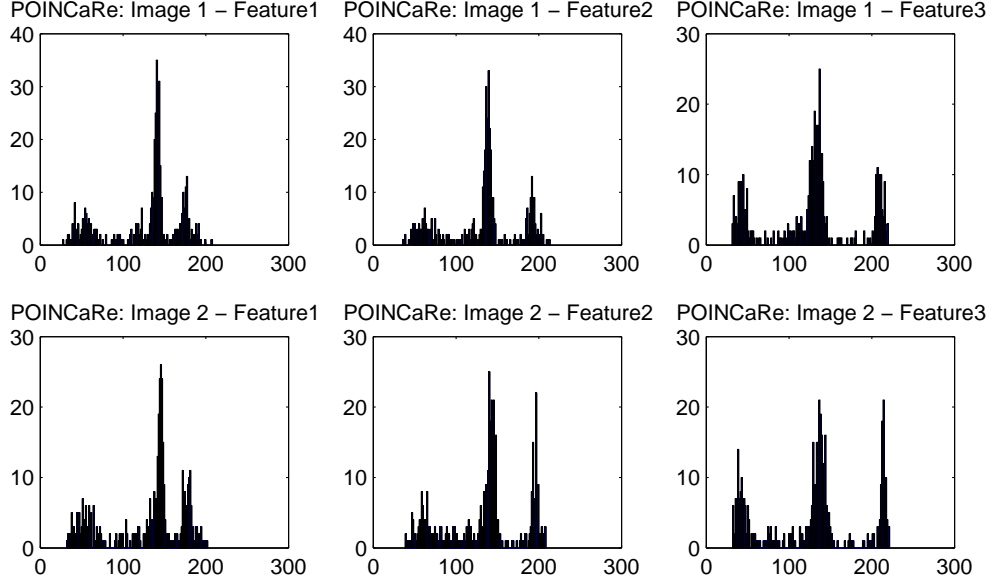


Figure 6: POINCaRe: Histograms of the subimage average R,G and B for the pair of images shown in figure 1

4.5 Selecting a region of interest (ROI)

Instead of image comparison between a query image (image 1) and a test image (image 2), the program can compare part of image 1 (a region of interest as a query image) with image 2. After clicking on [Select ROI] in image panel 1, the user can select a region of interest by clicking on the top left and bottom right corner of a region of interest in the image and the ROI will be selected as a set of subimages. Figure 7 show the step needed to select an ROI.

4.5.1 Upper and lower approximation of ROI by tolerance neighborhoods

After selection of an ROI, lower and upper approximations in the tolerance space will be automatically calculated and displayed on GUI. Roughly speaking, lower approximation of an ROI is the union of all the tolerance neighborhoods (/tolerance classes) which are a proper subset of the ROI. Moreover, upper approximation of an ROI is the union of all the tolerance neighborhoods (/tolerance classes) which have a non-empty intersection with the ROI. Therefore, objects in a lower approximation, have very similar descriptions to ROI and objects that do not belong to upper approximation have very different description from ROI. The exact definitions of these approximations and some examples as as follow

Definition 1. Lower Approximation $B_*(ROI)$ and Upper Approximation $B^*(ROI)$

Let Q is —the set of subimages in— a query image and Y is —the set of subimages in— a test image and $O = Q \cup Y$ is the set of all subimages. Let $ROI \subseteq Q$ is a region of interest in query image and $N_O^{\cong_{B,\epsilon}}$ is the set of all tolerance neighborhoods in the union of query and test image. Then:

$$B_*(ROI) = \bigcup \{A \in N_O^{\cong_{B,\epsilon}} \text{ such that } (A \cap Q) \subseteq ROI\} \quad (1)$$

$$B^*(ROI) = \bigcup \{A \in N_O^{\cong_{B,\epsilon}} \text{ such that } (A \cap Q) \cap ROI \neq \emptyset\} \quad (2)$$

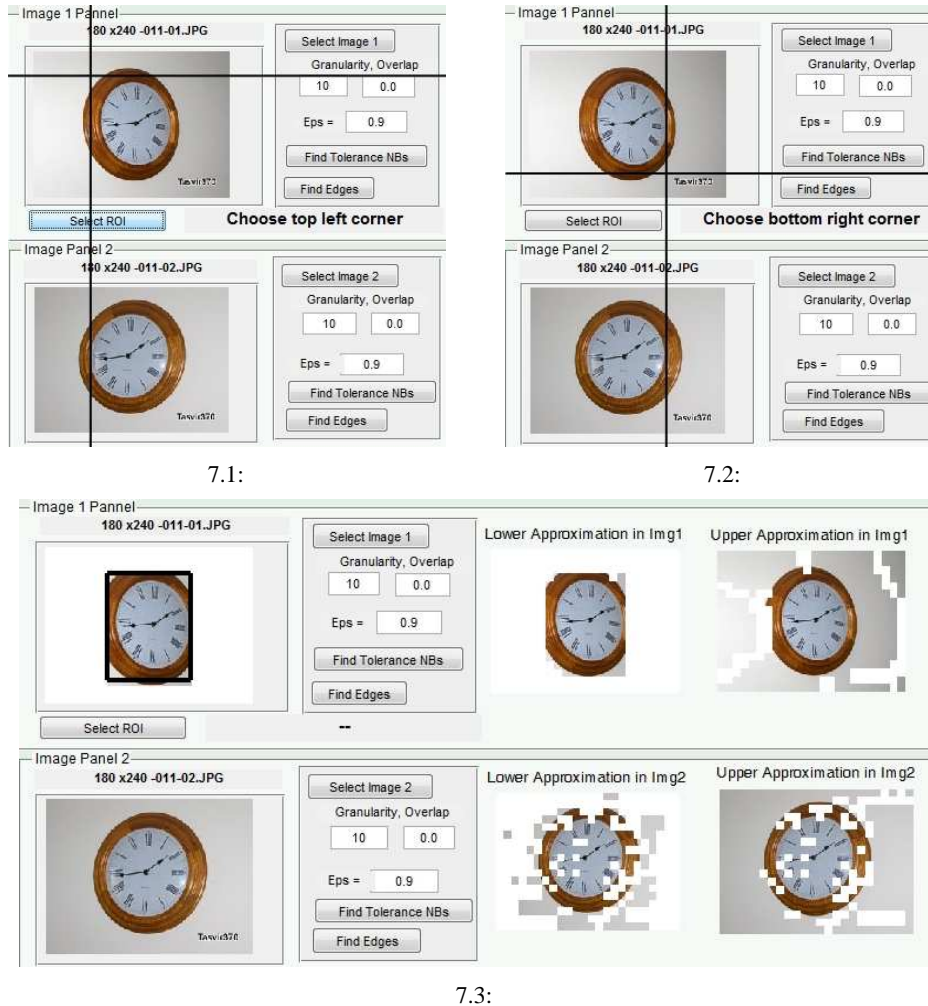


Figure 7: Steps in selecting an ROI and rough set approximation of ROI with tolerance neighborhoods

4.5.2 Nearness between an ROI and a test image

Region of interest (ROI) can be selected to show an object of interest for example in a query image where the rest of image is not important for image comparison problem. After an ROI is selected and displayed as a query itself, the user can push the [Compare two images] button to compare the ROI with test image. **NOTE:** many nearness measure need a relatively large number of subimages in each image to be compared with another image. Therefore selecting a small ROI that contains very few number of subimages yields a nearness measure which is meaningless.

4.6 Content-based image retrieval

In an image retrieval experiment, a query image (or a region of interest in the query image) is compared with all the images in an image dataset. The path to the images in image dataset is specified by [Image directory] edit box in the top left corner of the GUI. After choosing the path and selecting the query image in image panel 1, choosing the required parameters and probe functions, the user can start image retrieval by clicking on [Image Retrieval] button. All the images in image directory will be compared against the given query and the results will be stored and saved in an HTML file. Depending on the number

of images and the size of subimages, image retrieval may take some time to complete. A status bar and sand-watch icon in the bottom right corner of GUI update the user about the status of the experiment and the time to completion. After all the images are compared, the program automatically opens the default Internet browser and displays the data. The program is tested with Google chrome. If you cannot see the images or if the HTML output file has not been opened for any reason, go to the program directory and open the file: (Poincare-CBIR.html). Figure 8 is an example of an output file generated after the given query image is compared with 210 test image in Tasvir-3x70 data set of images. Images are ranked based on their similarity to the query image and the value of nearness and distance is shown for each image.

5 Visual Features

Describing a describable object is possible through a set of probe functions that produce a feature vector representing the object. An important question remains on what visual descriptions (features) are important and how to properly extract such features (the choice of probe functions). A correct answer to this question is highly subjective and depends on the application. The following sections explain all the visual features used in the program such as color, texture and edge information (table 1).

Table 1: Probe functions in use: $\mathcal{B} = \{\phi_1, \phi_2, \dots, \phi_{18}\}$

Probe function	Feature Type	Description
$\phi_1(x)$	Color	Average gray level of pixels
$\phi_2(x)$	Texture	Entropy of the gray level values
$\phi_3(x), \phi_4(x), \phi_5(x)$	Color	R, G and B color components
$\phi_6(x), \phi_7(x), \phi_8(x)$	Color	H, S and V color components
$\phi_9(x)$	Shape	Average intensities of edges
$\phi_{10}(x)$	Shape	Average orientation of edges
$\phi_{11}(x), \phi_{15}(x)$	Texture	Contrast
$\phi_{12}(x), \phi_{16}(x)$	Texture	Correlation
$\phi_{13}(x), \phi_{17}(x)$	Texture	Energy (Uniformity)
$\phi_{14}(x), \phi_{18}(x)$	Texture	Homogeneity

5.1 List of Probe Functions

The following probe functions are defined for a small subimage that contains local information about the image content. Each subimage is named a visual element and is considered here as a describable object x .

5.1.1 Average gray level value

The average of gray level values for all the pixels in a given subimage (x), is calculated as the first feature $\phi_1(x)$. In case of color images, the image is first converted into grayscale and then is used to calculate ϕ_1 .


5.1.2 Information content (entropy)

The entropy of each subimage is calculated using the gray level values of all pixels in each subimage. There are $N = 256$ different gray levels in an 8 bit digital image. For each gray level, the number of pixels in the

Poincare-CBIR.html

Poincare-CBIR.html

Query image



Query File: C:\Near\my Image Database\Tasvir-370-small\020-02.JPG
 Dataset Files: C:\Near\my Image Database\Tasvir-370-small
 Analysis Name: Poincare-CBIR











Rank	tNM				tcNM			
Rank	Image	File Name	Nearness	Distance	Image	File Name	Nearness	Distance
1		020-02.JPG	1	0		020-02.JPG	1	0
2		021-02.JPG	0.59201	0.40799		021-02.JPG	0.78374	0.46503
3		021-01.JPG	0.52508	0.47492		021-01.JPG	0.75385	0.49614
4		047-03.JPG	0.4573	0.5427		020-03.JPG	0.65915	0.58382
5		028-01.JPG	0.44453	0.55547		047-02.JPG	0.64025	0.5998

Figure 8: An example of an output file generated by POINCaRe after CBIR

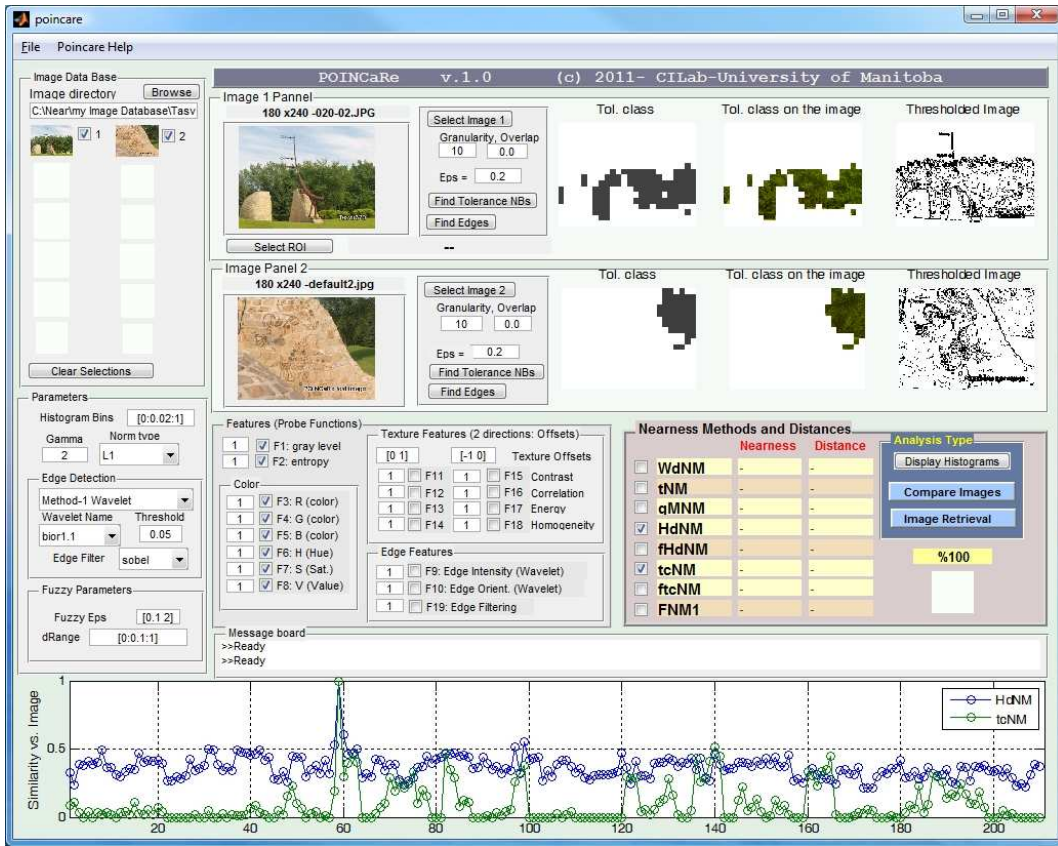


Figure 9: Plotting similarity between a query image and the rest of images

subimage having that gray level is counted and the Shannon entropy is defined as follows,

$$E = - \sum_{k=1}^N p_k \log_2(p_k),$$

where p_k is the normalized number of pixels with a gray level value that belongs to k^{th} level. This definition is based on a probabilistic view of the image where the gray level of the pixels is considered as a random variable, p_k is the probability of k^{th} gray level for each pixel and E (Shanon entropy) is a measure of the uncertainty associated with the random variable. The higher values of entropy correspond to more information content in the subimage.

5.1.3 Color features

$\phi_3(x), \phi_4(x), \phi_5(x)$ are the average *Red*, *Green* and *Blue* color components of the pixels respectively. $\phi_6(x), \phi_7(x), \phi_8(x)$ are the average *Hue*, *Saturation* and *Value* color components of the pixels.

5.1.4 Texture and statistical features

Texture features in this paper are defined based on the gray-level co-occurrence matrix (GLCM) ([4], see also [3, 12]). GLCM (also named gray-tone spatial-dependence matrix) can be used to define 14 different textural measures [4]. Elements of GLCM consist the relative frequencies with which two neighboring pixels (separated by a given offset δx and δy) with gray level values i and j , occur in the image [7].

$$C_{\Delta x, \Delta y}(i, j) = \begin{cases} \sum_{p=1}^n \sum_{q=1}^m 1, & \text{if } I(p, q) = i \text{ and } I(p + \Delta x, q + \Delta y) = j, \\ \text{otherwise.} \end{cases}$$

For notational convenience and with the understanding that the offset $(\Delta x, \Delta y)$ is known, let $p_{i,j}$ denote (i, j) th element of the normalized GLCM matrix ($p_{i,j} = C_{\Delta x, \Delta y}(i, j)/(m \times n)$). The following texture features are used in this paper based on GLCM.

- $\phi_{11}(x)$ (Contrast): intensity contrast between a pixel and its neighbors (Element difference moment of order 2) [2], [5]

$$\sum_i \sum_j (i - j)^k p_{i,j}.$$

- $\phi_{12}(x)$ (Correlation): correlation between a pixel and its neighbors is defined ([17]) as

$$\sum_i \sum_j \frac{(ij)p_{i,j} - \mu_x \mu_y}{\sigma_x \sigma_y},$$

where

$$\mu_x = \sum_i \sum_j i \cdot p_{i,j}, \quad \mu_y = \sum_i \sum_j j \cdot p_{i,j},$$

$$\sigma_x = \sum_i \sum_j (i - \mu_x)^2 \cdot p_{i,j} \quad \sigma_y = \sum_i \sum_j (j - \mu_y)^2 \cdot p_{i,j}.$$

- $\phi_{13}(x)$ (Energy or Uniformity): is defined ([5, 2]) as

$$\sum_{i,j} p_{i,j}^2.$$

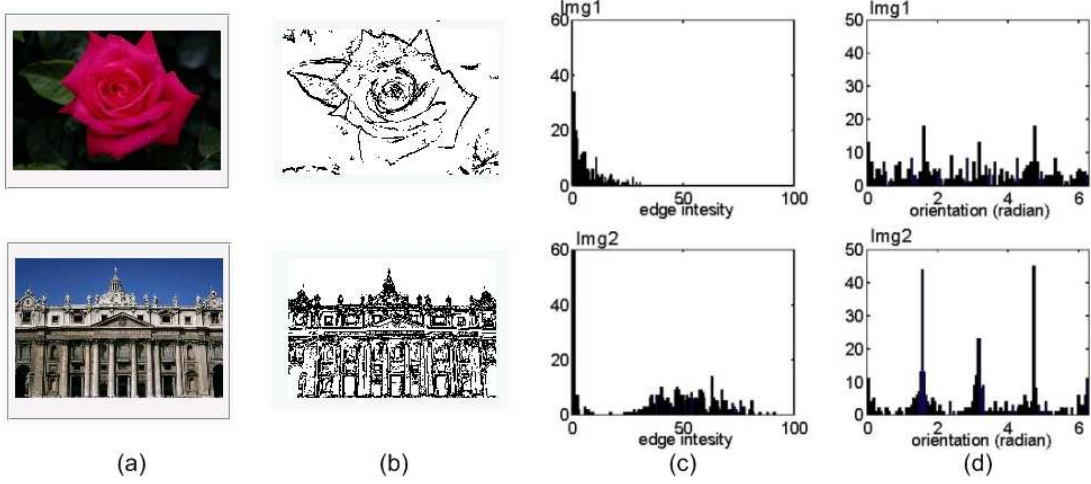


Figure 10: Sample images (a), edges after proper thresholding (b), histogram of dominant edge intensity and orientation in subimages (c,d)

- $\phi_{14}(x)$ (Homogeneity): is defined ([5]) as

$$\sum_i \sum_j \frac{p_{ij}}{1 + |i - j|}.$$

5.1.5 Edge features

Edges contain important information about the shape of the objects in an image. There are many methods for detecting the intensity and orientation of a possible edge at each pixel of an image (for example, Canny [1] or compass edge detectors [15, 16]). For sample edge intensity and edge orientation histograms produced by POINCaRe, see Figure 10. Here, the wavelet-based edge detection method by Mallat [6] is used. This method can be used with different wavelet functions at different scales to extract both the intensity and orientations of edges at different levels of details.

In each pixel location (x, y) , edge intensity and orientation at a given scale (2^j) is defined respectively as below:

$$M_{2^j}(x, y) = \sqrt{|W_{2^j}^1 f_{x,y}|^2 + |W_{2^j}^2 f_{x,y}|^2},$$

$$A_{2^j}(x, y) = \arctan \frac{W_{2^j}^1 f_{x,y}}{W_{2^j}^2 f_{x,y}}.$$

where $W_{2^j}^1 f_{x,y}$ and $W_{2^j}^2 f_{x,y}$ are 2-D discrete wavelet transforms of $f_{x,y}$ at each scale 2^j . Furthermore, the highest value of edge intensity and its corresponding orientation value (in radians) in each subimage x is shown with $\phi_9(x)$ and $\phi_{10}(x)$, respectively.

References

- [1] J. Canny, A computational approach to edge detection, IEEE Trans. Pattern Anal. Mach. Intell. 8 (6) (1986) 679–698, 11275.

- [2] R. C. Gonzalez, R. E. Woods, Digital Image Processing, Prentice Hall, 2007.
- [3] R. M. Haralick, Statistical and structural approaches to texture, Proceedings of the IEEE 67 (5) (1979) 786–804.
- [4] R. M. Haralick, K. Shanmugam, I. Dinstein, Textural features for image classification, Systems, Man and Cybernetics, IEEE Transactions on 3 (6) (1973) 610–621.
- [5] P. Howarth, S. Ruger, Robust texture features for still-image retrieval, Vision, Image and Signal Processing, IEE Proceedings - 152 (6) (2005) 868–874.
- [6] S. Mallat, S. Zhong, Characterization of signals from multiscale edges, Pattern Analysis and Machine Intelligence, IEEE Transactions on 14 (7) (1992) 710–732.
- [7] D. J. Marceau, P. J. Howarth, J. M. Dubois, D. J. Gratton, Evaluation of the grey-level co-occurrence matrix method for land-cover classification using spot imagery, Geoscience and Remote Sensing, IEEE Transactions on 28 (4) (1990) 513–519.
- [8] A. H. Meghdadi, Fuzzy tolerance neighborhood approach to image similarity in content-based image retrieval, Ph.D. thesis, University of Manitoba, supervisor: J.F. Peters (2012).
- [9] A. H. Meghdadi, J. F. Peters, Perceptual information system approach to image resemblance, in: J. F. P. Sankar K. Pal (ed.), Rough Fuzzy Image Analysis Foundations and Applications, CRC Press, 2010, pp. 8(1–22).
- [10] A. H. Meghdadi, J. F. Peters, Perceptual tolerance neighborhood-based similarity in content-based image retrieval and classification, International Journal of Intelligent Computing and Cybernetics 5 (3).
- [11] A. H. Meghdadi, J. F. Peters, S. Ramanna, Tolerance classes in measuring image resemblance, in: Knowledge-Based and Intelligent Information and Engineering Systems, vol. 5712 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2009, pp. 127–134.
- [12] P. P. Ohanian, R. C. Dubes, Performance evaluation for four classes of textural features, Pattern Recognition 25 (8) (1992) 819–833, doi: DOI: 10.1016/0031-3203(92)90036-I.
- [13] J. Peters, S. Naimpally, Applications of near sets, Amer. Math. Soc. Notices, *to appear*.
- [14] S. Ramanna, A. H. Meghdadi, J. F. Peters, Nature-inspired framework for measuring visual image resemblance: A near rough set approach, Theoretical Computer Science 412 (42) (2010) 5926–5938.
- [15] M. A. Ruzon, C. Tomasi, Color edge detection with the compass operator, in: Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on., vol. 2, 1999.
- [16] M. A. Ruzon, C. Tomasi, Edge, junction, and corner detection using color distributions, Pattern Analysis and Machine Intelligence, IEEE Transactions on 23 (11) (2001) 1281–1295.
- [17] L. K. Soh, C. Tsatsoulis, Texture analysis of sar sea ice imagery using gray level co-occurrence matrices, Geoscience and Remote Sensing, IEEE Transactions on 37 (2) (1999) 780–795.