Proximity System

Christopher Henry
Garrett Smith
ch.henry@uwinnipeg.ca
garrettwhsmith@gmail.com
University of Winnipeg

515 Portage Avenue

Winnipeg, Manitoba R3B 2E9

UM CI Laboratory Technical Report
Number TR-2012-021
August 24, 2012

University of Manitoba

Computational Intelligence Laboratory
URL: http://wren.ee.umanitoba.ca

Copyright © CI Laboratory, University of Manitoba 2011.

Project no. 2012-021

Proximity System

Christopher Henry
Garrett Smith
ch.henry@uwinnipeg.ca
garrettwhsmith@gmail.com
University of Winnipeg
515 Portage Avenue
Winnipeg, Manitoba R3B 2E9

CI Laboratory TR-2012-021

August 24, 2012

Contents

Abstract

This report introduces the Proximity System, an application developed to demonstrate descriptive-based topological approaches to nearness and proximity within the context of digital image analysis. Specifically, the system implements the descriptive-based intersection, compliment, and difference operations defined on four different types of neighbourhoods. Each neighbourhood can be considered a Region Of Interest (ROI), which plays an important role in discerning perceptual similarity within a single image, or between a pair of images. In terms of pixels, closeness between ROIs is assessed in light of the traditional closeness between points and sets and closeness between sets using topology or proximity theory. The contribution of this report is a detailed discussion on the Proximity System.

Keywords: Digital images, metric-free distance measure, probe functions, image analysis, near sets, nearness, neighbourhoods, proximity, topological structures.

1 Introduction

The Proximity System is an application developed to demonstrate descriptive-based topological approaches to nearness and proximity within the context of digital image analysis. The Proximity System grew out of the work of S. Naimpally and J. Peters [1–10], was also influenced by work reported in [11–14], and has resulted in one publication [15] (thus far). The Proximity System was written in Java and is intended to run in two different operating environments, namely on Android smartphones and tablets, as well as desktop platforms running the Java Virtual Machine. With respect to the desktop environment, the Proximity System is a cross-platform Java application for Windows, OSX, and Linux systems, which has been tested on Windows 7 and Debian Linux using the Sun Java 6 Runtime. In terms of the implementation of the theoretical approaches presented in the report, both the Android and the desktop based applications use the same back-end libraries to perform the description-based calculations, where the only differences are the user interface and the Android version has less available features (*i.e.* probe functions given in Definition3) due to restrictions on system resources.

The inspiration for the approach implemented in the Proximity System is an observation in [9] that the concept of nearness[1] is a generalization of set intersection. The idea follows from the notion of set description [10, §4.3], which is a collection of the unique feature vectors (n-dimensional real-valued feature vectors representing characteristics of the objects) associated with all the objects in the set. Describing sets in this manner, at some level, matches the human approach to describing sets of objects. Furthermore, in comparing disjoint sets of objects, we must at some level be performing a comparison of the descriptions we associate with the objects within the sets. Thus, a natural approach for quantifying the degree of similarity (*i.e.* the *nearness* or *apartness*) between two sets would be to look at the intersection of the sets containing their unique feature vectors.

The sets considered in the Proximity System are obtained from digital images. For instance, Regions Of Interest (ROI) play an important role in discerning perceptual similarity within a single image, or between a pair of images. In this work, four different ROIs are considered. Namely, a simple set of pixels, a spatial neighbourhood, a descriptive neighbourhood, and a hybrid approach in which the neighbourhood is formed by spatial and descriptive characteristics of the objects. In terms of pixels, closeness between ROIs can be assessed in light of the traditional closeness between points and sets and closeness between sets using topology or proximity theory [10, 16].

The approach reported here builds on the work of many others. The idea of sets of similar sensations was first introduced by J. H. Poincaré in which he reflects on experiments performed by E. Weber in 1834, and G. T. Fechner's insight in 1850 [17–20]. Poincaré's work was inspired by Fechner, but the key difference is Poincaré's work marked a shift from stimuli and sensations to an abstraction in terms of sets together with an implicit idea of tolerance. Next, the idea of tolerance is formally introduced by E. C. Zeeman [21] with respect to the brain and visual perception. Zeeman makes the observation that a single eye cannot identify a 2D Euclidean space because the Euclidean plane has an infinite number of points. Instead, we see things only within a certain tolerance. This idea of tolerance is important in mathematical applications where systems deal with approximate input and results are accepted with a tolerable level of error, an observation made by A. B. Sossinsky [17], who also connected Zeeman's work with that of Poincaré's. In addition to these ideas on tolerance, F. Riesz first published a paper in 1908 on the nearness of two sets [1, 5], initiating the mathematical study of proximity spaces and the eventual discovery of descriptively near sets. Specifically, Near set theory was inspired by a collaboration in 2002 by Z. Pawlak and J. F. Peters on a poem entitled "How Near" [2], which lead to the introduction of descriptively near sets [3, 4]. Next, tolerance near sets were also introduced by Peters [7, 8], which combines near set theory with the ideas of tolerance spaces and relations. Finally, a tolerance-based nearness measure was introduced in [12, 13].

---

[1]Introduced within the context of Riesz's proximity [5].

2 Description-based Set Operators

Many interesting properties can be considered by introducing the description of a set. The following gives definitions of new operators considered in the light of object descriptions (as originally reported in [15]). A logical starting point for introducing descriptive-based operators begins with establishing a basis for describing elements of sets. All sets in this work consist of perceptual objects.

Definition 1. Perceptual Object. *A perceptual object is something that has its origin in the physical world.*

A perceptual object is anything in the physical world with characteristics observable to the senses such that they can be measured and are knowable to the mind. In keeping with the approach to pattern recognition suggested by M. Pavel [22], the features of a perceptual object are quantified by probe functions.

Definition 2. Feature [23]. *A feature characterizes some aspect of the makeup of a perceptual object.*

Definition 3. Probe Function [3, 24]. *A probe function is a real-valued function representing a feature of a perceptual object.*

Next, a perceptual system is a set of perceptual objects, together with a set of probe functions.

Definition 4. Perceptual System [25]. *A perceptual system $\langle O, \mathbb{F} \rangle$ consists of a non-empty set O of sample perceptual objects and a non-empty set $\mathbb{F}$ of real-valued functions $\phi \in \mathbb{F}$ such that $\phi : O \to \mathbb{R}$.*

Combining Definitions 1 & 3, the description of a perceptual object within a perceptual system can be defined as follows.

Definition 5. Object Description. *Let $\langle O, \mathbb{F} \rangle$ be a perceptual system, and let $\mathcal{B} \subseteq \mathbb{F}$ be a set of probe functions. Then, the* description *of a perceptual object $x \in O$ is a feature vector given by*

$$\Phi_{\mathcal{B}}(x) = (\phi_1(x), \phi_2(x), \ldots, \phi_i(x), \ldots, \phi_l(x)),$$

where l is the length of the vector $\Phi_{\mathcal{B}}$, and each $\phi_i(x)$ in $\Phi_{\mathcal{B}}(x)$ is a probe function value that is part of the description of the object $x \in O$.

Note, the idea of a feature space is implicitly introduced along with the definition of object description. An object description is the same as a feature vector as described in traditional pattern classification [26]. The description of an object can be considered a point in an l-dimensional Euclidean space $\mathbb{R}^l$ called a feature space. Further, a collection of these points, *i.e.*, a set of objects $A \subseteq O$, is characterized by the unique description of each object in the set.

Definition 6. Set Description [10, §4.3]. *Let A be a set. Then the* set description *of A is defined as*

$$\Phi(A) = \{\Phi(a) : a \in A\}.$$

Example 1. *Let $\langle O, \mathbb{F} \rangle$ be a perceptual system, where O contains the pixels in Fig. 1, $A \subseteq O$, and $\mathcal{B} \subseteq \mathbb{F}$ contains probe functions based on the RGB colour model. Then, the set description of A is $\Phi(A) = \{\blacksquare, \blacksquare, \blacksquare, \blacksquare, \square\}$, where each coloured box represents the 3-dimensional real-valed rgb vector associated the box's colour.*

Next, J. Peters and S. Naimpally observed that, from a spatial point of view, the idea of nearness is a generalization of set intersection [9]. In other words, when considering the metric proximity, two sets are near each other when their intersection is not the empty set. Furthermore, they applied this idea to the concept of descriptive nearness in [10, §4.3] by focusing on the descriptions of objects within the sets. In this case, two sets are considered near each other if the intersection of their descriptions is not the empty set.
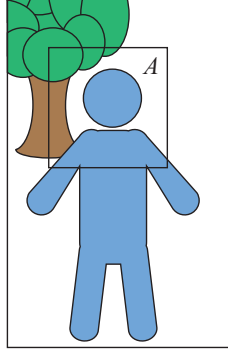
Figure 1: Example demonstrating Definition 6.

Definition 7. Descriptive Set Union. *Let A and B be any two sets. The* descriptive (set) union *of A and B is defined as*

$$A \underset{\Phi}{\cup} B = \{x \in A \cup B : \Phi(x) \in \Phi(A) \text{ or } \Phi(x) \in \Phi(B)\}.$$

Definition 8. Descriptive Set Intersection [9, 10]. *Let A and B be any two sets. The* descriptive (set) intersection *of A and B is defined as*

$$A \underset{\Phi}{\cap} B = \{x \in A \cup B : \Phi(x) \in \Phi(A) \text{ and } \Phi(x) \in \Phi(B)\}.$$

Example 2. *Let $\langle O_1, \mathbb{F} \rangle$ and $\langle O_2, \mathbb{F} \rangle$ be perceptual systems corresponding to Fig. 2a & 2c, respectively, where the perceptual objects and probe functions are defined in the same manner as Example 1. Moreover, let the blue rectangles in Fig. 2b (resp. Fig. 2d) represent two sets, A, B, for which the descriptive intersection is considered. Then, the inverted pixels (i.e. $p_i = \{c_i, r_i, 255 - R_i, 255 - G_i, 255 - B_i)^T\}$) within these sets represent their descriptive intersection, i.e. the inverted pixels represent the objects with matching descriptions in both sets.*
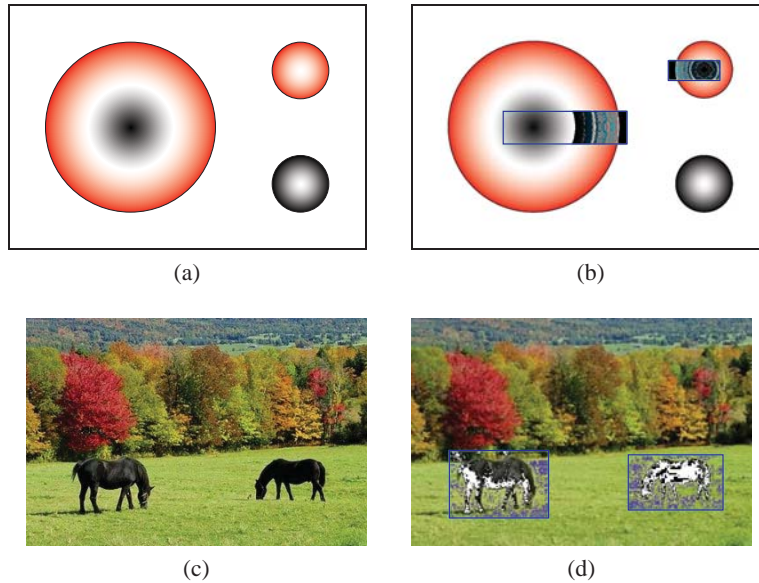

Figure 2: Example demonstrating Definition 8.

Definition 9. Descriptive Set Difference. *The* descriptive (set) difference *(or* descriptive difference set*) between two sets A and B is defined as*

$$A \underset{\Phi}{\setminus} B = \{x \in A : \Phi(x) \notin \Phi(B)\}.$$

Example 3. *The descriptive difference between the sets introduced in Example 2 are given Fig. 3. In this case, the inverted pixels represent all the objects that do not have matching descriptions in the other set.*
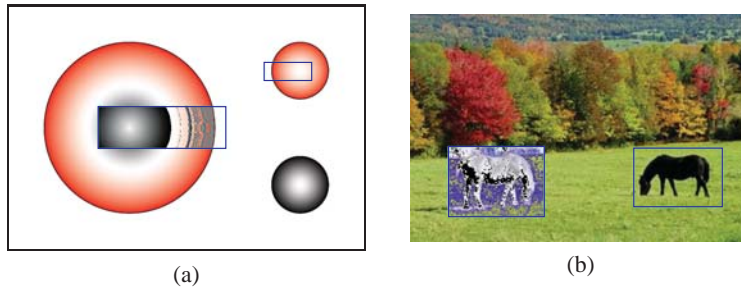

(a)

(b)

Figure 3: Example demonstrating Definition 9.

Definition 10. Relative Descriptive Complement. *Let A be a set, and let $B \subseteq A$. Then, the relative descriptive complement of B in A is defined as*

$$\underset{\Phi}{\complement}_A (B) = \{x \in A : \Phi(x) \notin \Phi(B)\}.$$

Definition 11. Descriptive Set Complement. *The descriptive (set) complement of a set A in the universe U is defined as*

$$\underset{\Phi}{\complement} (A) = \underset{\Phi}{\complement}_U (A) = U \underset{\Phi}{\setminus} A$$

Example 4. *Considering the perceptual systems introduced in Example 2, the descriptive complement of each set represented by a blue rectangles in Fig. 4 is given by the inverted pixels. In other words, the inverted pixels represent objects that do not have matching descriptions to those contained inside the blue rectangle.*

(a)

(b)

Figure 4: Example demonstrating Definition 11.

3 Application to Description-based Neighbourhoods

This section outlines several types of neighbourhoods to which the above operators can be applied.

Definition 12. Spatial Neighbourhood (without focus). *A spatial neighbourhood without focus is a traditionally defined set,* i.e. *it is a collection of objects.*

The set operands from Examples 1-4 are examples of spatial neighbourhoods without focus, which are simply collections of pixels.

Definition 13. Spatial Neighbourhood (with focus). *Let $x, y \in O$ be perceptual objects, let $d(x, y)$ denote any form of distance metric between x and y, and let $N_d(x, \mathfrak{r}) = \{y \in O : d(x, y) < \mathfrak{r}\}$ denote an open ball (using any distance metric $d(x, y)$) with with radius $\mathfrak{r} \in [0, \infty)$, and centre x. Then, a* spatial neighbourhood *with focus x is defined as $N_d(x, \mathfrak{r})$ for some $x \in O$.*

The term *focus* used here is synonymous with the *centre* associated with an open ball, yet is preferred (in this context) since spatial neighbourhoods may still have an object that can be considered the spatial centre of the set. Moreover, the term *focus* implies conscious directed attention, which is more in line with the idea of using description-based neighbourhoods as part of a formal framework for quantifying the perceptual nearness of objects and sets of objects.

Definition 14. Description-Based Neighbourhood. *Let $x, y \in O$ be perceptual objects with object descriptions given by $\Phi(x), \Phi(y)$. Then, a* description-based neighbourhood *is defined as*

$$\mathfrak{N}_{x,\varepsilon} = \{y \in O : |\Phi(x) - \Phi(y)| < \varepsilon\}.$$

A point y is a member of $\mathfrak{N}_{x,\varepsilon}$, if and only if, $|\Phi(x) - \Phi(y)| < \varepsilon$.

Example 5. *Consider a perceptual system defined in a manner similar to Example 1. Then, the inverted pixels in Fig. 5b represent a description-based neighbourhood, where the focus (centre) of the neighbourhood is represented by the enlarged dark pixel. Note, $\varepsilon = 0.23$ (out of a maximum of $\sqrt{3}$) was used to generate this neighbourhood.*

(a)   (b)

(c)   (d)
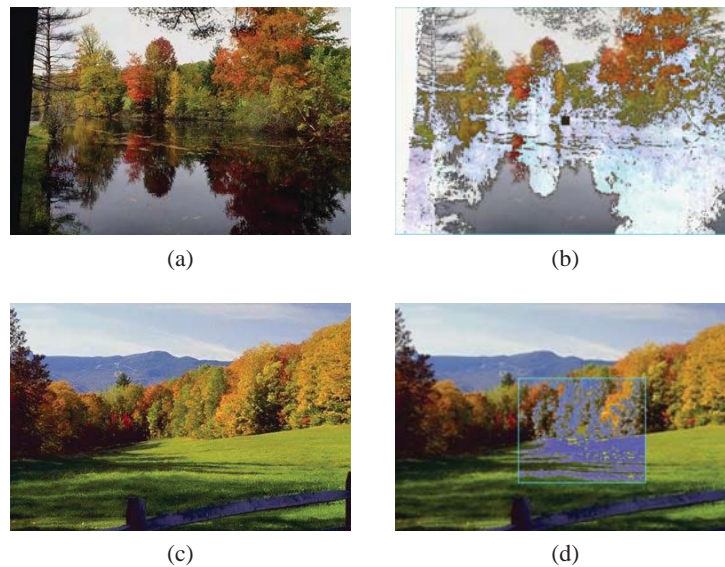
Figure 5: Example demonstrating Definitions 13 & 14.

Definition 15. Bounded-Descriptive Neighbourhood. *Let $x, y \in O$ be perceptual objects with object descriptions given by $\Phi(x), \Phi(y)$. Then, a* bounded-descriptive neighbourhood *is defined as*

$$\mathfrak{N}_{x,\varepsilon}^{\circ} = \{y \in O : |\Phi(x) - \Phi(y)| < \varepsilon| \text{ and } y \in N_d(x, \mathfrak{r})\}.$$

In other words, a point y is a member of $\mathfrak{N}^{\circ}_{x,\varepsilon}$, if and only if, y is descriptively similar to some point z inside $N_d(x, \mathfrak{r})\}$ with centre x and radius $\mathfrak{r}$.

Example 6. *As in all the previous examples, assume a perceptual system similar to Example 1. Then, the inverted pixels in Fig. 5d represent a bounded-descriptive neighbourhood, where the focus (centre) of the neighbourhood is the enlarged green pixel. Here, $\varepsilon = 0.17$ was used to generate this neighbourhood.*

4 Metric-Free Nearness Measure

Next, a metric-free description-based nearness measure using the descriptive operators introduced in Section 2 is presented, which is related to work on a tolerance-based nearness measure reported in [12, 13]. Furthermore, the approach presented here has direct application to image analysis and is related to the rough set image analysis approaches reported in [27–34]. Similarly, this measure can be applied to the problem of content-based image retrieval [35] in a manner analogous to the tolerance nearness measure approached taken in [36–38]. As in the case of the tolerance nearness measure, both approaches aim to quantify the similarity between sets of objects based on object description. However, the tolerance nearness measure is obtained using tolerance classes (see, *e.g.*, [39]) obtained from the union of the sets under consideration, while the description-based nearness measure is based on the descriptive operators presented in this article. The idea that motivated this measure comes from the observation in [9] that nearness is considered a generalization of intersection. Intuitively speaking, we perceive sets of objects to be similar or near in some manner when they share common characteristics. Thus, if considering set descriptions (as given in Definition 6), the descriptive intersection should not be empty if we consider the sets to be similar with respect to one or more features. Keeping these ideas in mind, a metric-free description-based nearness measure, dNM, is defined as follows.

Definition 16. Metric-Free Description-Based Nearness Measure. *Let $X, Y \subseteq O$ be sets of perceptual objects within a perceptual system. Then, a* metric-free description-based nearness measure *is defined as*

$$dNM(X, Y) = 1 - \frac{|X \underset{\Phi}{\cap} Y|}{|X \cup Y|}.$$

The nearness measure produces values in the interval $[0, 1]$, where, for a pair of sets X, Y, a value of 0 represents complete resemblance, and a value of 1 intimates no resemblance.

Example 7. *Consider a perceptual system defined in a manner similar to Example 1, except using only probe functions based on the red and green components from the RGB colour model. Then, the dNM values of the images in Fig. 6 are given in Table 1, where two different types of neighbourhoods are considered in the descriptive intersection. Specifically, Fig. 6a & 6b contain spatial neighbourhoods, and Fig. 6c & 6e depict the bounded-descriptive neighbourhoods (obtained with $\varepsilon = 0.23$) that are used in generating the descriptive intersection given in Fig. 6d & 6f. Notice, as expected, in all cases the dNM is lower when comparing the two mushrooms. Also, there are no objects in the two neighbourhoods in Fig. 6e that have matching descriptions. Hence, the empty intersection and $dNM = 1$.*

5 Practical Application

Note, the practical application of Definitions 8, 9, & 11 to real world problems may require relaxing their implicit equivalence requirement since these types of applications tend to work with approximate input data and solutions can tolerate a certain level of error [9, 12, 17]. In other words, instead of requiring object
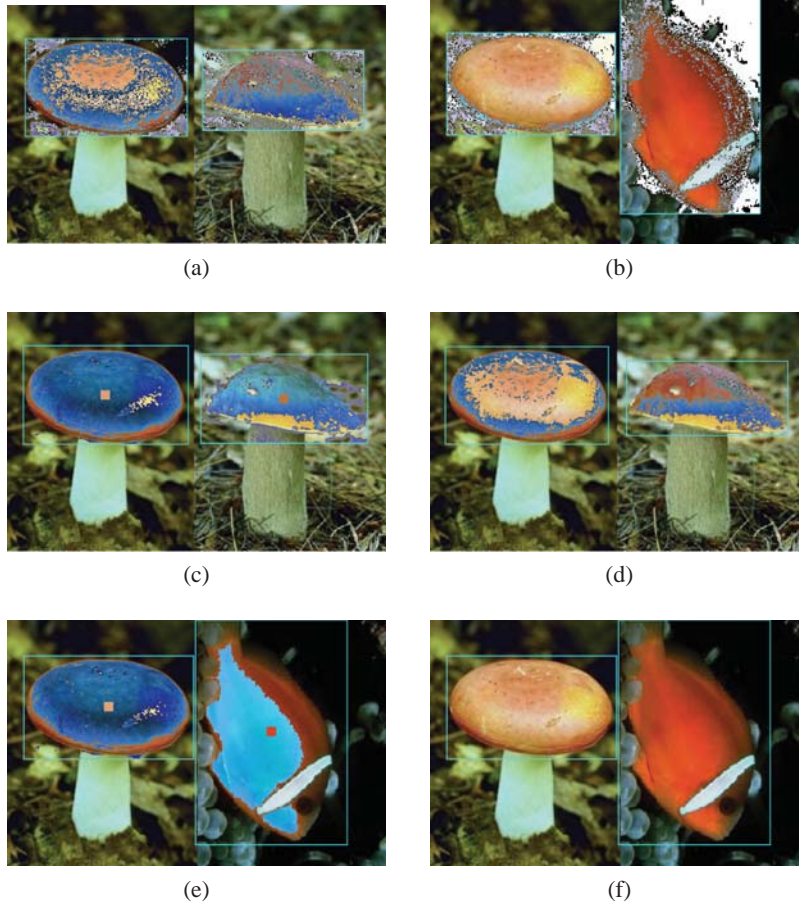
Figure 6: Example demonstrating Definition 16.

Table 1: Nearness Measure Values for Images in Fig. 6

Image	Neighbourhood Type	dNM
Fig. 6a	Spatial (without focus)	0.54
Fig. 6b		0.74
Fig. 6d	Bounded-Descriptive Neighbourhood	0.37
Fig. 6f		1

descriptions to match in order to appear within the output of the descriptive intersection, better results may be achieved if the difference between object descriptions are allowed to be within some ε, as, *e.g.*, in the following equation

$$A \underset{\Phi}{\cap} B = \{a \in A, b \in B : |\Phi(a) - \Phi(b)| \leq \varepsilon\}.$$

Similar modifications can also be made to Definitions 9, & 11. In fact, this functionality has been built into the Proximity System and is described in Section 6.5 below. Moreover, as shown by Table 2, this approach was used to improve the results from Example 7 by increasing the difference in dNM between sets considered perceptually near (*i.e.* the two mushrooms) and those that are not (*i.e.* the mushroom and the fish).

Table 2: Improved Nearness Measure Values for Images in Fig. 6

Image	Neighbourhood Type	dNM
Fig. 6a	Spatial (without focus)	0.30
Fig. 6b		0.61
Fig. 6d	Bounded-Descriptive Neighbourhood	0.33
Fig. 6f		0.99

6 Desktop Application

The following details the usage of the desktop version of the Proximity System. To start the application, open the jar file using the Java runtime engine. If required, the the Java runtime can be downloaed from http://www.java.com/en/download/index.jsp.

6.1 Opening Images

The Proximity System desktop application supports JPEG, PNG, GIF, and BMP file formats. On startup, the user must select an image via the *Select Image* button (see, *e.g.*, Fig. 7). A new image can be opened at any time by selecting *File → Open* or using the key combination *Ctrl + O*. If an image is already opened, a confirmation will appear to ensure the user wants to proceed with the action. Finally, recently opened images can be selected by choosing the corresponding item in the *File → Open Recent* menu.
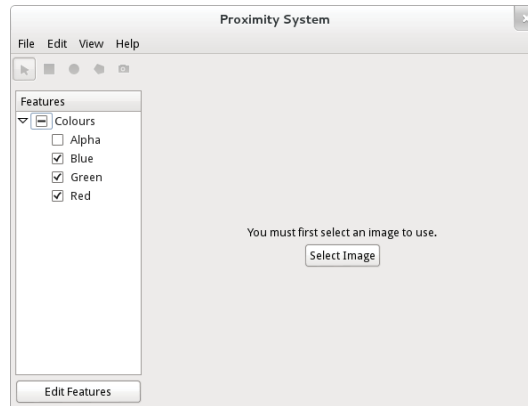

Figure 7: The Proximity System on startup.

6.2 Image Snapshots

Once an image has been opened, the user may save a snapshot of the current display. There are three ways to open the snapshot dialog (shown in Fig. 8): Selecting the snapshot icon in the main toolbar, *File → Snapshot*, or *Ctrl + S*. Once the snapshot dialog has been opened, the user can name of the file and the directory in which to save the file, which results from selecting the *Save* button. The user will be prompted before overwriting a file or if any error occurred while saving the image.

6.3 View

The following subsections detail the features that can change the view of the current image.
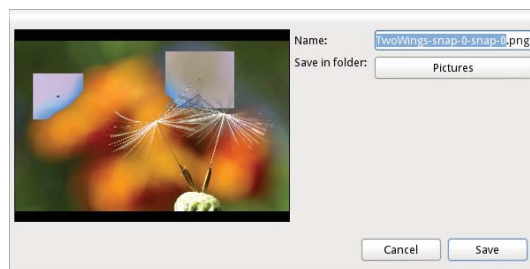
Figure 8: Example demonstrating the screen capture dialog.

6.3.1 Panning

To pan the view of the image canvas click and drag using the *middle mouse button*. Or, to centre the image, select *View → Center*.

6.3.2 Zooming

The magnification of the image can be changed using the following actions:

- *Ctrl + Scroll* the mouse wheel;

- *View → Zoom → Zoom In* and *Zoom Out*; or

- using the + and - keys.

6.3.3 Full Size

The image can be displayed at its full size by selecting *View → Zoom → Zoom 1:1* or by using the *1* key.

6.3.4 Zoom to Image

To fit the image within the Proximity System canvas, select *View → Zoom → Zoom to Image* or use *3* key. This will centre the image and set the zoom to ensure the image fits within the image canvas.

6.4 ROIs

The following sections describe the process of adding ROIs to the Proximity System. Note, the Proximity System calculates Definitions 8, 9, & 11 as soon as a new ROI is added by the user. For instance, consider the cases where a user adds a first, second, and third ROI to the system, denoted by A, B, C, respectively. Immediately after the first ROI is added, the calculations for the descriptive intersection, set difference, and compliment are started (in this case the output of the intersection and difference operations is trivial). Similarly, these operations are re-calculated after the addition of a second ROI. Here, the system will calculate $A \underset{\Phi}{\cap} B$, $A \underset{\Phi}{\setminus} B$, and $\underset{\Phi}{\complement} (A \cup B)$. Then, the following operations are started once a third ROI is specified by the user. Namely, $(A \underset{\Phi}{\cap} B) \underset{\Phi}{\cap} C$, $(A \underset{\Phi}{\setminus} B) \underset{\Phi}{\setminus} C$, and $\underset{\Phi}{\complement} (A \cup B \cup C)$. This process is continued for each new ROI added by the user, and the time to completion of each operation is specified by the progress bar.
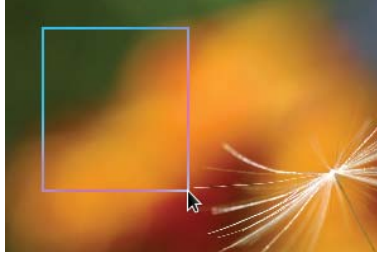
Figure 9: Example demonstrating the addition of a ROI.

6.4.1   Rectangular and Oval ROIs

Rectangle and oval ROIs can be added to an image by using the corresponding tools on the toolbar and performing the following steps (as demonstrated in Fig. 9).

1. Select either the rectangle or oval drawing tool.

2. Move the cursor to a point within the image pane that represents a corner of the new ROI.

3. Click and drag the cursor to the opposite corner of the ROI and release. Press the *escape* key to cancel adding the ROI while dragging the mouse.

6.4.2   Polygonal ROIs

Polygon ROIs are added to the image using the polygon tool and can have an arbitrary number of vertices (see, *e.g.*, Fig. 10).

Figure 10: Example demonstrating the addition of a polygon ROI.

1. Click the points within the image to form vertices of a polygon. A previously placed vertex my be removed by pressing the *backspace* key. Press the *escape* key to cancel adding the ROI at any time.

2. Repeat the previous step until all desired points have been added.

3. The polygon can be completed by either pressing the *enter* key, or clicking on the first vertex, which is indicated by a line connecting the first and last vertices.

6.4.3   Selecting ROIs

The selection tool is used to select ROIs that have been added to the image. A dotted box surrounding the ROI indicates a ROI has been selected. To select an ROI, click the desired ROI or drag a box around the desired ROIs. An example of two selected ROIs is given in Fig. 11.
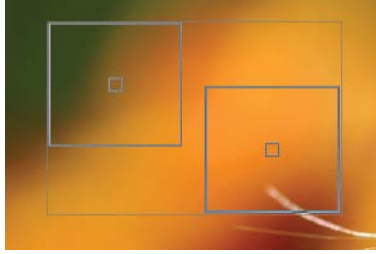
Figure 11: Example demonstrating two selected ROIs.

6.4.4 Deleting ROIs

ROIs can be removed from the image in three ways. First, a ROI must be selected before it can be removed. Then, the ROI can be removed using any of the following.

- *Right click* a selected ROI and select *Delete* from the context menu

- *Edit → Delete*

- *Delete* key

6.4.5 Moving ROIs

Once a ROI has been added to the image, the selection tool can be used to move the ROI (within the bounds of the image). This can be accomplished by selecting and dragging the ROI with the mouse.

6.5 Descriptive-based Operators and Neighbourhoods

Above the image canvas are the headings *Regions*, *Neighbourhoods*, *Intersection*, *Compliment* and *Difference*. The *Neighbourhood* tab is used to display the ROIs given in Definitions 12 & 14[2], while the latter three headings provide the output corresponding to the operators given in Definitions 8, 9, & 11. Recall, as was mentioned in Section 6.4, the output of these operators is calculated immediately after each new ROI is added to the system (via the *Regions* tab), and the time to completion is specified by the progress bar.

6.5.1 Epsilon

As was mentioned in Section 5, real world applications may benefit from allowing objects in which their descriptions differ by some ε to satisfy the conditions of Definitions 8, 9, & 11. For example, redefining Definition 8 as

$$A \underset{\Phi}{\cap} B = \{a \in A, b \in B : |\Phi(a) - \Phi(b)| \leq \varepsilon\}. \tag{1}$$

As a result, at a cost to performance, each of the tabs *Neighbourhoods*, *Intersection*, *Compliment* and *Difference* allow the user to specify a separate value of ε, which can be used to relax the equivalence requirement of each definition. Thus, $\varepsilon = 0$ specifies the operations given in Definitions 8, 9, & 11, while a value of $\varepsilon > 0$ redefines these operations as, *e.g.*, the descriptive set intersection given in Eq. 1. Finally, the *Neighbourhoods* tab also allows the user to specify ε, which corresponds to the value from Definition 15. Here, ε determines the membership of the neighbourhoods, which are used as operands for the descriptive-based operations in the other tabs.

---

[2]Note, Definition 13 can be realized by creating a circular ROI, and Definition 14 occurs when the ROI is the selected as the entire image (in conjunction with the feature described in Section).

6.5.2 Using Neighbourhoods

Within the Proximity System, the operands to the descriptive intersection, compliment, and difference operators are either[3] spatial neighbourhoods (without focus) or bounded-descriptive neighbourhoods. By default, the calculations are performed with spatial neighbourhoods. Selecting the *Use Neighbourhoods* checkbox changes the operands to bounded-descriptive neighbourhoods and automatically restarts the calculations with the new input parameters. Note, the large square within the ROI represents the centre pixel used to determine the membership of the bounded-descriptive neighbourhood and may hidden and shown from *View → Show Pivots*.

6.5.3 Descriptive Intersection Output

Besides the result of the descriptive set intersection, the *Intersection* tab also displays the metric-free description-based nearness measure (given in Definition 16) under the heading *Degree*, the cardinality of the descriptive union, and the cardinality of the descriptive intersection.

6.5.4 Progress Bar

Any change to the parameters in the Proximity System causes the system to recalculate the descriptive neighbourhoods, intersection, difference, and complement. The time to completion of the currently viewed operation is specified by the progress bar at the bottom of the image canvas.

6.6 Probe Functions

The probe functions used to determine object descriptions are displayed in the pane to the left of the application window. Probe functions can be enabled and disabled by checking and unchecking their corresponding boxes. Entire categories can be enabled and disabled in the same manner. The status of features affects all tabs and is remembered between sessions.

6.6.1 Adding Probe Functions

Additional probe functions can be added to the Proximity System using the dialog shown in Fig.12. The input to the dialog is a class file extending the ProbeFunc class described in Section **comeback**. The steps to add new probe functions are as follows.

1. Click the *Edit Features* button below the features pane.

2. Click *Browse* and select a folder containing the desired class files.

3. Select the probe functions you would like to add.

4. Enter a category name in the *Category* text field.

5. Select *OK*

6.6.2 Removing Probe Functions

To remove features or categories right click the feature or category and click *Remove*. Any default features or categories that are removed will return the next time the application is started.

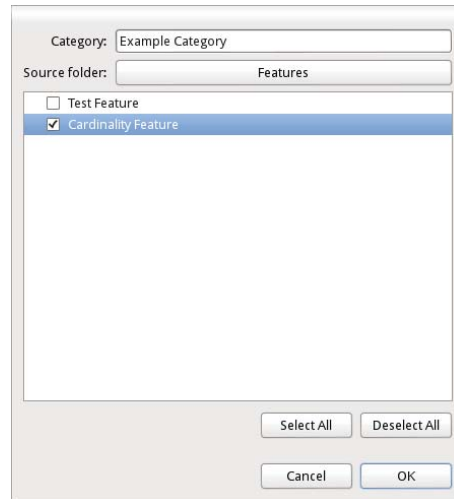[3]As was mentioned in Section 6.5, Definitions 13 & 14 are possible.

Figure 12: Example demonstrating the addition of user-defined features.

6.7   Implementation

The Proximity System desktop application was developed using the Eclipse IDE on Debian Linux. The interface was built using the SWT library with helper classes from the JFace toolkit. The WindowBuilder Pro Eclipse plugin was used to design the interface of the application. The cross platform Jar package was created with an Ant script using SWTJar.

7   Android Application

The Android-based version of the Proximity System runs on the Android platform. The application was tested on 4.0.3 and should be backwards compatible down to version 2.2 (although older phones my lack sufficient resources). The application can be downloaded via the Google Play Store **INSERT LINK ONCE UPLOADED**. Fig. 13 contains a screenshot of the Android version.

Figure 13: GUI of the Android-based Proximity System.

7.1 Opening Images

On startup, the user will be prompted to select an image via their system's image selection application. Opening a new image requires restarting the application.

7.2 Image Snapshots

The user may save a snapshot of the current display at any time by selecting *Snapshot* from the menu. The images are saved in the Proximity-System folder within the system's pictures directory. A sample snapshot is given in Fig. 14.

Figure 14: Sample snapshot from the Android-based Proximity System.

7.3 View

The following subsections detail the features that can change the view of the current image.

7.3.1 Panning

To pan the current view, press and *drag* your finger across the image.

7.3.2 Zooming

The magnify the current view, use a *pinch* gesture, or *double-click* to toggle between a zoomed-in and zoomed-out view of the image.

7.4 ROIs

The detailed discussion on ROIs in Section 6.4 also applies to the Android-based version and will not be repeated here.

7.4.1 Rectangular and Oval ROIs

Rectangle and oval ROIs can be added to an image by selecting *Add a ROI*, represented by the the *Plus* icon, from the action bar. Once selected, the user can choose the shape of the ROI from the drop-down menu in the action bar. Drag the edges of the ROI (with your finger) to grow and shrink to the desired size. Drag within the ROI to move it over the image. The view will ensure the ROI stays within the screen. Select the *Check* icon to finalize the ROI. An example of adding a rectangular ROI is shown in Fig. 15.

Figure 15: Example demonstrating adding a rectangular ROI.

7.4.2 Polygonal ROIs

Polygon ROIs are added to the image by selecting the *Plus* icon from the action bar and selecting *Polygon* from the drop down menu. Tap the screen to add vertices, which can be moved by dragging them. A vertex can be removed by dragging it off the image. Select the *Check* icon to finalize the ROI. An example of adding a polygon ROI is shown in Fig. 16.
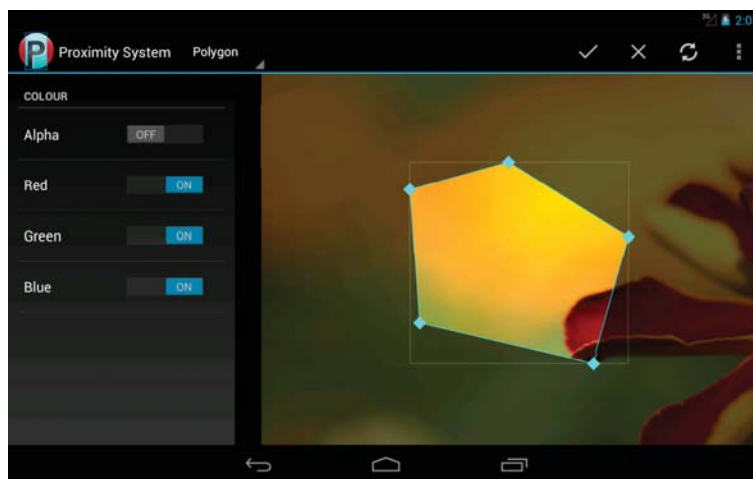

Figure 16: Example demonstrating adding a polygon ROI.

7.4.3 Clearing ROIs

To clear all the regions, open the overflow menu (represented by the three vertical boxes in the action bar or the device's hardware menu button) and select *Clear Regions*.

7.5 Descriptive-based Operators and Neighbourhoods

A discussion of the different navigation tabs was presented in Section 6.5 is not repeated here.

7.6 Epsilon

The *Set Epsilon* item in the action bar is used to set the epsilon of the current tab. See Section6.5.1 for a discussion on ε.

7.6.1 Using Neighbourhoods

The *Use Neighbourhoods* item in the overflow menu is used to set using neighbourhoods in the current tab. Again, Section 6.5.2 also applies to the Android-based version of the Proximity System.

7.6.2 Descriptive Intersection Output

Section 6.5.3 details the output provided on the descriptive intersection tab's action bar and within its overflow menu.

7.6.3 Progress Bar

The functionality of the progress bar is the same as given in Section6.5.4, however, its location is above the action bar as depicted in Fig. 17.

Figure 17: Example showing depicting progress while finding a neighbourhood.

7.7 Probe Functions

The probe functions available in the Proximity System are displayed differently depending on screen size. On tablets and other large-screened devices, the probe functions are displayed on a pane to the left of the image canvas and can be toggled on and off. On phones and small-screened devices, the features are displayed on a separate screen that can be opened from the overflow menu (see, *e.g.*, Fig.18). The features used to calculate the properties of the image can be turned on or off using their corresponding switches.

7.8 Implementation

The Android-based Proximity System was developed using theEclipse IDE with the NDK plugin on Debian Linux. ActionBarSherlock was used to make the application backwards compatible with pre-Gingerbread devices.
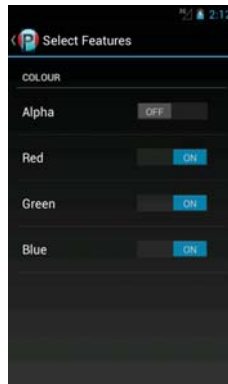
Figure 18: Probe function selection dialog for small screen devices.

8 Java Library

Both the desktop and Android based applications share a common library to perform the description-based operations. The Proximity System library has two primary classes with some additional helpers classes, and an image specific implementation used in the applications.

8.1 ProbeFunc.java

ProbeFuncs are Java objects used to map a perceptual object to a probe function (feature) value. The Probe-Func class is a generic abstract class with a map method that maps perceptual objects to their corresponding probe function output. This class also has a minimum and maximum value used to normalize the result of the map method, which is necessary to ensure that the output of probe functions with large magnitude do not dominate the distance calculation. The toString method is used to provide the name of the feature within both applications, and consequently, should be overridden. An example ProbeFunc is given in Listing 1.

Listing 1: An Example ProbeFunc

```java
import ca.uwinnipeg.proximity.ProbeFunc;

// The class must be in the default package, that is you can not
// use the package line, in order for it to be added.

// ProbeFunc is a generic class so you must give it the type
// of objects it will operate on.
public class ExampleFunc extends ProbeFunc<Object> {

  public ExampleFunc() {
    // 0 is the minimum value and 1 is the maximum value, these
    // must be set properly to ensure the value is normalized.
    super(0, 1);
  }

  // This maps an object to a value between the minimum and
  // maximum   values inclusive.
  // In this example it simply returns 1 for all objects.
  public double map(Object o) {
    return 1;
  }
```

```
  // This returns the string used as the name of your feature.
  public String toString() {
    return "Example Feature";
  }
}
```

8.2 Optimizations

The code used to calculate the description-based operations presented in this report needed to be optimized in order to run to completion without running out of memory. In particular, memory resources are tightly regulated by the Android OS to ensure the system remains responsive, and our earlier versions of the Java library could not run to completion within this memory constraint. Finally, note the difference in the following discussion between perceptual objects (*i.e.* pixels in the case of the Proximity System), and objects defined in traditional Java programming.

The PerceptualSystem class implements the methods used to calculate the output of probe functions. Perceptual objects and probe functions must be added to a PerceptualSystem object before probe function values can be calculated. Each Perceptual object is given an index number when it is added to the PerceptualSystem object. Probe function calculations are then made using object indices. This allows arrays to be used rather than list objects, which removes some overhead (especially where look ups are concerned).

Maps of unique object description are created between objects indicies and their corresponding object descriptions, which greatly reduced the overhead with comparisons. Specifically, as a result of this modification, comparisons between sets are made solely on set descriptions (*i.e.* lists of unique object descriptions associated with a set), and then the lists of objects associated with the matched descriptions can be combined into the final result. This optimization was particularly important since these comparisons were found to be one of the most time consuming part of the calculation.

For example, the descriptive intersection is calculated on two sets A and B by comparing $\Phi(A)$ to $\Phi(B)$ using an additional set $\Phi(C)$, which is a copy of $\Phi(B)$. Object descriptions will subsequently be removed from $\Phi(C)$ during calculation of the descriptive intersection, causing it to become a subset of $\Phi(B)$. During the calculation process, an object description $a \in \Phi(A)$ is compared to each object description $b \in \Phi(B)$. If a matching description[4] is found, both descriptions are marked as matched and the description from $\Phi(B)$ is removed from $\Phi(C)$, *i.e.* $\Phi(C) \leftarrow \Phi(C) \backslash b$. Once a match occurs, a is then compared to the remaining object descriptions in $\Phi(C)$, starting at the index of b. Any additional matches are also removed from $\Phi(C)$. Each new iteration starts by comparing an object in $\Phi(A)$ with $\Phi(B)$, and only switches to the reduced set $\Phi(C)$ if a match is found. In this way, the problem of comparing two descriptions that have both already been found to be within the descriptive intersection is avoided and the number of comparisons is reduced.

9 Conclusion

This report presented details on the Proximity System and background on description-based set operations, their application to description-based neighbourhoods, and a metric-free nearness measure. The contribution of the report was a systematic discussion of all functions of the Proximity System. A particularly nice feature of this tool is the ability for users to define there own probe functions. This tool has already proved vital in the study of descriptive-based topological approaches to nearness and proximity within the context of digital image analysis, as can be seen by results reported in [15].

---

[4]Recall, this includes the case where the difference in object descriptions is within some ε.

References

[1] S. A. Naimpally and B. D. Warrack, "Proximity spaces," in *Cambridge Tract in Mathematics No. 59*. Cambridge, UK: Cambridge University Press, 1970.

[2] Z. Pawlak and J. F. Peters, "Jak blisko (how near)," *Systemy Wspomagania Decyzji*, vol. I, pp. 57, 109., 2002.

[3] J. F. Peters, "Near sets. General theory about nearness of objects," *Applied Mathematical Sciences*, vol. 1, no. 53, pp. 2609–2629, 2007.

[4] ——, "Near sets. Special theory about nearness of objects," *Fundamenta Informaticae*, vol. 75, no. 1-4, pp. 407–433, 2007.

[5] S. A. Naimpally, "Near and far. A centennial tribute to Frigyes Riesz," *Siberian Electronic Mathematical Reports*, vol. 6, pp. A.1–A.10, 2009.

[6] ——, *Proximity Approach to Problems in Topology and Analysis*. München: Oldenburg Verlag, 2009, iSBN 978-3-486-58917-7.

[7] J. F. Peters, "Corrigenda and addenda: Tolerance near sets and image correspondence," *International Journal of Bio-Inspired Computation*, vol. 2, no. 5, pp. 310–318, 2010.

[8] ——, "Tolerance near sets and image correspondence," *International Journal of Bio-Inspired Computation*, vol. 1, no. 4, pp. 239–245, 2009.

[9] J. F. Peters and S. A. Naimpally, "Applications of near sets," *Notices of the American Mathematical Society*, vol. 59, no. 4, pp. 536–542, 2012.

[10] S. A. Naimpally and J. F. Peters, *Topology with Applications.Topological Spaces via Near and Far*. Singapore: World Scientific, 2013, *In Press*.

[11] C. Henry, "Near set Evaluation And Recognition (NEAR) system," in *Rough Fuzzy Analysis Foundations and Applications*, S. K. Pal and J. F. Peters, Eds. CRC Press, Taylor & Francis Group, 2010, pp. 7–1 – 7–22, exe. availabe at: http://wren.ee.umanitoba.ca.

[12] C. J. Henry, "Near Sets: Theory and Applications," Ph.D. dissertation, 2010, Available at: https://mspace.lib.umanitoba.ca/handle/1993/4267.

[13] ——, "Perceptually indiscernibility, rough sets, descriptively near sets, and image analysis," *Transactions on Rough Sets*, vol. LNCS 7255, pp. 41–121, 2012.

[14] C. J. Henry and S. Ramanna, "Maximal clique enumeration in finding near neighbourhoods," *Transactions on Rough Sets*, p. 25 pp., 2012, accepted.

[15] C. J. Henry, "Metric free nearness measure using description-based neighbourhoods," *Mathematics in Computer Science*, p. 27 pp., 2012, submitted.

[16] A. Di Concilio, "Proximity: A powerful tool in extension theory, function spaces, hyperspaces, boolean algebras and point-free geometry." in *Beyond Topology*, F. Mynard and E. Pearl, Eds. American Mathematical Society, 2009.

[17] A. B. Sossinsky, "Tolerance space theory and some applications," *Acta Applicandae Mathematicae: An International Survey Journal on Applying Mathematics and Mathematical Applications*, vol. 5, no. 2, pp. 137–167, 1986.

[18] H. Poincaré, *Science and Hypothesis*.   Brock University: The Mead Project, 1905, L. G. Ward's translation.

[19] L. T. Benjamin, Jr., *A Brief History of Modern Psychology*.   Malden, MA: Blackwell Publishing, 2007.

[20] B. R. Hergenhahn, *An Introduction to the History of Psychology*.   Belmont, CA: Wadsworth Publishing, 2009.

[21] E. C. Zeeman, "The topology of the brain and the visual perception," in *Topoloy of 3-manifolds and selected topices*, K. M. Fort, Ed.   New Jersey: Prentice Hall, 1965, pp. 240–256.

[22] M. Pavel, *Fundamentals of Pattern Recognition*.   NY: Marcel Dekker, Inc., 1993.

[23] J. F. Peters, "Classification of objects by means of features," in *Proceedings of the IEEE Symposium Series on Foundations of Computational Intelligence (IEEE SCCI 2007)*, 2007, pp. 1–8.

[24] ——, "Classification of perceptual objects by means of features," *International Journal of Information Technology & Intelligent Computing*, vol. 3, no. 2, pp. 1 – 35, 2008.

[25] J. F. Peters and P. Wasilewski, "Foundations of near sets," *Info. Sci.*, vol. 179, no. 18, pp. 3091–3109, 2009.

[26] R. Duda, P. Hart, and D. Stork, *Pattern Classification*, 2nd ed.   Wiley, 2001.

[27] S. K. Pal and P. Mitra, "Multispectral image segmentation using rough set initialized em algorithm," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 11, p. 24952501, 2002.

[28] J. F. Peters and M. Borkowski, "k-means indiscernibility over pixels," pp. 580–585, 2004.

[29] S. K. Pal, B. U. Shankar, and P. Mitra, "Granular computing, rough entropy and object extraction," *Pattern Recognition Letters*, vol. 26, no. 16, pp. 401–416, 2005.

[30] M. Borkowski and J. F. Peters, "Matching 2d image segments with genetic algorithms and approximations spaces," *Transactions on Rough Sets*, vol. LNCS 4100, 2006.

[31] M. Borkowski, "2d to 3d conversion with direct geometrical search and approximation spaces," Ph.D. dissertation, 2007.

[32] P. Maji and S. K. Pal, "Maximum class separability for rough-fuzzy c-means based brain mr image segmentation," *Transactions on Rough Sets*, vol. IX, LNCS-5390, pp. 114–134, 2008.

[33] M. Mushrif and A. K. Ray, "Color image segmentation: Rough-set theoretic approach," *Pattern Recognition Letters*, vol. 29, no. 4, pp. 483–493, 2008.

[34] A. E. Hassanien, A. Abraham, J. F. Peters, G. Schaefer, and C. Henry, "Rough sets and near sets in medical imaging: A review," *IEEE Transactions on Information Technology in Biomedicine*, vol. 13, no. 6, pp. 955–968, 2009.

[35] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain, "Content-based image retrieval at the end of the early years," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 12, pp. 1349–1380, 2000.

[36] C. J. Henry, S. Ramanna, and D. Levy, "Quantifying nearness in visual spaces," *Cybernetics and Systems Journal*, 2012, under review.

[37] C. J. Henry and S. Ramanna, "Maximal clique enumeration in finding near neighbourhoods," *Transactions on Rough Sets*, 2012, under review.

[38] ——, "Signature-based perceptual nearness. application of near sets to image retrieval," *Mathematics in Computer Science*, 2012, under review.

[39] J. F. Peters and P. Wasilewski, "Tolerance spaces: Origins, theoretical aspects and applications," *Information Sciences*, vol. 195, no. 0, pp. 211–225, 2012.