Near set Evaluation And Recognition (NEAR) System V2.0

Christopher Henry
James F. Peters, Supervisor
{*chenry,jfpeters*}*@ee.umanitoba.ca*
University of Manitoba
Computational Intelligence Laboratory
ENGR E1-526 Engineering & Information Technology Complex
75A Chancellor's Circle
Winnipeg, Manitoba R3T 5V6
Tel.: 204.474.9603
Fax.: 204.261.4639

UM CI Laboratory Technical Report
Number TR-2010-017
August 12, 2010

University of Manitoba

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

Project no. 2010-017

Near set Evaluation And Recognition (NEAR) System V2.0

Christopher Henry
James F. Peters, Supervisor
{chenry,jfpeters}@ee.umanitoba.ca
University of Manitoba
Computational Intelligence Laboratory
ENGR E1-526 Engineering & Information Technology Complex
75A Chancellor's Circle
Winnipeg, Manitoba R3T 5V6
Tel.: 204.474.9603
Fax.: 204.261.4639

CI Laboratory TR-2010-017

August 12, 2010

Abstract

This report presents the Near set Evaluation And Recognition (NEAR) system. The goal of the NEAR system is to extract perceptual information from images using near set theory, which provides a framework for measuring the perceptual nearness of objects. The contributions of this report are an introduction to the NEAR system as an application of near set theory to image processing, a feature-based approach to solving the image correspondence problem, and a first step toward automating the extraction of perceptual information from images where there is interest in measuring the degree of resemblance between images.

1 Introduction

The goal of the NEAR system is to demonstrate applications of the near set theory presented in [1–10] (see also, [11]). The system implements a Multiple Document Interface (MDI) (see, *e*.g., Fig. 1) where each separate processing task is performed in its own child frame. The objects (in the near set sense) in this system are subimages of the images being processed and the probe functions (features) are image processing functions defined on the subimages. The system was written in C++ and was designed to facilitate the addition of new processing tasks and probe functions[1]. Currently, the system performs six major tasks, namely, displaying equivalence and tolerance classes for an image, performing segmentation evaluation, measuring the nearness of two images, performing Content Based Image Retrieval (CBIR), and displaying the output of processing an image using a specific probe function. This report is organized as follows: Section 2 gives some background on near set theory, Section 3 presents the nearness measures implmented in the NEAR system, Section 4 demonstrates the application of near set theory to images, Section 5 presents

This research work has been funded by Manitoba Hydro grants T137, T247, T260, T270, T277, and by the Natural Sciences & Engineering Research Council of Canada (NSERC) grant 185986, NSERC Postgraduate Doctoral Fellowship PGS-D3, University of Manitoba Faculty of Engineering grant, and Canadian Arthritis Network grant SRI-BIO-05.

[1]Parts of the Graphical User Interface (GUI) were inspired by the GUI reported in [12] and the wxWidgets example in [13].

the probe functions implemented in the NEAR system, and Section 6 gives an explanation of the settings used to perform approximate nearest neighbour searching. Finally, Sections 7-10 describe the operation of the GUI.

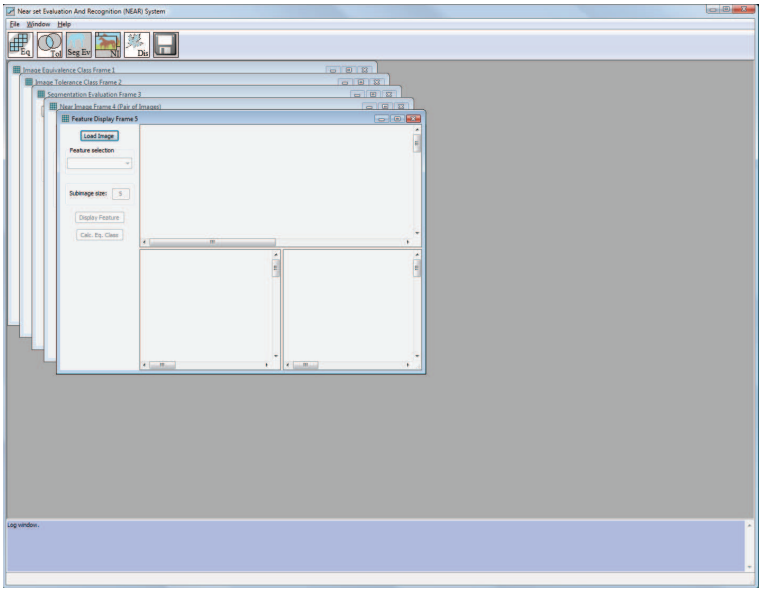


Figure 1: NEAR system GUI.

2 Near sets

Near set theory focuses on sets of perceptual objects with matching descriptions (see, *e.g.* [7, 11]). Specifically, let O represent a set of perceptual objects, *i*.e. O consists of objects that have their origin in the physical world. Each perceptual object can be described by a probe functions, a real-valued function representing a feature of a perceptual object. The description of an object $x \in O$ is given by

$$\boldsymbol{\phi}(x) = (\phi_1(x), \phi_2(x), \ldots, \phi_i(x), \ldots, \phi_l(x)),$$

where l is the length of the description and each $\phi_i(x)$ is a probe function that describes the object x. The notion of a probe function in near sets is inspired by Monique Pavel [14], where a probe function that is invariant relative to a transformation of the images results in matching function (feature) values. In a near set approach, a real-valued funcion $\phi : O \rightarrow \Re$, O a set of images, is a *probe function* if, and only if ϕ represents an image feature with values that are in the description of a perceptual object, in particular, in the description of an image [15–17]. Furthermore, a set $\mathbb{F}$ can be defined that represents all the probe functions used to describe an object x. Next, a perceptual system $\langle O, \mathbb{F} \rangle$ consists of a non-empty set O of sample perceptual objects and a non-empty set $\mathcal{F}$ of real-valued functions $\phi \in \mathcal{F}$ such that $\phi : O \rightarrow \mathbb{R}$.

Definition 1 Perceptual Indiscernibility Relation [7, 18]. *Let $\langle O, \mathbb{F} \rangle$ be a perceptual system. For every $\mathcal{B} \subseteq \mathbb{F}$ the* perceptual indiscernibility relation $\sim_{\mathcal{B}}$ *is defined as follows:*

$$\sim_{\mathcal{B}} = \{(x, y) \in O \times O : \| \boldsymbol{\phi}(x) - \boldsymbol{\phi}(y) \|_2 = 0\},$$

where $\| \cdot \|_2$ represents the L_2 norm.

The perceptual indiscernibility relation is a variation of the one given by Z. Pawlak in 1981 [18]. Furthermore, notice that equivalence is defined with respect to the description of an object, *i.e.* objects are considered equivalent when the features used to describe them are the same.

Using the indiscernibility relation (together with the probe functions in $\mathcal{B}$), a set of objects can be partitioned into classes of objects with matching descriptions such that each class has the highest possible object resolution under the indiscernibility relation. These classes are called elementary sets or equivalence classes and are defined as

$$x_{/\sim_{\mathcal{B}}} = \{x' \in O \mid x' \sim_{\mathcal{B}} x\}.$$

Observe that a single object is sufficient to label the class since all objects in a class have the same description. Moreover, the set of all equivalence classes induced by the partition of a set using the indiscernibility relation is called a quotient set and is given as

$$O_{/\sim_{\mathcal{B}}} = \{x_{/\sim_{\mathcal{B}}} \mid x \in O\}.$$

Definition 1 provides the framework for comparisons of sets of objects by introducing a concept of nearness within a perceptual system. Sets can be considered near each other when they have "things" in common. In the context of near sets, the "things" can be quantified by objects or equivalence classes. The simplest example of nearness between sets sharing "things" in common is the case when two sets have indiscernible elements. This idea leads to the definition of a weak nearness relation.

Definition 2 Weak Nearness Relation [7]. *Let $\langle O, \mathbb{F}\rangle$ be a perceptual system and let $X, Y \subseteq O$. A set X is weakly near to a set Y within the perceptual system $\langle O, \mathbb{F}\rangle$ ($X \bowtie_{\mathbb{F}} Y$) iff there are $x \in X$ and $y \in Y$ and there is $\mathcal{B} \subseteq \mathbb{F}$ such that $x \sim_{\mathcal{B}} y$. In the case where sets X, Y are defined within the context of a perceptual system, then X, Y are weakly near each other.*

An example of disjoint sets that are weakly near each other is given in Fig. 2(a) where each colour represents an equivalence class. These sets are weakly near each other since both sets share objects belonging to the same equivalence class. As a practical example of weakly near sets, consider a database of images where each image is described by some feature vector, *i.e.* the images are considered perceptual objects and the feature vectors are the object descriptions. Examples of features are the values of different colour models [19] or moments [20]. In this case, two disjoint sets of images are weakly near each other if each set contains one or more images with descriptions that match an image in the other set.

Next, the notion of nearness in Definition 2 can be strengthened by considering equivalence classes rather than objects which is the case in the following definition.

Definition 3 Nearness Relation [7]. *Let $\langle O, \mathbb{F}\rangle$ be a perceptual system and let $X, Y \subseteq O$. A set X is near to a set Y within the perceptual system $\langle O, \mathbb{F}\rangle (X \bowtie_{\mathbb{F}} Y)$ iff there are $\mathbb{F}_1, \mathbb{F}_2, \subseteq \mathbb{F}$ and $f \in \mathbb{F}$ and there are $A \in O_{/\sim_{\mathbb{F}_1}}, B \in O_{/\sim_{\mathbb{F}_2}}, C \in O_{/\sim_f}$ such that $A \subseteq X, B \subseteq Y$, and $A, B \subseteq C$. If a perceptual system is understood, than a set X is near to a set Y.*

2.1 Tolerance near sets

A perception-based approach to discovering resemblances between images leads to a tolerance class form of near sets that models human perception in a physical continuum viewed in the context of image tolerance spaces. A tolerance space-based approach to perceiving image resemblances hearkens back to the observation about perception made by Ewa Orłowska in 1982 [21] (see, also, [22]), *i.e.*, classes defined in an approximation space serve as a formal counterpart of perception.

The term *tolerance space* was coined by E.C. Zeeman in 1961 in modeling visual perception with tolerances [23]. A tolerance space is a set X supplied with a binary relation $\simeq$ (*i.e.*, a subset $\simeq \subset X \times X$)

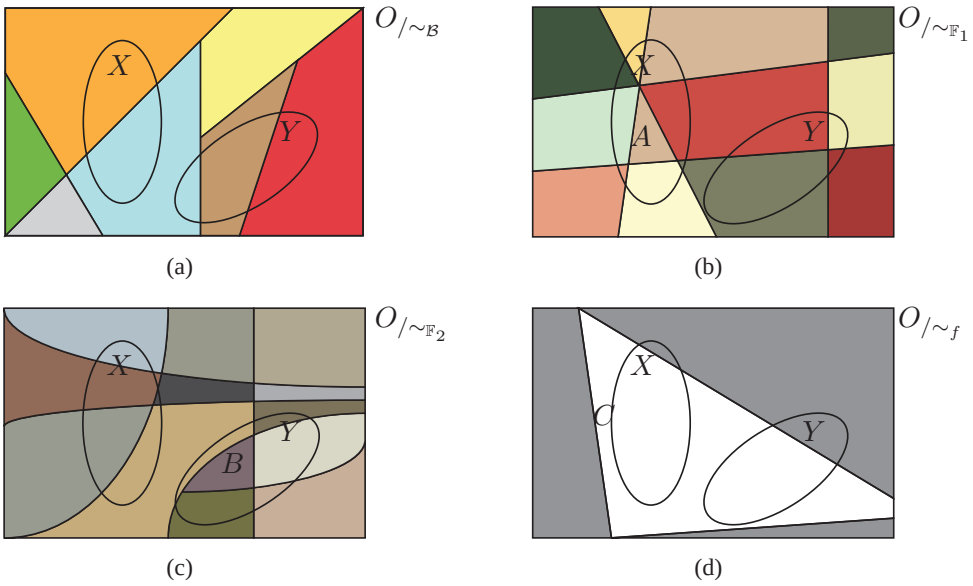


Figure 2: Examples of Definitions 2 & 3: (a) Example of Definition 2, (b) example of $O_{/\sim_{\mathbb{F}_1}}$, (c) example of $O_{/\sim_{\mathbb{F}_2}}$, and (d) example of $O_{/\sim_f}$ showing (together with (b) and (c)) that sets X and Y are near to each other according to Definition 3.

that is reflexive (for all $x \in X$, $x \simeq x$) and symmetric (*i.e.*, for all $x, y \in X$, $x \simeq y$ implies $y \simeq x$) but transitivity of $\simeq$ is not required. When dealing with perceptual objects (especially, components in images), it is sometimes necessary to relax the equivalence condition of Defn. 1 to facilitate observation of associations in a perceptual system. This variation is called a tolerance relation that defines yet another form of near sets [3, 4, 8, 9] and is given in Defn. 4.

Definition 4 Perceptual Tolerance Relation [8,9] (see [10,24] for applications). *Let $\langle O, \mathbb{F} \rangle$ be a perceptual system and let $\varepsilon \in \mathbb{R}$. For every $\mathcal{B} \subseteq \mathbb{F}$ the* perceptual tolerance relation $\cong_{\mathcal{B},\varepsilon}$ *is defined as follows:*

$$\cong_{\mathcal{B},\varepsilon} = \{(x, y) \in O \times O : \| \phi(x) - \phi(y) \|_2 \leq \varepsilon\}.$$

For notational convenience, this relation is written $\cong_{\mathcal{B}}$ instead of $\cong_{\mathcal{B},\varepsilon}$ with the understanding that ε is inherent to the definition of the tolerance relation.

Under the tolerance relation there is a need to define a method by which objects are grouped together when transitivity no longer applies. In an equivalence class, an object is added to a class if its description matches the description of the objects already in the class, which by definition are all the same. However, the lack of transitivity gives rise to the two very different classes, namely a neighbourhood and a pre-class. A *neighbourhood* is defined as

$$N(x) = \{y \in O : x \cong_{\mathcal{B},\varepsilon} y\}.$$

In contrast, all the pairs of objects within a pre-class must satisfy the tolerance relation. For $\mathcal{B} \subseteq \mathbb{F}$ and $\varepsilon \in \mathbb{R}$, a set $X \subseteq O$ is a *pre-class* iff $x \cong_{\mathcal{B},\varepsilon} y$ for any pair $x, y \in X$, and a maximal pre-class with respect to inclusion is called a tolerance class.

Notice, objects can belong to more than one tolerance class. Consequently, the following notation is required to differentiate between classes and facilitate discussions in subsequent sections. The set of all

Table 1: Tolerance Class Example

x_i	$\phi(x)$	x_i	$\phi(x)$	x_i	$\phi(x)$	x_i	$\phi(x)$
x_1	.4518	x_6	.6943	x_{11}	.4002	x_{16}	.6079
x_2	.9166	x_7	.9246	x_{12}	.1910	x_{17}	.1869
x_3	.1398	x_8	.3537	x_{13}	.7476	x_{18}	.8489
x_4	.7972	x_9	.4722	x_{14}	.4990	x_{19}	.9170
x_5	.6281	x_{10}	.4523	x_{15}	.6289	x_{20}	.7143

tolerance classes using only the objects in O is given by $H_{\cong_{\mathcal{B},\varepsilon}}(O)$ (also called the cover of O), a single tolerance class is represented by $C \in H_{\cong_{\mathcal{B},\varepsilon}}(O)$, and the set of all tolerance classes containing an object x is denoted by $C_x \subset H_{\cong_{\mathcal{B},\varepsilon}}(O)$.

The following simple example highlights the need for a tolerance relation as well as demonstrates the construction of tolerance classes from real data. Consider the 20 objects in Table 1 that where $|\phi(x_i)| = 1$. Letting $\varepsilon = 0.1$ gives the following tolerance classes:

$$\begin{aligned} H_{\cong_{\mathcal{B},\varepsilon}}(O) \quad = \quad & \{\{x_1, x_8, x_{10}, x_{11}\}, \{x_1, x_9, x_{10}, x_{11}, x_{14}\}, \\ & \{x_2, x_7, x_{18}, x_{19}\}, \\ & \{x_3, x_{12}, x_{17}\}, \\ & \{x_4, x_{13}, x_{20}\}, \{x_4, x_{18}\}, \\ & \{x_5, x_6, x_{15}, x_{16}\}, \{x_5, x_6, x_{15}, x_{20}\}, \\ & \{x_6, x_{13}, x_{20}\}\} \end{aligned}$$

Observe that each object in a tolerance class satisfies the condition $\| \boldsymbol{\phi}(x) - \boldsymbol{\phi}(y) \| \leq \varepsilon$, and that almost all of the objects appear in more than one class. Moreover, there would be twenty classes if the indiscernibility relation was used since there are no two objects with matching descriptions.

Definition 5 Tolerance Near Sets [8, 9]. *Let $\langle O, \mathbb{F} \rangle$ be a perceptual system and let $\varepsilon \in \mathbb{R}, \mathcal{B} \subseteq \mathbb{F}$. Further, let $X, Y \subseteq O$, denote disjoint sets with coverings determined by the tolerance relation $\cong_{\mathcal{B},\varepsilon}$, and let $H_{\cong_{\mathcal{B},\varepsilon}}(X), H_{\cong_{\mathcal{B},\varepsilon}}(Y)$ denote the set of tolerance classes for X, Y, respectively. Sets X, Y are* tolerance near sets *iff there are tolerance classes $A \in H_{\cong_{\mathcal{B},\varepsilon}}(X), B \in H_{\cong_{\mathcal{B},\varepsilon}}(Y)$ such that $A \underset{\cong_{\mathbb{F}}}{\bowtie} B$.*

3 Nearness measure

The nearness measure was first proposed in working with the indiscernibility relation and equivalence classes [11]. The approach was that the degree of nearness of sets in a perceptual system is determined by the cardinalities of the equivalence classes that have the same description (an idea that is visualized in Fig. 3). For example, sets that are considered "more similar" as in Fig. 4(a), should contain more pairs of equivalence classes (from the respective sets) that have matching descriptions. Consequently, the nearness measure is determined by counting the number of objects in equivalence classes that have matching descriptions. Thus, the sets in Fig. 4(a) are closer (more near) to each other in terms of their descriptions than the sets in Fig. 4(b). Moreover, this notion can be generalized to tolerance classes as is the case in the following definition.

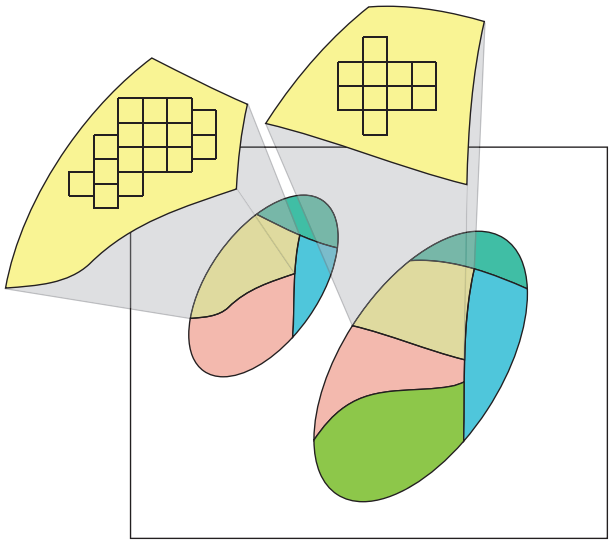
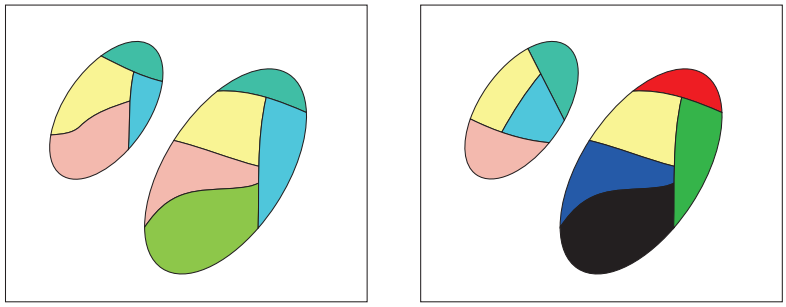


Figure 3: Visualization of nearness measure based on equivalence classes and the indiscernibility relation. Similar images should produce equivalence classes that are evenly divided between X and Y. This is measured by counting the number of objects that belong to sets X and Y for each equivalence class, and comparing the results.



(a) (b)

Figure 4: Example of degree of nearness between two sets: (a) High degree of nearness, and (b) low degree of nearness.

Definition 6 Nearness Measure [5, 10, 11]. *Let $\langle O, \mathbb{F} \rangle$ be a perceptual system, with $\varepsilon \in \mathbb{R}$, and $\mathcal{B} \subseteq \mathbb{F}$. Furthermore, let X and Y be two disjoint sets and let $Z = X \cup Y$. Then a* nearness measure *between two sets is given by*

$$tNM_{\cong_{\mathcal{B},\varepsilon}}(X,Y) = \left(\sum_{C \in H_{\cong_{\mathcal{B},\varepsilon}}(Z)} |C| \right)^{-1} \cdot \sum_{C \in H_{\cong_{\mathcal{B},\varepsilon}}(Z)} |C| \frac{\min(|C \cap X|, |[C \cap Y|)}{\max(|C \cap X|, |C \cap Y|)}.$$

3.1 Other measures

This section introduces two additional measures for determining the degree that near sets resemble each other. These measures were created out of a need for making comparisons of the results generated by the nearness measure. Here, one of two approaches could have been investigated. Namely, the nearness measure could be compared with a content-based image retrieval system or measure that is currently regarded as

the best approach for a database with given characteristics. Or, the nearness measure could be compared with measures that determine nearness in a manner comparable to tNM. In the NEAR system, the latter approach was taken. As a result, approaches were created, based on existing theories, that measure the distance between sets.

3.1.1 Hausdorff Distance

The Hausdorff distance is used to measure the distance between sets in a metric space [25] (see [26] for English translation), and is defined as

$$d_H(X,Y) = \max\{ \sup_{x\in X} \inf_{y\in Y} d(x,y), \sup_{y\in Y} \inf_{x\in X} d(x,y) \},$$

where $\sup$ and $\inf$ refer to the supremum and infimum, and $d(x,y)$ is the distance metric (in this case it is the l^2 norm). The distance is calculated by considering the distance from a single element in a set X to every element of set Y, and the shortest distance is selected as the infimum (see, *e.g.*, Fig. 5). This process is repeated for every $x \in X$ and the largest distance (supremum) is selected as the Hausdorff distance of the set X to the set Y. This process is then repeated for the set Y because the two distances will not necessarily be the same. Keeping this in mind, the measure tHD [5] is defined as

$$tHD_{\cong_{\mathcal{B},\varepsilon}}(X,Y) = \left(\sum_{C\in H_{\cong_{\mathcal{B},\varepsilon}}(Z)} |C| \right)^{-1} \cdot \sum_{C\in H_{\cong_{\mathcal{B},\varepsilon}}(Z)} |C|(\sqrt{l} - d_H(C\cap X, C\cap Y)).$$

Observe, that low values of the Hausdorff distance correspond to a higher degree of resemblance than larger distances. Consequently, the distance is subtracted from the largest distance $\sqrt{l}$. Also, notice that the performance of the Hausdorff distance is poor for low values of ε, since, as tolerance classes start to become equivalence classes (*i.e.* as $\varepsilon \to 0$), the Hausdorff distance approaches 0 as well. Thus, if each tolerance class is close to an equivalence class, the resulting distance will be zero, and consequently the measure will produce a value near to 1, even if the images are not alike. In contrast, as ε increases, the members of classes tend to become separated in feature space, and, as a result, only classes with objects that have objects in X that are close to objects in Y will produce a distance close to zero. What does this imply? If for a larger value of ε, relatively speaking, the set of objects $Z = X \cup Y$ still produces tolerance classes with objects that are tightly clustered, then this measure will produce a high measure value. Notice, that this distinction is only made possible if ε is relaxed. Otherwise, all tolerance classes will be tightly clustered.

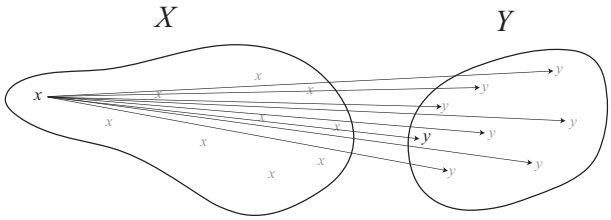


Figure 5: Example demonstrating a single step in determining the Hausdorff distance between two sets.

The Hausdorff distance is a natural choice for comparison with the tNM nearness measure because it measures the distance between sets in a metric space. Recall, that tolerance classes are sets of objects with descriptions in l-dimensional feature space. The nearness measure evaluates the split of a tolerance class between sets X and Y, where the idea is that a tolerance class should be evenly divided between X and Y, if the two sets are similar (or the same). In contrast, the Hausdorff distance measures the distance between

two sets. Here the distance being measured is between the portions of a tolerance class in sets X and Y. Thus, two different measures can be used on the same data, namely the tolerance classes obtained from the union of X and Y.

3.1.2 Hamming Measure

The Hamming measure introduced in this section was inspired by the Hamming measure in [27], and since the Hamming measure is not defined in terms of sets, it was modified to give the following

$$tHM_{\cong_{\mathcal{B}}}(X,Y) = \frac{1}{|H_{\cong_{\mathcal{B}}}(Z)|} \cdot \sum_{C \in H_{\cong_{\mathcal{B}}}(Z)} \mathbf{1}(|\text{avgn}(C \cap X) - \text{avgn}(C \cap Y)| \leq th),$$

where $\mathbf{1}(\cdot)$ is the indicator function and $\text{avgn}(C \cap X)$ is the average feature vector used to describe objects in $C \cap X$. For example, the average feature vector can be calculated by adding all the values for a specific feature in the feature vector in $C \cap X$, and then dividing by the number of objects. The idea behind this measure is that, for similar sets, the average feature vector of the portion of a tolerance class (obtained from $Z = X \cup Y$) that lies in X should have values similar to the average feature vector of the portion of the tolerance class that lies in Y. Observe, that if $th = \varepsilon$, this function will simply count the number of classes that are not singletons, *i.e.* class that contain more than one element, since all objects have descriptions whose distances are less than ε. If $th = \varepsilon$, than this measure will perform best for low levels of ε, since only sets that resemble each other will contain classes with cardinality greater than one. Otherwise, this measure will perform in a similar manner to tHD, namely, this measure will produce high values for classes which have objects in X that are close to objects in Y with respect to th.

4 Perceptual image processing

Near set theory can easily be applied to images by partitioning an image into subimages and considering each subimage as an object in the near set sense, *i.e.* each subimage is a perceptual object, and each object description consists of the values obtained from techniques of image processing on the subimage (see, *e.g.* Fig. 6). Moreover, this technique of partitioning an image, and assigning feature vectors to each subimage is an approach that has also been traditionally used in content-based image retrieval.

Formally, define a RGB image as $f = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_T\}$, where $\mathbf{p}_i = (c, r, R, G, B)^{\mathrm{T}}$, $c \in [1, M]$, $r \in [1, N]$, $R, G, B \in [0, 255]$, and M, N respectively denote the width and height of the image and $M \times N = T$. Further, define a square subimage as $f_i \subset f$ such that $f_1 \cap f_2 \ldots \cap f_s = \emptyset$, and $f_1 \cup f_2 \ldots \cup f_s = f$, where s is the number of subimages in f. Next, O can be defined as the set of all subimages, *i.e.*, $O = \{f_1, \ldots, f_s\}$, and $\mathbb{F}$ is a set of image processing descriptors or functions that operate on images. Then, the nearness of two images can be discovered by partitioning each of the images into subimages and letting these represent objects in a perceptual system, *i.e*, let the sets X and Y represent the two images to be compared where each set consists of the subimages obtained by partitioning the images. Then, the set of all objects in this perceptual system is given by $Z = X \cup Y$.

5 Probe functions

This section describes the probe functions used in the NEAR system, and gives example NEAR system output images processed using these probe functions.

Figure 6: Example demonstrating the application of near set theory to images, namely the image is partitioned into subimages where each subimage is considered a perceptual object, and object descriptions are the results of image processing techniques on the subimage.

5.1 Average greyscale value

Conversion from RGB image to greyscale is accomplished using Magick++, the object-orientated C++ API to the ImageMagick image-processing library [28]. First, an RGB image is converted to greyscale using

$$Gr = 0.299R + 0.587G + 0.114B, \tag{1}$$

and then the values are averaged over each subimage. An example is given in Fig. 7.

(a) (b) (c)

Figure 7: Example of average greyscale probe function: (a) Original image [29], (b) average greyscale over subimages of size 5×5, and (c) average greyscale over subimages of size 10×10.

5.2 Normalized RGB

The normalized RGB values is a feature described in [30], and the formula is given by

$$N_X = \frac{X}{R_T + G_T + B_T},$$

where the values R_T, G_T, and B_T are respectively the sum of R, G, B components of the pixels in each subimage, and $X \in [R_T, G_T, B_T]$. See Fig. 8 for an example using this probe function. Note, these images were produces by finding the normalized value and multiplying it by 255.

(a) (b) (c)

Figure 8: Example of normalized RGB probe function: (a) Original image [31], (b) normalized R over subimages of size 5×5, and (c) normalized R over subimages of size 10×10.

5.3 Shannon's entropy

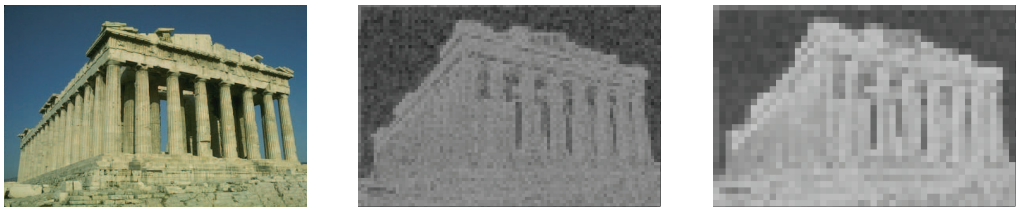
Shannon introduced entropy (also called information content) as a measure of the amount of information gained by receiving a message from a finite codebook of messages [32]. The idea was that the gain of information from a single message is proportional to the probability of receiving the message. Thus, receiving a message that is highly unlikely gives more information about the system than a message with a high probability of transmission. Formally, let the probability of receiving a message i of n messages be p_i, then the information gain of a message can be written as

$$\Delta I = \log(1/p_i) = -\log(p_i), \tag{2}$$

and the entropy of the system is the expected value of the gain and is calculated as

$$H = -\sum_{i=1}^{n} pi \log(p_i).$$

This concept can easily be applied to the pixels of a subimage. First, the subimage is converted to greyscale using Eq. 1. Then, the probability of the occurrence of grey level i can be defined as $p_i = h_i/T_s$, where h_i is the number of pixels that take a specific grey level in the subimage, and T_s is the total number of pixels in the subimage. Information content provides a measure of the variability of the pixel intensity levels within the image and takes on values in the interval $[0, \log_2 L]$ where L is the number of grey levels in the image. A value of 0 is produced when an image contains all the same intensity levels and the highest value occurs when each intensity level occurs with equal frequency [33]. An example of this probe function is given in Fig. 9. Note, these images were formed by multiplying the value of Shannon's entorpy by 32 since $L = 256$ (thus giving a maximum value of 8).



(a) (b) (c)

Figure 9: Example of Shannon's entropy applied to images: (a) Original image [31], (b) Shannon's entropy applied to subimages of size 5×5, and (c) Shannon's entropy applied to subimages of size 10×10.

5.4 Pal's entropy

Work in [32, 34] shows that Shannon's definition of entropy has some limitations. Shannon's definition of entropy suffers from the following problems: it is undefined when $p_i = 0$; in practise the information gain tends to lie at the limits of the interval $[0, 1]$; and statistically speaking, a better measure of ignorance is 1 - p_i rather than $1/p_1$ [32]. As a result, a new definition of entropy can be defined with the following desirable properties:

P1: $\Delta I(p_i)$ is defined at all points in $[0, 1]$.

P2: $\lim_{p_i \to 0} \Delta I(p_i) = \Delta I(p_i = 0) = k_1, k_1 > 0$ and finite.

P3: $\lim_{p_i \to 1} \Delta I(p_i) = \Delta I(p_i = 1) = k_2, k_2 > 0$ and finite.

P4: $k_2 < k_1$.

P5: With increase in p_i, $\Delta I(p_i)$ decreases exponentially.

P6: $\Delta I(p)$ and H, the entropy, are continuous for $0 \leq p \leq 1$.

P7: H is maximum when all p_i's are equal, *i.e.* $H(p_1, \ldots, p_n) \leq H(1/n, \ldots, 1/n)$.

With these in mind, [32] defines the gain in information from an event as

$$\Delta I(p_i) = e^{(1-p_i)},$$

which gives a new measure of entropy as

$$H = \sum_{i=1}^{n} p_i e^{(1-p_i)}.$$

Pal's version of entropy is given in Fig. 10. Note, these images were formed by first converting the original image to greyscale, calculating the entropy for each subimage, and multiplying this value by 94 (since the maximum of H is $e^{1-1/256}$).

(a) (b) (c)

Figure 10: Example of Pal's entropy applied to images: (a) Original image [31], (b) Pal's entropy applied to subimages of size 5×5, and (c) Pal's entropy applied to subimages of size 10×10.

5.5 Edge based probe functions

The edge based probe functions integrated in the NEAR system incorporate an implementation of Mallat's Multiscale edge detection method based on Wavelet theory [35]. The idea is that edges in an image occur at points of sharp variation in pixel intensity. Mallat's method calculates the gradient of a smoothed image

using Wavelets, and defines edge pixels as those that have locally maximal gradient magnitudes in the direction of the gradient.

Formally, define a 2-D smoothing function $\theta(x, y)$ such that its integral over x and y is equal to 1, and converges to 0 at infinity. Using the smoothing function, one can define the functions

$$\psi^1(x, y) = \frac{\partial\theta(x, y)}{\partial x} \quad \text{and} \quad \psi^2(x, y) = \frac{\partial\theta(x, y)}{\partial y},$$

which are, in fact, wavelets given the properties of $\theta(x, y)$ mentioned above. Next, the dilation of a function by a scaling factor s is defined as

$$\xi_s(x, y) = \frac{1}{s^2}\xi(\frac{x}{s}, \frac{y}{s}).$$

Thus, the dilation by s of ψ^1, and ψ^2 is given by

$$\psi^1_s(x, y) = \frac{1}{s^2}\psi^1(x, y) \quad \text{and} \quad \psi^2_s(x, y) = \frac{1}{s^2}\psi^2(x, y).$$

Using these definitions, the wavelet transform of $f(x, y) \in L^2(\mathbb{R}^2)$ at the scale s is given by

$$W^1_s f(x, y) = f * \psi^1_s(x, y) \quad \text{and} \quad W^2_s f(x, y) = f * \psi^2_s(x, y),$$

which can also be written as

$$\begin{pmatrix} W^1_s f(x, y) \\ W^2_s f(x, y) \end{pmatrix} = s \begin{pmatrix} \frac{\partial}{\partial x}(f * \theta_s)(x, y) \\ \frac{\partial}{\partial y}(f * \theta_s)(x, y) \end{pmatrix} = s\vec{\nabla}(f * \theta_s)(x, y).$$

Finally, edges can be detected by calculating the modulus and angle of the gradient vector defined respectively as

$$M_s f(x, y) = \sqrt{|W^1_s f(x, y)|^2 + |W^2_s f(x, y)|^2}$$

and

$$A_s f(x, y) = \text{argument}(W^1_s f(x, y) + iW^2_s f(x, y)),$$

and then finding the modulus maximum defined as pixels with modulus greater than the two neighbours in the direction indicated by $A_s f(x, y)$ (see [35] for specific implementation details). Examples of Mallatt's edge detection method obtained using the NEAR system are given in Fig. 11.

5.5.1 Edge present

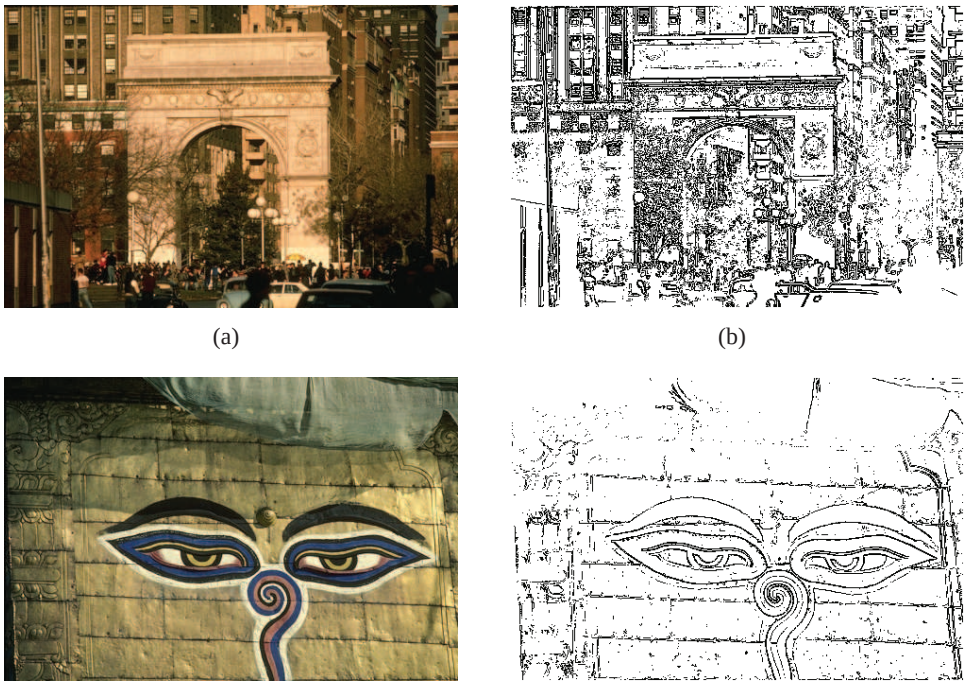
This prob function simply returns true if there is an edge pixel contained in the subimage (see, *e.g.*, Fig. 12).

5.5.2 Number of edge pixels

This probe function returns the total number of pixels in a subimage belonging to an edge (see, *e.g.*, Fig. 13).

5.5.3 Edge orientation

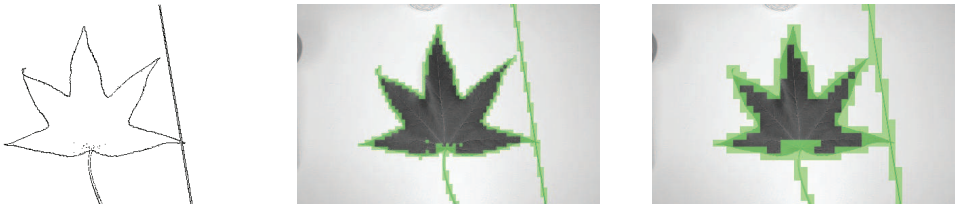
This probe function returns the average orientation of subimage pixels belonging to an edge (see, *e.g.*, Fig. 14).



(a) (b)

(c) (d)

Figure 11: Example of NEAR system edge detection using Mallat's method: (a) Original image, (b) edges obtained from (a), (c) original image, and (d) obtained from (c).



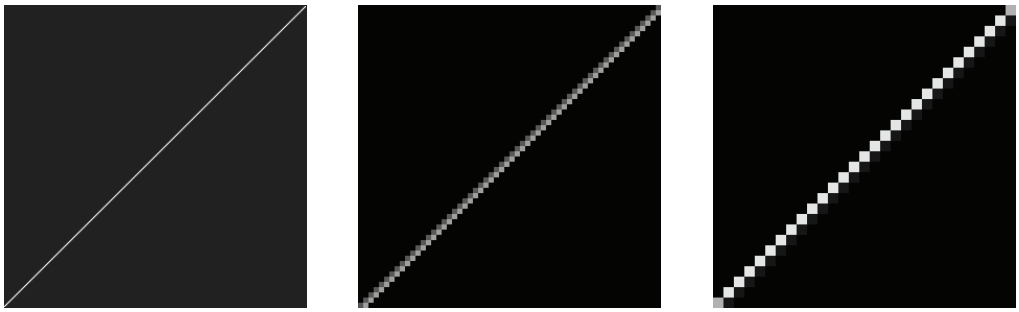
(a) (b) (c)

Figure 12: Example of edge present probe function: (a) Edges obtained from Fig. 7(a), (b) Application to image with subimages of size 5×5, and (c) Application to image with subimages of size 10×10.

5.6 Grey level co-occurrence matrices

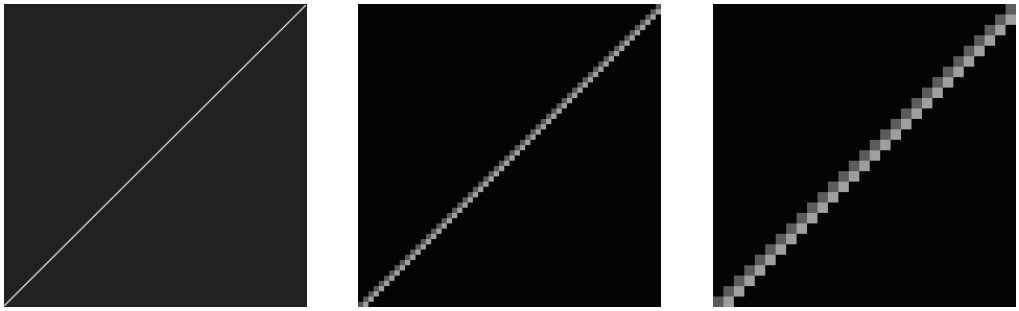
Image texture is an important part of perceiving images. Texture is difficult to describe, and is generally associated with a region of the image, rather than restricted to a specific pixel. Generally, there are statistical and structural approaches to identifying texture [36]. The textural features used in this thesis are based on second order measures, as reported in [37–39], where the approach is considered second-order, since the measures are not derived directly from the pixel values themselves, but rather on statistics generated from relationships between groups of two pixels given by a grey-level co-occurrence matrix. In other words, the features are based on the average spatial relationship between pixel values [37].

In general, the grey level co-occurrence matrix is defined with respect to the angle and distance between pixel pairs. However, to keep things simple, the grey level co-occurrence matrix will first be defined with respect to horizontally adjacent pixels, which corresponds to an angle of 0° and a distance $d = 1$ in the



(a) (b) (c)

Figure 13: Example of number of edge pixels probe function: (a) Original image, (b) Application to image with subimages of size 5×5, and (c) Application to image with subimages of size 10×10.



(a) (b) (c)

Figure 14: Example of average orientation probe function: (a) Original image, (b) Application to image with subimages of size 5×5, and (c) Application to image with subimages of size 10×10.

traditional literature. Using the notation given in [37], let $L_x = \{1, 2, \ldots, N_x\}$ and $L_y = \{1, 2, \ldots, N_y\}$ respectively denote the horizontal and vertical spatial domains of a grey level image quantized to N_g levels, *i.e.* the grey levels in an image are in the set $G = \{0, 1, \ldots, N_g - 1\}$. Then, $L_y \times L_x$ is the set of all pixel coordinates belonging to an image I, where $I : L_y \times L_x \rightarrow G$, and the grey level co-occurrence matrix is given as

$$P(i,j) = |\{((k,l),(m,n)) \in (L_y \times L_x) \times (L_y \times L_x) : m - k = 0, n - l = 1, I(k,l) = i, I(m,n) = j\}|. \quad (3)$$

For clarity, an example of Eq. 3 is given graphically in Fig. 15. One can add the degree and distance to Eq. 3, by the following simple modification,

$$P(i,j,d,0^\circ) = |\{((k,l),(m,n)) \in (L_y \times L_x) \times (L_y \times L_x) : m - k = 0, n - l = 1, I(k,l) = i, I(m,n) = j\}|.$$

For angles $45^\circ, 90^\circ$, and 135°, see [37]. Finally, the following textural features can be derived from the grey

level co-occurrence matrix,

Maximum Probability	$\max_{i,j}(p_{ij})$,
Contrast	$\sum_{i=0}^{N_g-1}\sum_{j=0}^{N_g-1}(i-j)^2 p_{ij}$,
Uniformity (also called Energy)	$\sum_{i=0}^{N_g-1}\sum_{j=0}^{N_g-1} p_{ij}^2$, and
Homogeneity	$\sum_{i=0}^{N_g-1}\sum_{j=0}^{N_g-1}\frac{p_{ij}}{1+\lvert i-j\rvert}$,

where $p_{ij} = P(i,j)$ divided by the sum of the elements in P. In brief, the maximum probability returns the strongest response of P, contrast measures the intensity contrast between a pixel and its neighbour, uniformity is the angular second moment, and homogeneity measures the spatial closeness of the distribution of elements in P to the diagonal (see [40] for further details). Finally, Fig. 16 gives examples of each of the grey level co-occurance matrix probe functions implemented in the NEAR system.

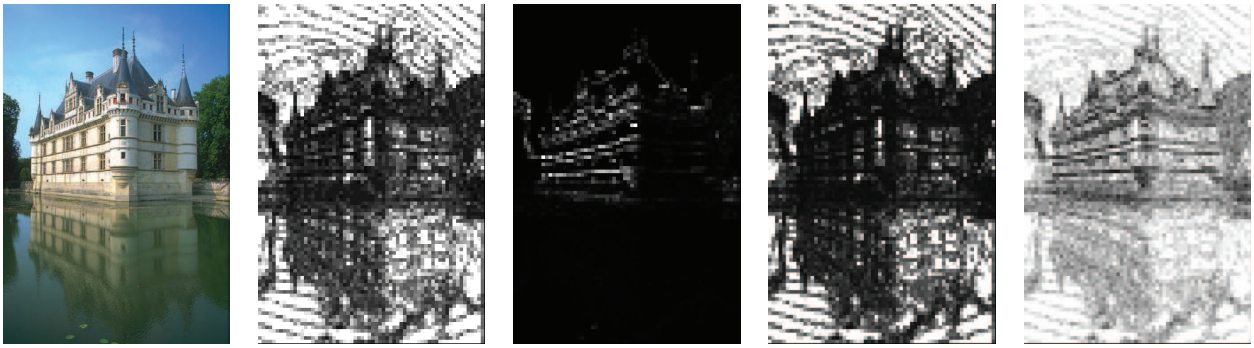
3	3	1	1	2	0	0	0
3	0	3	2	1	2	3	3
2	3	2	0	0	1	0	0
0	1	2	0	1	2	3	3
0	0	3	2	0	2	2	3
1	1	1	3	3	2	3	3
3	2	3	3	1	1	0	0
1	1	3	0	2	0	1	1

(a)

	0	1	2	3
0	6	4	2	2
1	2	6	4	2
2	5	1	1	6
3	2	2	5	6

(b)

Figure 15: Example demonstrating the creation of a grey level co-occurrence matrix. (a) Quantized image, and (b) grey level co-occurrence matrix of 0° and $d = 1$.



(a) (b) (c) (d) (e)

Figure 16: Example of GLCM probe functions using subimages of size 5×5: (a) Original image, (b) maximum probability, (c) contrast, (d) uniformity, and (e) homogeneity.

5.7 Zernike moments

Zernike moments can be used to provide region-based descriptors of an image that are invariant with respect to rotation and reflections [20]. Moreover, a small set of Zernike moments can characterize the global shape of a pattern effectively, where the lower order moments represent the global shape, and the higher order moments represent the detail [41–43].

As given in [20], for a continuous image function $f(x, y)$, the Zernike moment of order n with repetition m is defined as

$$A_{nm} = \int\int_D f(x, y) V^*_{nm}(x, y) dx dy, \tag{4}$$

where the double integral is defined over the unit disk $D = \{(x, y) : x^2 + y^2 \leq 1\}$, n is a non-negative integer, and m is an integer that makes result of $n - |m|$ even and non-negative. In Eq. 4, $V_{nm}(x, y)$ is a Zernike function defined as

$$V_{nm}(x, y) = R_{nm}(\rho) e^{jm\theta},$$

where $\rho = \sqrt{x^2 + y^2}$, $\theta = \tan^{-1}(y/x)$, and the radial Zernike polynomial $R_{nm}(\rho)$ is defined by

$$R_{nm}(\rho) = \sum_{s=0}^{(n-|m|)/2} \frac{(-1)^s (n-s)! p^{n-2s}}{s!(\frac{n+|m|}{2} - s)!(\frac{n-|m|}{2} - s)!}.$$

As explained in [20], Eq. 4 cannot be applied directly to digital images. Consequently, a mapping of the digital image must occur. Let $F(i, j), i = 1, \ldots, N, j = 1, \ldots, N$ denote an $N \times N$ image, then $F(i, j)$ can be mapped onto a function $f(x_i, y_i)$ defined on $[-1, 1]^2$ according to

$$f(x_i, y_i) = F(i, j), i = 1, \ldots, N, j = 1, \ldots, N,$$

where $x_i = (2i - N - 1)/N$ and $y_j = (2j - N - 1)/N$. Note, it can be assumed, without loss of generality, that $f(x_i, y_i)$ is a function with all its pixels inside the unit circle [20]. Moreover, since the image is not analog, but actually a discrete function, the following approximation can be used to calculate the Zernike moments from sampled data

$$\widetilde{A}_{nm} = \sum_i \sum_j w_{nm}(x_i, y_j) f(x_i, y_j), \tag{5}$$

where i and j are taken such that $(x_i, y_j) \in D$,

$$w_{nm}(x_i, y_j) = \int_{x_i - \frac{\Delta}{2}}^{x_i + \frac{\Delta}{2}} \int_{y_j - \frac{\Delta}{2}}^{y_j + \frac{\Delta}{2}} V^*_{nm}(x, y) dx dy,$$

and $\Delta = 2/N$ is the pixel width/height. Finally, $w_{nm}(x_i, y_j)$ can be computed using

$$w_{nm}(x_i, y_j) \approx \Delta^2 V^*_{nm}(x_i, y_j). \tag{6}$$

Note, it was shown in [20] that using Eq. 5 & 6 is a highly inaccurate approach to computing Zernike moments due to both the geometric error caused by the difference between the total area covered by the pixels in Eq. 5 and the actual area of the unit circle, as well as the error due to the approximation of $w_{nm}(x_i, y_j)$ in Eq. 6. Instead, a method for calculating Zernike moments in polar coordinates (rather than the Cartesian method given above) is given that eliminates the previously mentioned errors. Nevertheless, Eq. 5 & 6 were still used to generate rotationally invariant features due to the following reasons. First, only low order moments were used (*e.g.* $n \leq 4$), and evidence in [20] demonstrated that the results of using only low orders of Zernike moments produced magnitudes with acceptable level of errors, both in comparisons of the magnitudes on a constant image and for use in reconstructing images. Also, others have reported success using low order Zernike moments for content-based image retrieval (see, *e.g.* [44, 45]), and implementation of Eq. 5 & 6 is simple and fast. See Fig. 17 for examples of the probe functions based on Zernike moments implemented in the NEAR system.

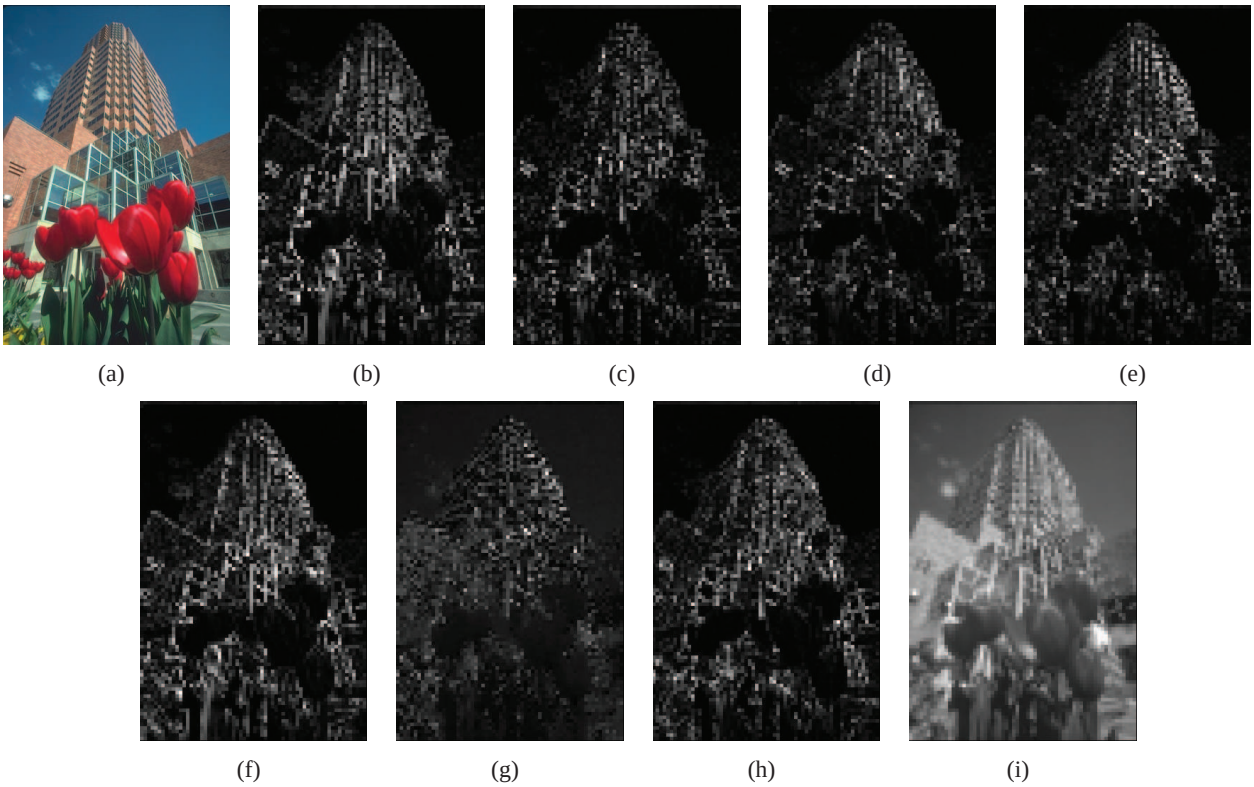


Figure 17: Example of Zernike probe functions using subimages of size 5×5: (a) Original image, (b) - (i) in $\{(1,1),(2,0),(2,2),(3,1),(3,3),(4,0),(4,2),(4,4)\}$.

5.7.1 CIELUV Colour Space

The CIE 1976 $L^*u^*v^*$ Colour Space (also written CIELUV) is a uniform colour space where the Euclidean distances between points in the space is proportional to human perception of differences in colour [46]. In contrast, the RGB colour space represents a non-uniform space with respect to the human visual system. The L^*u*v* colour components are given (in terms of the XYZ colour components) by the following equations [47]:

$$\begin{aligned} L^* &= 116(\frac{Y}{Y_n})^{1/3} - 16, (\frac{Y}{Y_n}) > 0.008856, \\ L^* &= 903.3(\frac{Y}{Y_n}), (\frac{Y}{Y_n}) \leq 0.008856, \\ u^* &= 13L^*(u' - u'_n), \\ v^* &= 13 * L^*(v' - v'_n), \end{aligned}$$

where

$$\begin{aligned} u' &= 4X/(X + 15Y + 3Z), \qquad u'_n = 4X_n/(X_n + 15Y_n + 3Z_n), \\ v' &= 9Y/(X + 15Y + 3Z), \qquad v'_n = 9Y_n/(X_n + 15Y_n + 3Z_n), \end{aligned}$$

and Y_n, X_n, and Z_n are based on the reference white point. For the results presented in this thesis, the D50 reference white point was used giving values of $Y_n = 1, X_n = 0.964221$, and $Z_n = 0.825211$. Similarly,

the XYZ colour components can be calculated using

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.607 & 0.174 & 0.200 \\ 0.299 & 0.587 & 0.114 \\ 0.000 & 0.006 & 1.116 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}.$$

Examples of the the average U and V components produced by the NEAR system are given in Fig. 18.

(a) (b) (c)

Figure 18: Example of CIE based probe functions using subimages of size 5×5: (a) Original image, (b) average U, and (c) average V.

6 Approximate Nearest Neighbours

This section describes the theory of approximate nearest neighbour searching, and introduces the Fast Library for Approximate Nearest Neighbours (FLANN) [48]. Further, this section describes the selection of parameters for configuring FLANN. FLANN is employed by the NEAR system to improve the speed of tolerance class calculation (see, *e.g.* [11]). Approximate nearest neighbour searching can be desribed as follows. Given a set of points $P = \{p_1, \ldots, p_n\}$ in an d-dimensional vector space X and a query point $q \in X$, the nearest neighbour search problem is defined as finding the point in P that is closest to q [49]. This problem arises in many research areas, especially in computer vision, and for high dimensional data, there is no known algorithm that performs much better than a linear search of the data points in P [49]. As a result, α-approximate[2] nearest neighbour searching has been introduced where query times can be reduced by orders of magnitude while sill achieving near-optimal accuracy. An α-approximate nearest neighbour to a query point $q \in X$ is defined as $p \in X$ if $dist(p, q) \leq (1 + \alpha) dist(p^*, q)$ where p^* is the true nearest neighbour [49].

FLANN uses two data structures to efficiently perform approximate nearest neighbour searches, namely the randomized kd-tree algorithm and the hierarchical k-means tree algorithm [49]. A kd-tree organizes the data using a binary tree where the tree nodes are points from P. Since points belong to a d-dimensional vector space, each node must have an associated splitting dimension (*i.e.* a dimension used to divide subsequent nodes in the tree). The next data point added to the tree is assigned to either the left or right child node depending on whether its value in the splitting dimension is less than or greater than the value of the current node. The kd-tree algorithm used in FLANN is called randomized because the splitting dimension for each node is selected randomly from the first D dimensions that have the greatest variance [49]. The other data structure used is the hierarchical k-means tree. This structure is created by recursion, *i.e.* the set of data is partitioned into K regions using the k-means clustering algorithm and then each region is again partitioned into K regions *etc.* The recursion is terminated when there are less than K data points in a region [49].

FLANN is the ideal library for performing approximate nearest neighbour searching because of the option for automatic optimization. The choice of algorithm used for approximate nearest neighbour searching

[2]Note: the symbol α is being used instead of ϵ (as is traditional in the literature) to avoid confusion with the tolerance relation.

is highly dependent on the dataset [49]. Consequently, the FLANN library has an option to select automatically the search algorithm and to optimize the input parameters of the selected algorithm. Both options are based on the points in P. Optimization is guided by a set of parameters specified by the user in the following equation

$$\text{cost} = \frac{s + w_b b}{(s + w_b b)_{\text{opt}}} + w_m m,$$

where s is the search time for the number of vectors in the sample dataset, b is the build time, $m = m_t/m_d$ is the ratio of memory used for the tree and memory used to store the data, w_b is the importance of build time over search time, and w_m is the importance of memory overhead [49]. Setting $w_b = 0$ means that the fastest search time is desired, and similarly, setting $w_m = 0$ means that faster search time is more important then memory requirements. Additionally, optimization is also performed based on the desired precision (percentage of query points for which the correct nearest neighbour is found) of the results from a nearest neighbour search (see [49] for more details). To generate the results presented here, a target precision of 0.8 was used together with $w_b = w_m = 0$.

The following describes the FLANN parameters (see [48] for more details):

- Auto. Parm. Select: Determines whether the NEAR system uses the FLANN library feature of automatic parameter selection. Selecting yes adds significant time to calculation process. Also note, for comparing a query image to a directory, the automatic parameter selection occurs before each image comparison.
- Algorithms: The algorithm to use for building the index. The possible values are ‘linear’, ‘kdtree’, ‘kmeans’, and ‘composite’. The ‘linear’ option does not create any index, it uses brute-force search in the original dataset points, ‘kdtree’ creates one or more randomized kd-trees, ‘kmeans’ creates a hierarchical kmeans clustering tree and ‘composite’ is a mix of both kdtree and kmeans trees.
- Checks: Denotes the number of times the tree(s) in the index should be recursively traversed. A higher value for this parameter would give better search precision, but also take more time.
- cb index: This parameter (cluster boundary index) influences the way exploration is performed in the hierarchical kmeans tree. When cb index is 0, the next kmeans domain to be explored is choosen to be the one with the closest centre. A value greater than zero takes into account the size of the domain.
- Trees: The number of randomized kd-trees to create. This parameter is required only when the algorithm used is ‘kdtree’.
- Branching: The branching factor to use for the hierarchical kmeans tree creation. While kdtree is always a binary tree, each node in the kmeans tree may have several branches depending on the value of this parameter. This parameter is required only when the algorithm used is ‘kmeans’.
- Iterations: The maximum number of iterations to use in the kmeans clustering stage when building the kmeans tree. A value of -1 used here means that the kmeans clustering should be performed until convergence. This parameter is required only when the algorithm used is ‘kmeans’.
- Centers Init.: The algorithm to use for selecting the initial centers when performing a kmeans clustering step. The possible values are ‘random’ (picks the initial cluster centers randomly), ‘gonzales’ (picks the initial centers using the Gonzales algorithm), and ‘kmeanspp’ (picks the initial centers using the algorithm suggested in [50]).
- Target Precision: A number between 0 and 1 specifying the percentage of the approximate nearest-neighbor searches that return the exact nearest-neighbor. Using a higher value for this parameter gives more accurate results, but the search takes longer.

- Build Weight: Specifies the importance of the index build time compared to the nearest-neighbor search time. In some applications it's acceptable for the index build step to take a long time if the subsequent searches in the index can be performed very fast. In other applications it's required that the index be built as fast as possible even if that leads to slightly lower search times
- Memory Weight: Used to specify the trade off between time (index build time and search time) and memory used by the index. A value less than 1 gives more importance to the time spent and a value grerater than 1 gives more importance to the memory usage.
- Num. of Threads: This parameter is specific to the NEAR system. The process involved in finding tolerance classes is to determine all the tolerance classes containing a specific object, then proceed to the next object in the queue, making these calculations easy to perform in parallel (see, *e.g.* [11]). As a result, the NEAR system is multithreaded in order to reduce computation time. The program was written on a quad-core machine, and thus the default number of threads is 4.

7 Equivalence class frame

This frame calculates equivalence classes using the Indiscernibility relation of Defn. 1, *i*.e., given an image X, it will calculate $X_{/\sim_{\mathcal{B}}}$ where the objects are subimages of X. A sample calculation using this frame is given in Fig. 19 and was obtained by the following steps:

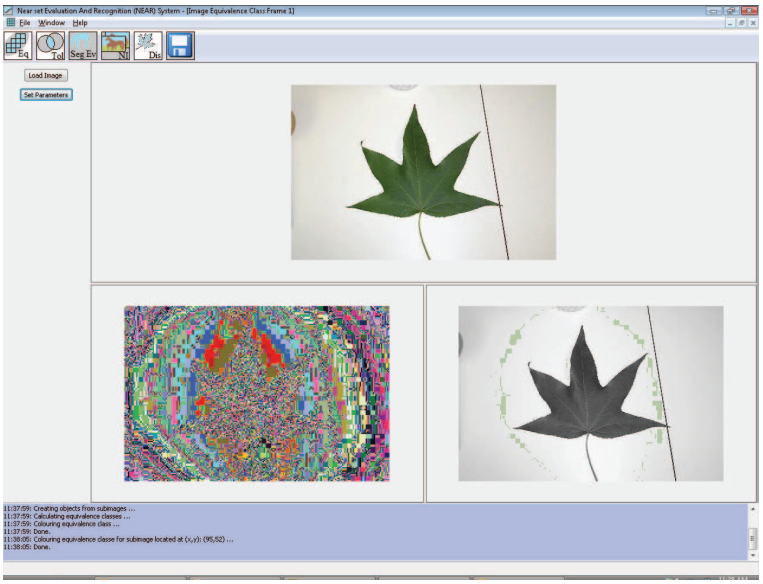


Figure 19: Sample run of the equivalence class frame using a window size of 2×2 and $\mathcal{B} = \{\phi_{\text{Average Grey}}\}$.

1. Click *Load Image* button and select an image.
2. Click the *Set Parameters* button.
3. Select window size. The value is taken as the square root of the area for a square subimage, *e*.g., a value of 5 creates a subimage of 25 pixels.
4. Select number of features (maximum allowed is 24).

5. Select features (see Section 5 for a list of probe functions).

6. Click *Run*.

The result is given in Fig. 19. The bottom left window contains an image of the equivalence classes where each colour represents a single class. The bottom right window is used to display a single equivalence classes by clicking in any of the three images. The coordinates of the mouse click determine the equivalence class that is displayed. The results may be saved by clicking on the save button.

8 Tolerance class frame

This frame calculates tolerance classes using the Perceptual Tolerance Relation of Defn. 4, where the objects are subimages of X. A sample calculation using this frame is given in Fig. 20 and was obtained by the following steps:

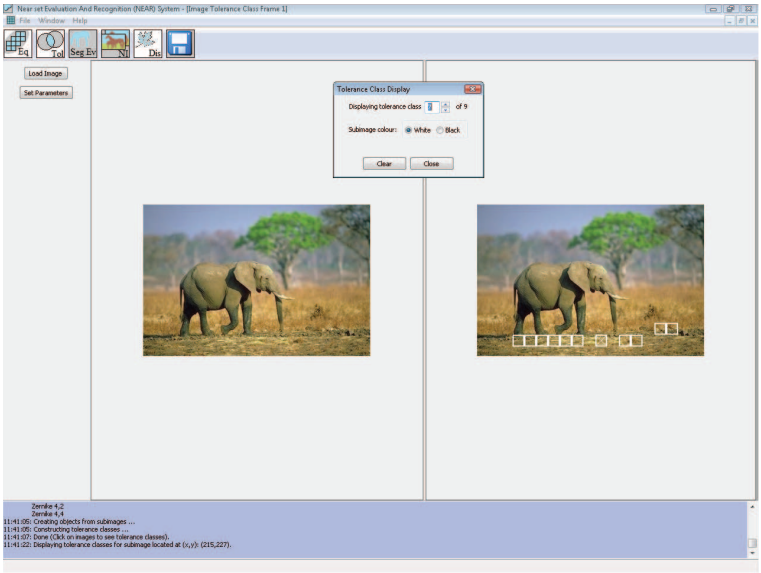


Figure 20: Sample run of the tolerance class frame using a window size of 20×20, 18 features, and $\varepsilon = 0.7$.

1. Click *Load Image* button and select an image.

2. Click the *Set Parameters* button.

3. Select window size. The value is taken as the square root of the area for a square subimage, *e*.g., a value of 5 creates a subimage of 25 pixels.

4. Select ε, a value in the interval $[0, \sqrt{l}]$ where l is the number of features (length of object description).

5. Select number of features (maximum allowed is 24).

6. Select features (see Section 5 for a list of probe functions).

7. Click on *FLANN Parameters* tab, and select the FLANN parameters for calculating tolerance classes (see Section 6 for details).

8. Click *Run*.

The result is given in Fig. 20 where the left side is the original image, and the right side is used to display the tolerance classes. Since the tolerance relation does not partition an image, the tolerance classes are displayed upon request. For instance, by clicking on either of the two images, a window appears letting the user display each tolerance classes containing the subimage selected by the mouse. Further, the subimage containing the mouse click contains an 'X', and the subimages can be coloured white or black.

9 Segmentation evaluation frame

This frame performs segmentation evaluation using perceptual morphology as described in [2, 6], where the evaluation is labelled the Near Set Index (NSI). For instance, given a set of probe functions $\mathcal{B}$, and an image A, this frame can perform the perceptual erosion or dilation using $B = O_{/\sim_{\mathcal{B}}}$ as the SE. Also, the NSI is calculated if perceptual erosion was selected. A sample calculation using this frame is given in Fig. 21 and was obtained by the following steps:

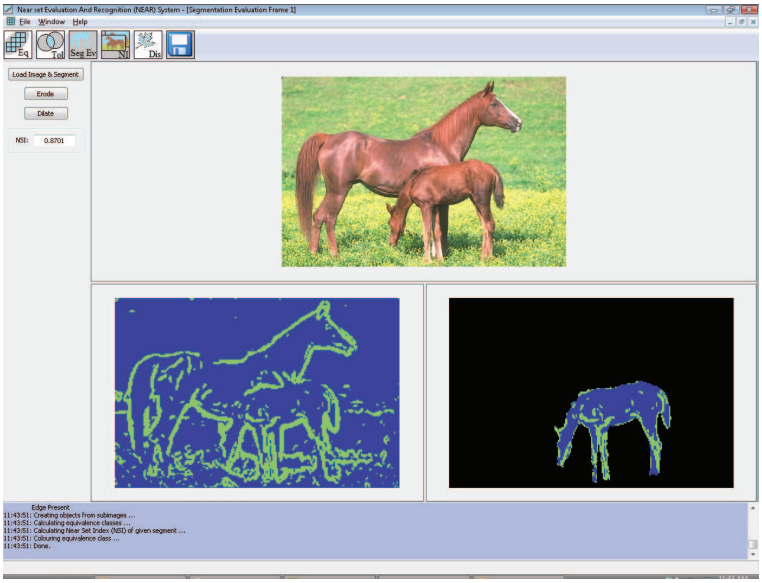


Figure 21: Sample run of the segmentation evaluation frame using a window size of 2×2, and $\mathcal{B} = \{\phi_{\text{Edge Present}}\}$.

1. Click *Load Image & Segment* button.
2. Select an image click *Open*.
3. Select segmentation image and click *Open*. Image should contain only one segment and the segment must be white $(255, 255, 255)$ and the background must be black $(0, 0, 0)$. The image is displayed in the top frame, while the segment is displayed in the bottom right (make sure this is the case).
4. Click either *Erode* to perform perceptual erosion and segmentation evaluation, or *Dilate* to perform perceptual dilation (no evaluation takes place during dilation).
5. Select window size. The value is taken as the square root of the area for a square subimage, *e*.g., a value of 5 creates a subimage of 25 pixels.

6. Select number of features (maximum allowed is four).
7. Select features (see Section 5 for a list of probe functions).
8. Click *Run*

The result is given in Fig. 21. The bottom left window contains an image of the equivalence classes where each colour represents a different class. The bottom right window contains either the erosion or dilation of the segment. Clicking on any of the three images will display the equivalence class containing the mouse click in the bottom right image. The NSI is also displayed on the left hand side (if applicable).

10 Near image frame

This frame is used to evaluate the similarity of images using the similarity measures given in Section 3. The user has the option of comparing a pair of images (and viewing the resulting tolerance classes), or comparing a query image to an entire directory of images. The following two subsections outline the steps involved under both options.

10.1 Evaluating a pair of images

The steps involved in comparing a pair of images is a follows, and sample output for this process is given in Fig. 22.

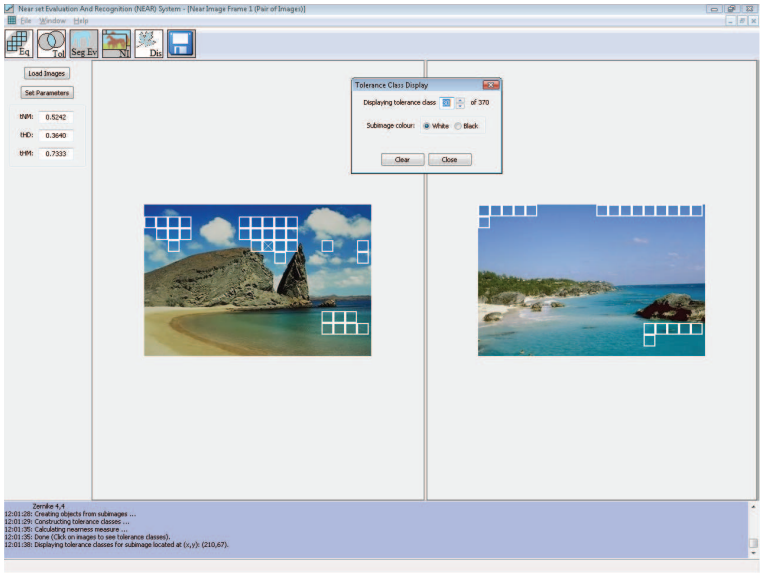


Figure 22: Sample run comparing a pair of images using a window size of 20×20, 18 features, and $\varepsilon = 0.7$.

1. Select the *New near image window* icon, select File→New near image window, or press Alt+N.
2. Select *A pair of images* (the default value) from the *Select type of Comparison* window, and click OK.
3. Click *Load Images* button and select two images.
4. Click the *Set Parameters* button.

5. Select window size. The value is taken as the square root of the area for a square subimage, *e*.g., a value of 5 creates a subimage of 25 pixels.
6. Select ε, a value in the interval $[0, \sqrt{l}]$ where l is the number of features (length of object description).
7. Select number of features (maximum allowed is 24).
8. Select features (see Section 5 for a list of probe functions).
9. Click on *FLANN Parameters* tab, and select the FLANN parameters for calculating tolerance classes (see Section 6 for details).
10. Click *Run*.

The result is given in Fig. 22 where the left side contains the first image, and the right side contains the second image. Clicking in any of the two images will bring up a window that allows the user to view each tolerance class containing the subimage selected by the mouse. Further, the subimage containing the mouse click contains an 'X', and the subimages can be coloured white or black. Also, the similarity of the images is evaluated using the measures described in Section 3, where the results are displayed on the left hand side.

10.2 Comparing a query image with a directory of images

The steps involved in comparing a query image with a directory containing images is as follows, and sample output for this process is given in Fig. 23.

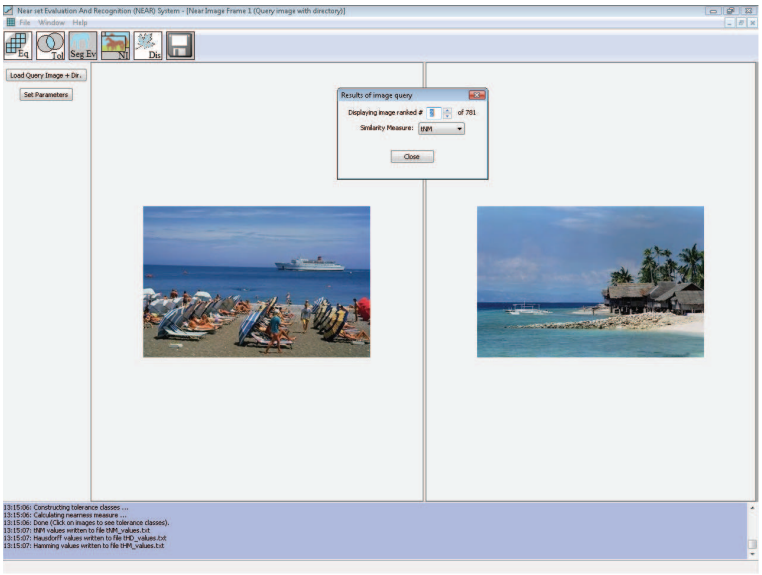


Figure 23: Sample run comparing an image to a directory of images using a window size of 20×20, 18 features, and $\varepsilon = 0.2$.

1. Select the *New near image window* icon, select File→New near image window, or press Alt+N.
2. Select *Query image with a directory of images* from the *Select type of Comparison* window, and click OK.

3. Click *Load Query Image + Dir.* button and select an image plus a directory containing images for comparison with query image.
4. Click the *Set Parameters* button.
5. Select window size. The value is taken as the square root of the area for a square subimage, *e*.g., a value of 5 creates a subimage of 25 pixels.
6. Select ε, a value in the interval $[0, \sqrt{l}]$ where l is the number of features (length of object description).
7. Select number of features (maximum allowed is 24).
8. Select features (see Section 5 for a list of probe functions).
9. Click on *FLANN Parameters* tab, and select the FLANN parameters for calculating tolerance classes (see Section 6 for details).
10. Click *Run*.

The result is given in Fig. 22 where the left side contains the query image, and the right side contains an image from the directory. Clicking in any of the two images will bring up a window that allows the user to view the images from the directory in the order they were ranked by the selected similarity measure (see Section 3 for a description of the measures). In addition, three output files are created containing the similarity measure of each image in the database, sorted from most similar to least similar. Finally, three figures are also displayed plotting the similarity measures vs. images in the directory for all three measures. Note, the results are sorted from best to worst, so the output files are also required to related the abscissae to actual image files.

11 Feature display frame

This frame is used to display the output of processing an image with a specific probe function. A sample calculation using this frame is given in Fig. 24 and was obtained by the following steps:

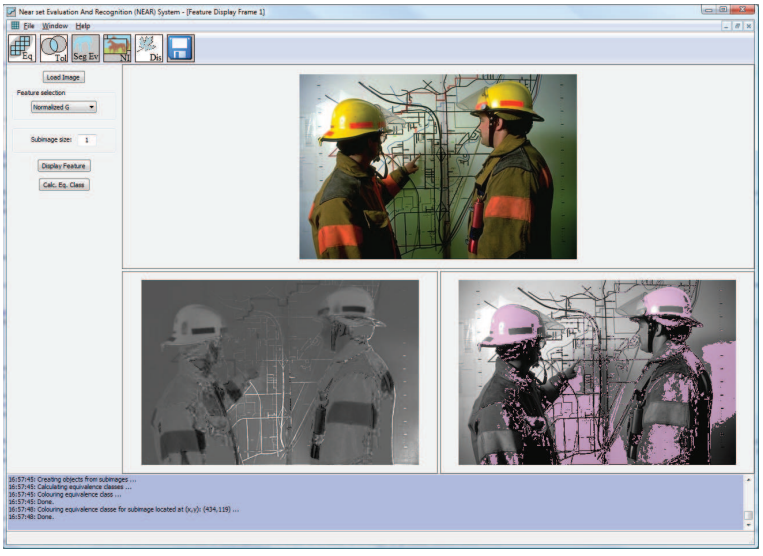


Figure 24: Sample run of the feature display frame.

1. Click *Load Image* button and select an image.
2. Select features (see Section 5 for a list of probe functions).
3. Select probe function
4. Click *Display feature*.

References

[1] C. Henry and J. F. Peters, "Image pattern recognition using near sets," in *Proceedings of the Eleventh International Conference on Rough Sets, Fuzzy Sets, Data Mining and Granular Computer (RSFDGrC 2007), Joint Rough Set Symposium (JRS07), Lecture Notes in Computer Science*, vol. 4482, 2007, pp. 475–482.

[2] ——, "Near set index in an objective image segmentation evaluation framework," in *Proceedings of the GEOgraphic Object Based Image Analysis: Pixels, Objects, Intelligence*, University of Calgary, Alberta, 2008, pp. 1–8.

[3] J. F. Peters and S. Ramanna, "Affinities between perceptual granules: Foundations and perspectives," in *Human-Centric Information Processing Through Granular Modelling*, A. Bargiela and W. Pedrycz, Eds. Berlin: Springer-Verlag, 2009, pp. 49–66.

[4] J. F. Peters, "Discovery of perceptually near information granules," in *Novel Developements in Granular Computing: Applications of Advanced Human Reasoning and Soft Computation*, J. T. Yao, Ed. Hersey, N.Y., USA: Information Science Reference, 2009, p. *in press*.

[5] C. Henry and J. F. Peters, "Perception-based image classification," *International Journal of Intelligent Computing and Cybernetics*, p. *accepted*, 2010.

[6] ——, "Perception image analysis," *International Journal of Bio-Inspired Computation*, vol. 2, no. 2/3, p. *In Press*, 2010.

[7] J. F. Peters and P. Wasilewski, "Foundations of near sets," *Elsevier Science*, vol. 179, no. 1, pp. 3091–3109, 2009.

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

[9] ——, "Corrigenda and addenda: Tolerance near sets and image correspondence," *International Journal of Bio-Inspired Computation*, vol. 2, no. 5, p. *in press*, 2010.

[10] 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. 3, no. 6, pp. 955–968, 2009.

[11] C. Henry, "Near Sets: Theory and Applications," Ph.D. dissertation, University of Manitoba, CAN, 2010.

[12] C. Christoudias, B. Georgescu, and P. Meer, "Synergism in low level vision," in *Proceedings of the 16th International Conference on Pattern Recognition*, vol. 4, Quebec City, 2002, pp. 150–156.

[13] wxWidgets, "wxwidgets cross-platform gui library v2.8.9," 2009, www.wxwidgets.org.

[14] M. Pavel, ""shape theory" and pattern recognition," *Pattern Recognition*, vol. 16, no. 3, pp. 349–356, 1983.

[15] J. F. Peters, "Near sets. special theory about nearness of objects," *Fundamenta Informaticae*, vol. 75, no. 1-4, pp. 407–433, 2007.

[16] ——, "Classification of objects by means of features," in *Proceedings of the IEEE Symposium Series on Foundations of Computational Intelligence (IEEE SCCI 2007)*, Honolulu, Hawaii, 2007, pp. 1–8.

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

[18] Z. Pawlak, "Classification of objects by means of attributes," Institute for Computer Science, Polish Academy of Sciences, Tech. Rep. PAS 429, 1981.

[19] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 2nd ed. Toronto: Prentice-Hall, 2002.

[20] M. Pawlak, *Image analysis by moments: reconstruction and computational aspects*. Wrocław: Wydawnictwo Politechniki, 2006.

[21] E. Orłowska, "Semantics of vague concepts. applications of rough sets," Institute for Computer Science, Polish Academy of Sciences, Tech. Rep. 469, 1982.

[22] ——, "Semantics of vague concepts," in *Foundations of Logic and Linguistics. Problems and Solutions*, G. Dorn and P. Weingartner, Eds. London/NY: Plenum Pres, 1985, pp. 465–482.

[23] 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.

[24] 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, p. *accepted*, iSBN 13: 9781439803295.

[25] F. Hausdorff, *Grundzüge der mengenlehre*. Leipzig: Verlag Von Veit & Comp., 1914.

[26] ——, *Set theory*. New York: Chelsea Publishing Company, 1962.

[27] M. A. Ferrer, A. Morales, and L. Ortega, "Infrared hand dorsum images for identification," *IET Electronic Letters*, vol. 45, no. 6, pp. 306–308, 2009.

[28] Magick++, "Imagemagick image-processing library," 2009, www.imagemagick.org.

[29] M. Weber, "Leaves dataset," 1999, url: www.vision.caltech.edu/archive.html.

[30] J. Marti, J. Freixenet, J. Batlle, and A. Casals, "A new approach to outdoor scene description based on learning and top-down segmentation," *Image and Vision Computing*, vol. 19, pp. 1041–1055, 2001.

[31] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proceedings of the 8th International Conference on Computer Visison*, vol. 2, 2001, pp. 416–423.

[32] N. R. Pal and S. K. Pal, "Entropy: A new definition and its applications," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 21, no. 5, pp. 1260 – 1270, 1991.

[33] T. Seemann, “Digital image processing using local segmentation,” Ph.D. dissertation, School of Computer Science and Software Engineering, Monash University, 2002.

[34] N. R. Pal and S. K. Pal, “Some properties of the exponential entropy,” *Information Sciences*, vol. 66, pp. 119–137, 1992.

[35] S. Mallat and S. Zhong, “Characterization of signals from multiscale edges,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 7, pp. 710–732, 1992.

[36] H. Tamura, M. Shunji, and T. Yamawaki, “Textural features corresponding to visual perception,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 8, no. 6, pp. 460–473, 1978.

[37] R. M. Haralick, “Textural features for image classification,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-3, no. 6, pp. 610–621, 1973.

[38] ——, “Statistical and structural approaches to texture,” *Proceedings of the IEEE*, vol. 67, no. 5, pp. 786–804, 1979.

[39] P. Howarth and S. Ruger, “Robust texture features for still-image retrieval,” *IEE Proceedings Vision, Image, & Signal Processing*, vol. 152, no. 6, pp. 868–874, 2005.

[40] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 3rd ed. Upper Saddle River, NJ, USA: Person/Prentice Hall, 2008.

[41] C. H. Teh and R. T. Chin, “On image analysis by the methods of moments,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, no. 4, pp. 496–513, 1988.

[42] A. Khotanzad and Y. H. Hong, “Invariant image reconstruction by zernike moments,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 5, pp. 489–497, 1990.

[43] W. Y. Kim and Y. S. Kim, “A region-based shape descriptor using zernike moments,” *Signal Processing: Image Communication*, vol. 16, pp. 95–102, 2000.

[44] R. S. Choraś, T. Andrysiak, and M. Choraś, “Integrated color, texture and shape information for content-based image retrieval,” *Pattern Analysis & Applications*, vol. 10, no. 4, pp. 333–343, 2007.

[45] P. Toharia, O. D. Robles, A. Rodriguez, and L. Pastor, “A study of zernike invariants for content-based image retrieval,” in *Advances in Image and Video Technology*. Berlin Heidelberg: Springer-Verlag, 2007, vol. LNCS 4872, pp. 944–957.

[46] J. M. Kasson and W. Plouffe, “An analysis of selected computer interchange color spaces,” *ACM Transactions on Graphics*, vol. 11, no. 4, pp. 373–405, 1992.

[47] K. Nallaperumal, M. S. Banu, and C. C. Christiyana, “Content based image indexing and retrieval using color descriptor in wavelet domain,” in *International Conference on Computational Intelligence and Multimedia Applications (ICCIMA 2007)*, vol. 3, 2007, pp. 185–189.

[48] M. Muja, “FLANN - Fast Library for Approximate Nearest Neighbors,” 2009, http://www.cs.ubc.ca/ mariusm/index.php/FLANN/FLANN.

[49] M. Muja and D. G. Lowe, “Fast approximate nearest neighbors with automatic algorithm configuration,” in *International Conference on Computer Vision Theory and Applications (VISAPP)*, Lisbon, Portugal, 2009, pp. 331–340.

[50] D. Arthur and S. Vassilvitskii, “k-means++: The advantages of careful seeding,” in *Proceedings of the eighteenth annual ACM-SIAM symposium on discrete algorithms*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035.