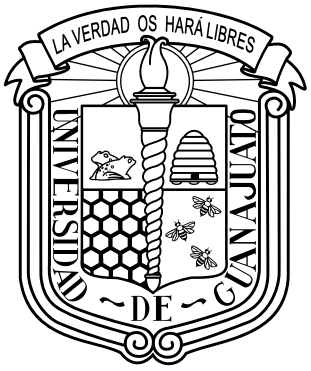


UNIVERSIDAD DE GUANAJUATO

CAMPUS IRAPUATO–SALAMANCA
DIVISIÓN DE INGENIERÍAS

Características Visuales para el Reconocimiento Rápido de Objetos

T E S I S

QUE PARA OBTENER EL GRADO DE:

DOCTOR EN INGENIERÍA ELÉCTRICA

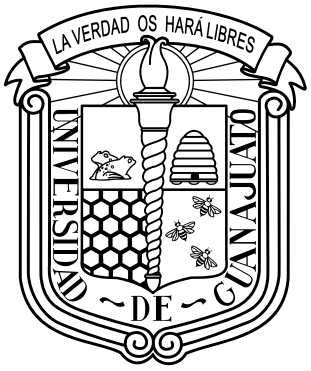
PRESENTA:

M.I. Fernando Enrique Correa Tomé

DIRECTOR:

Dr. Raúl Enrique Sánchez Yáñez
Universidad de Guanajuato

SALAMANCA, GTO. Septiembre 2015



UNIVERSITY OF GUANAJUATO

CAMPUS IRAPUATO–SALAMANCA
ENGINEERING DIVISION

Visual Features for Fast Object Recognition

T H E S I S

TO OBTAIN THE DEGREE OF:

DOCTOR OF ELECTRICAL ENGINEERING

PRESENTS:

M.Eng. Fernando Enrique Correa Tomé

SUPERVISOR:

Dr. Raúl Enrique Sánchez Yáñez
University of Guanajuato

SALAMANCA, GTO. September 2015

Contents

List of Figures

List of Tables

Acknowledgements

To the University of Guanajuato, and to its Engineering Division at the Campus Irapuato–Salamanca, for the academic training provided in the Electrical Engineering Doctoral program.

To the Office of Research and Graduate Programs (Dirección de Apoyo a la Investigación y al Posgrado) DAIP of the University of Guanajuato, for the financial support received via the scholarship "Training for Young Researchers (Formación de Jóvenes Investigadores)", with number NUA:384679.

To the Integral Institutional Strengthening Program (Programa Integral de Fortalecimiento Institucional) PIFI 2013, for the funding provided to cover publication costs.

To the National Council of Science and Technology (Consejo Nacional de Ciencia y Tecnología) CONACyT, for the financial support provided during the course of the doctoral program, through the scholarship 295697/226942.

Perception, cognition and robotics sinergy

To the Laboratory of Vision, Robotics, and Artificial Intelligence (Laboratorio de Visión, Robótica e Inteligencia Artificial) LaViRIA, for the resources and support provided to develop the investigation work, during the course of the doctorate.

Abstract

In this thesis study, we explored the recognition of objects using visual features. From the different approaches we studied during the development of this work, three main research lines arise: a study of fast object recognition by shape, a fast segmentation methodology, and an online object recognition system that learns from example images. For the first study, a visual similarity metric of shapes, based on Precision–Recall graphs is presented, as an alternative to the widely used Hausdorff distance (HD). Such metric, called maximum cardinality similarity metric, is computed between a reference shape and a test template, each one represented by a set of edge points. We address this problem using a bipartite graph representation of the relationship between the sets. The matching problem is solved using the Hopcroft–Karp algorithm, taking advantage of its low computational complexity. We present a comparison between our results and those obtained from applying the partial Hausdorff distance (PHD) to the same test sets. Similar results were found using both approaches for standard template-matching applications. Nevertheless, the proposed methodology is more accurate at determining the completeness of partial shapes under noise conditions. Furthermore, the processing time required by our methodology is lower than that required to compute the PHD, for a large set of points. The second study presents a split-and-merge segmentation methodology, that uses integral images to improve the execution time. We call our methodology integral split and merge (ISM) segmentation. The integral images are used here to calculate statistics of the image regions in constant time. Those statistics are used to guide the splitting process by identifying the homogeneous regions in the image. We also propose a merge criterion that performs connected component analysis of the homogeneous regions. Moreover, the merging procedure is able to group regions of the image showing gradients. Furthermore, the number of regions resulting from the segmentation process is determined automatically. In a

series of tests, we compare ISM against other state-of-the-art algorithms. The results from the tests show that our ISM methodology obtains image segmentations with a comparable quality, using a simple texture descriptor instead of a combination of color–texture descriptors. The proposed ISM methodology also has a piecewise linear computational complexity, resulting in an algorithm fast enough to be executed in real time. Finally, in the third study we present an object recognition system based on histograms of visual features, obtained from the images containing the objects to recognize. The system is divided in two stages, the feature extraction, and the classification stages. The first stage involves transforming the original image to the CIELuv color space, and then extracting four different visual features from this space. The features are: the color hue, the color saturation, the luminance, and a texture feature that is the standard deviation of the luminance, calculated for local neighborhoods. A histogram is then calculated from each feature and then merged together, in order to obtain a combined feature histogram called THSL histogram. Later, a collection of THSL histograms is built as our knowledge database, and is used for the classification of unseen images. This classification stage consists in the extraction of a THSL histogram from the image under test. This histogram is then compared against all the histograms in the knowledge database, and the label of the most similar histogram is retrieved as the classification result. We performed a series of tests, using images of small objects taken from different points of view. The images have no background, the objects are at the same distance from the camera, and the illumination is constant. The results obtained show a classification success rate around the 96% for individual objects, under the aforementioned conditions.

Chapter 1

Introduction

This thesis study explores the use of visual features for the recognition of objects, with a focus on real-time performance. We developed a shape feature approach for object recognition, using a template matching methodology that is performed in real-time. Also, a real-time image segmentation methodology is proposed, using integral images to improve the computational time of a split-and-merge approach. Finally, we present an object recognition system, using combined histograms of color and texture features to describe the objects. The system is able to learn from examples in real-time, and achieves a high recognition rate, for isolated objects under controlled conditions.

In this chapter, an overview of the work developed for this thesis project is presented. The thesis guidelines are described in Section 1.1. Then, Section 1.2 is included to clarify the different approaches, taken during the development of the different research lines of this work.

1.1 Thesis guidelines

In this section, we include the guideline used for the development of this thesis work. This guideline presents different approaches that may be explored in order to achieve the general objective of this study: the development of an object recognition system.

1.1.1 Summary

In this thesis study, we propose the development of a computational environment for the learning of visual features, relevant for the detection of specific objects in a natural scene.

The system should be able to extract different features from different views of the same object, and select those that allow the representation of that object.

In a stage using low-level computer vision, we propose the use of color, texture and shape features, for the description of objects. Moreover, an analysis of the data should be performed in order to automatically extract the relevant data, and the features that distinguish different objects.

In a recognition stage, the system should be able to identify the presence of the learned objects, among the different regions in the scene under analysis. The system will be tested and validated through different applications in the computer vision area.

1.1.2 Problem approach

The learning of objects from different views is fundamentally a perception task, tightly relating the human visual system and our learning capacity. Fundamentally, it should be consider how is that we, human beings, perceive our environment, and how do we dynamically extract the information that is important for us, maybe influenced by the context and by particular situations. To emulate this human processing system in an automatic system, the user should only show the object of interest, and the system should be able to learn from such demonstration. The development of a task such like this, that comprises the characterization and the automatic learning of objects from demonstrations [69], is a goal sought by researchers from a variety of fields.

The machine learning techniques have proven to be promissory for the optimization of visual attention tasks [10]. In some cases, the learning is performed by using sample cases or study instances.

The machines are commonly used to calculate functions that indicate certain level of fitness, likeliness, or error between what is observed and the

data stored in memory. The complexity of the problem considerably increases when such functions are not known a priori, and should be learned.

In one hand, in this project we propose the study of the visual characterization of objects and the learning of relevant features, and the subsequent recognition of those objects. On the other hand, we have as a goal the development of a computational system that allows us to incorporate such tasks.

1.1.3 General objective

To explore different approaches in order to develop a computational environment for the learning of objects from views, and the validation and publication of the scientific results obtained from this exploration.

Applicability of the results

The results of the research may be applied in a variety of computational vision tasks, such as the recognition and classification of objects, searches for images that contain those objects, or the dynamic tracking of objects, among others.

1.1.4 Particular objectives

- The study of the visual features of the objects, the extraction of those features, and their selection accordingly to their particular relevance.
- The evaluation of a number of machine learning algorithms.
- The integration of a development environment for systems of feature extraction and selection, the learning of objects regarding to such features, and the dynamic recognition of objects in the scenes under analysis.

1.1.5 Proposed methodology

In this section we briefly present the main methods to consider for addressing the presented problem.

This project has been conceived as an ensemble of different techniques and algorithms, mainly computational vision and machine learning methods, clarifying that this problem may have multiple edges.

Next, the main modules to include in the system are indicated, along with possible approaches for their development.

Feature extraction and selection

A feature is considered as an individual metric of a property of some phenomena under observation. For image analysis, numerical values of color, texture or shape are commonly considered for the characterization of elements in the scene [64].

Given a training set of d feature vectors (instances) of fixed size $\{(X_1, Y_1), \ldots, (X_d, Y_d)\}$, where an instance X_i is described as an assignment of values $X_i = (X_{i1}, \ldots, X_{iN})$, corresponding to a set of features $F = (F_1, \ldots, F_N)$, and assigned to a labeled class Y_i. The classification task consists in including a classifier $H : X \to Y$ that accurately obtains the labels of the new instances, based on the values of their features.

For real-world applications, the relevant set of features for the given task is rarely well defined. Commonly, it is a combination of some features. In this regard, the selection of independent or discriminant features is essential for any algorithm, such is our case for the recognition of objects.

Two possible settings arise at this point. The first one uses a large number of features, making not only more difficult and slow any learning process, but also adding confusion to the segmentation and classification processes. The second one uses a relatively small number of features, however its calculation is also computationally expensive, e.g. for real-time applications.

Intuitively, the use of a large number of features should lead to a better prediction. However, in the practice, large data sets along with a large number of features not only slow down the learning process, the use of redundant features also deteriorate the classifier outcome.

The selection of features [27] addresses this problem through the identification of relevant features, and by removing those features that are irrelevant, redundant, or noisy [91]. This improves the performance of any classifier, re-

duces the computational cost, and provides a better comprehension of the data sets [19] [54].

This stage is closely related to the learning stage, because the problem we face is precisely to determine what features are essential for the learning–recognition task, and which ones are redundant or even irrelevant.

Concept learning

In the project, the concept learning stage is closely related to the feature extraction and selection, and then to the recognition of the objects of interest. Both the features and their evaluation are aspects not known a priori and require to be learned.

The learning module should receive as input a set of classified objects, their attributes, and maybe some context explanation about their ranges and properties. Then, this module should automatically build the functions for the evaluation of other instances [58].

The literature contains a variety of approaches to the supervised machine learning. We can mention the artificial neural networks (ANN) [92], the support vector machines (SVM) [86], the classifier ensemble methods [71], or the continuous scalable template matching (CSTM) [10]. Also, there are methods based on trees, rules, connectionist nets, probabilistic nets, statistic models, evolutionary models, among others.

It is established that the system to develop should learn those features from an object that is common to the different views, presented to the system in the training stage [5]. In this regard, the system should be able to identify which descriptors are important in the learning–recognition process [66]. For the process of selection of an adequate set of features, the relevance of a feature may be a concept to explore [7]. For some case-based learning algorithms, even some psychological restrictions may be used effectively as indicators to guide the selection of features and their weighting [12].

During the development of the project, we will seek to explore those learning algorithms that consider the concepts as groups in a feature space [61]. This idea, associated to the weighting of those features in the groups, may be important for the approach. In this way, each object should be

represented by a feature vector or even by a tuple of them, along with a weighting metric associated to the features in the given space [62].

Segmentation

Even tough this stage may be seen as only related to the project, we consider it useful for effective object recognition. Because we look for a dynamic integrated system, it is not feasible to assume that the image (scene) has been correctly segmented, and that the segment classification is straightforward.

The image segmentation is one of the fundamental problems of the computer vision and pattern recognition fields. In short, the image segmentation consists in (*i*) separate the elements of the image in regions showing similar visual features, and (*ii*) to assign a label to each pixel in such a way that those pixels with the same label share the same visual features.

The outcome from the image segmentation process is a set of regions (groups of pixels) that together cover the entire image. On one hand, the segmentation should be made so the pixels in the same region are similar to each other, regarding to certain features or properties, such as the intensity, the color, or the texture. On the other hand, different regions, even adjacent ones, should be significantly different regarding to the considered features [75].

There are different approaches to image segmentation, we can mention methods using: heuristic approximations, detection of edges, region-based, and graph-based methods. Some heuristic approximations of general purpose for the segmentation of images are the thresholding methods [57], where decisions are taken based on the local information of the pixels. The methods based on the detection of edges [70] are oriented to the joint connection of lines of broken contours. These methods are prone to fail under the presence of an inadequate contrast. The methods based on regions [29], generally perform an image partition into regions connected by grouping neighboring pixels of similar intensity levels. Finally, for the methods based on graphs [23], the problem is commonly represented in therms of graphs. In this regard, each node of the graph corresponds to a pixel in the image, and an edge connects each pair of nodes, associating a weight to each edge accordingly to certain property of the pixels that it connects. Additionally, a number

of hybrid methods, using combinations of the aforementioned methods, had been also proposed.

Other approximations aim to formalize a goal criterion for the evaluation of a given segmentation, presenting the problem as an optimization one. Generally, the goal is to reduce the interclass variance, in the way performed by group analysis. Thereby, some methods had been proposed to handle segmentation problems. Among them we can find the k-means algorithm [30] as the most popular. This method partitions an image into several regions by using an iterative technique. Independently of the methodology used, the goal of the image segmentation is to simplify or change the representation of an image into something that is more significant and easy to analyze [75]. In our case, the candidate regions to be recognized as a previously learned object.

Object recognition

The object recognition may be seen as a supervised classification, that consists in the assignment of an object to a previously learned category.

The system should dynamically identify the similarities between the current view and the objects represented in memory [66], making this a challenging procedure. This approach to a real-time system impose a considerable number of restrictions to the development.

1.2 Thesis development

This section briefly describes the development stages of this thesis project. The work can be divided in three lines of research, presented next. Two articles regarding to this work were published in scientific journals with JCR impact factor, as mentioned below.

1.2.1 Identification of objects by shape

In the early stages of development of this thesis project, we explored the recognition of objects by shape, following the thesis guidelines. We used a metric developed in a previous work [18] in order to compare edge templates

of object shapes against edge images. This comparison allowed us to identify points in the image, where the probability of finding the shape of the object is high. The main advantage is that the method is fast enough to be used in real time applications. However, the method require to build a database of edge templates for each object, possibly for multiple views. A scientific article about this research was published in the Journal of Real-Time Image Processing [15]. This study is described in detail in Chapter 2

1.2.2 Fast image segmentation

Following the thesis guidelines, we explore the image segmentation as a means to extract information about the objects in an image. However, we found that the image segmentation methodologies are difficult to use in real-time systems.

In this regard, we developed a new segmentation algorithm that is based on a split-and-merge methodology. We incorporated a technique called *integral images*, in order to simplify the calculations required by the algorithm. The result of this line of work is an image segmentation algorithm, whose near-linear computational complexity allows it to be used in real time applications. Furthermore, the segmentation methodology proposed also obtains good results regarding to the quality of the segmentation, compared with other state-of-the-art methodologies. This work, discussed in detail in Chapter 3, produced an article published in the Journal of Electronic Imaging [16].

1.2.3 Object recognition system

For the object recognition system developed for this thesis work, we explored different color and texture image features. We choose four features, one of texture and three of color, to model the objects to be learned. We use an histogram-based approach, where the histograms obtained from the images are a combination of the chosen color and texture features. All histograms obtained from individual images are recorded in a knowledge database, used to recognize unseen examples of the objects learned. The resulting system achieves a classification rate around the 96% for the database used.

As a result of this work, we obtained an object recognition system that is able to learn in real time, from the examples presented to the system. Also,

the system is able to recognize a previously learned object, presented from unseen points of view.

Even though the classification rate for the system is very high, the results obtained from this system are not yet ready for publication. This is because at this stage of development, the system is not robust enough against illumination changes, scale, varying backgrounds, and other common problems present in images. Also, a thorough testing is still required in order to compare our system against state-of-the-art methodologies.

The object recognition system developed in this study is fully described in Chapter 4, along with a description of the tests performed and the results obtained.

1.2.4 Thesis organization

The next Chapters 2 to 4 present the research lines discussed above. Each chapter contains an introduction to its corresponding study, the methodology that was used, and a discussion about the results obtained. It is important to mention that, in order to maintain the mathematical notation simple, some of the symbols used are redefined in different chapters. This can be done because the mathematical notation of each chapter is independent from the others, so there is no ambiguity. Finally, the concluding remarks of this work are discussed in Chapter 5.

Chapter 2

Real-time template matching

In this chapter, a methodology that performs real-time template matching is presented. This methodology uses a similarity metric that is efficiently computed in near-linear time, and is used to compare edge templates (i.e. shape). This chapter is organized as follows. In Section 2.1, an introduction to the methodology used for real-time template matching is presented. Then, the details of the used methodology are described in Section 2.2. Finally, the tests performed using this methodology, and the results obtained from these tests are given in Section 2.3.

2.1 Introduction

Computer vision plays an important role in many applications nowadays. It is used to extract meaningful information from digital images and video, aiming to reproduce the human visual abilities. The extraction of such information normally requires the identification of objects in a scene and, before this information can be extracted, the presence of the object itself needs to be ascertained. The object is then located in the scene by obtaining different spatial features, such as translation, scale and orientation. For some video applications, the extraction of information needs to be performed in real-time, meaning that the processing time for a single frame (i.e. an image) is required to be below the frame rate of the video. Owing to this restriction, the video application requires to be optimized in time, and arises the need for fast algorithms.

To achieve real-time object detection, the volume of data and the complexity of the algorithms involved need to be reduced to fit the time requirements. In their article, Hossain *et al.* [37] point out that edge information offers more robustness than intensity information. Moreover, using edge information significantly reduces the amount of information to be processed. Nevertheless, object detection in edge images is affected by noise, changes in illumination, occlusions and small differences in edge locations for video sequences. Rucklidge [74] proposes a method based on the Hausdorff distance (HD) to identify objects under affine transformations in edge images, and Knauer *et al.* [48] propose a methodology also based on the HD that handles imprecision of the data. In both articles, the HD provides the robustness required to overcome the aforementioned problems found in edge images. Nevertheless, obtaining the HD is computationally expensive, having a $O(n^2)$ complexity expressed in Big-O notation, where n is the input size of the algorithm.

A number of approaches aim to use the HD in a more efficient way, to reduce the computing time required. Rucklidge [73] proposes a set of methods to efficiently search a large space of affine transformations to detect object projections, whilst Huttenlocher and Rucklidge [40] use the HD to locate objects under translation and scaling through a subdivision procedure that discards uninteresting regions, considerably reducing the number of evaluations. Moreover, Niu *et al.* [63] present a methodology based on Voronoi surfaces, increasing the calculation speed of the HD for image registration. Their methodology further reduces the data volume by keeping only the longer edges. For image registration tasks, Hossain *et al.* [36] present a low error approximation to the HD that is computed in a near-linear time. Following the idea of a HD replacement, Tsapanos *et al.* [78] propose a $O(\log n)$ algorithm based on HD for shape matching, applied to pedestrian detection. Other approaches include the use of hardware acceleration to achieve real-time image processing, such as the studies of Krishnamurthy *et al.* [49] and Hanniel *et al.* [28], in which GPU algorithms of the HD are presented.

Even though the use of the HD for image processing is not a recent approach, it is still widely used in the image processing field. Regarding to shape matching and image registration, Li and Stevenson [51] use a modified Hausdorff distance (MHD) as a similarity metric between curve segments called *sub-edges*, for a robust image registration. Also, Sim and Park [76] present an object alignment methodology that is robust against severe noise;

such methodology uses an MHD combined with the voting procedure from the Hough transform.

For biometric identification, Dastmalchi *et al.* [20] use a variant of the HD, the partial Hausdorff distance (PHD), proposed by Huttenlocher *et al.* [41] back in 1993. They report an increase in accuracy, and a reduction of the computing time required for face recognition applications. The HD had been previously used for face recognition purposes, such as in the article by Hu and Wang [39], where the HD is used to compare faces, surpassing conventional Hausdorff-based applications, and the work of Yang *et al.* [90], presenting a variant of the HD that combines normalized image gradients. This latter approach is robust against different illumination conditions for face recognition applications. Also, Lin *et al.* [52] propose two variations of the HD, used to make face recognition robust against facial expressions. In the field of biometrics, Ali *et al.* [2] present a methodology based on a variation of the HD, for palm print matching. This approach is robust against noise and occlusion. For their part, Lin *et al.* [53] use a weighted HD for automatic dental matching, with applications in forensic identification. In robotics, Ji *et al.* [43] use the HD to assert the similarity between polygons for robot localization. The HD approach is preferred in this area because robots often have limited computing resources. Also related to the robotics area, Xu [88] proposes a simultaneous camera autofocus and alignment method. The HD is used here to localize patterns and to obtain invariance against illumination changes.

In this study, we propose a similarity metric as an alternative to the PHD. Our approach is applied to edge images, being robust against small edge localization errors, partial edges, occlusions, and providing an accurate metric for edge completeness. Moreover, our algorithm is computed in a near-linear time, allowing it to be used in real-time applications. This methodology is an adaptation of the work of Correa-Tome *et al.* [18], originally conceived as an empirical segmentation evaluation metric [17] and used here for edge image comparison.

The proposed methodology works with binary images (i.e. images whose pixels have only two possible values, usually interpreted as black and white) containing edges, previously obtained using an edge detection algorithm such as the proposed by Canny [11]. The edge pixels (usually the white pixels) are

considered points in a two-dimensional space, whereas the non-edge pixels (usually the black pixels) are simply ignored.

The segmentation comparison methodology presented by Correa-Tome *et al.* [18] uses a standard evaluation technique described by van Rijsbergen [82] for information retrieval: the Precision and Recall graphs. This technique was previously used by Martin *et al.* [60] to compare the performance of edge detectors, using a collection of human-made references [59]. The Precision measure is proportional to the number of edge pixels correctly detected, whereas Recall is proportional to the number of edge pixels actually detected. In this study, Recall is used as a metric for the similarity between edge images, whilst Precision is not used.

Given two images with such characteristics, i.e. an image to test and a template, our methodology compares them by determining the number of edge pixels in the image to test that are present in the template. Doing this by means of simple overlapping performs poorly, since small edge localization errors are expected. Instead, our method inspects a neighborhood defined by a tolerance radius around a given pixel, looking for possible matchings. Then, a bipartite graph is built, where the two sets of nodes represent the two sets of points from each image, and the edges of the graph describe the relationships among possible matching edge pixels.

The maximum number of edge pixels that can be paired between the two images is determined by obtaining the maximum cardinality matching (MCM) of the bipartite graph. This value is consistent independently of the order in which the data are evaluated, and provides completeness accuracy to the metric. Obtaining the MCM value is often time consuming, however, the bipartite graph generated from the point sets has the characteristic of being sparse. The MCM can be obtained in near-linear time for sparse graphs, using the algorithm proposed by Hopcroft and Karp [33]. The Recall metric is directly obtained from the computed MCM value and, in this work, the resulting metric is named the Maximum Cardinality Similarity Metric (MCSM).

In our experiments, a comparison between the MCSM and the PHD is made both in detection accuracy and in computing time. We found that our MCSM methodology obtains similar results than the PHD for typical image registration and template matching applications. Nevertheless, the results show that unlike the PHD, the MCSM methodology accurately reflects the

completeness of the evaluated shape under noise conditions. Regarding to the algorithmic complexity, we observe a significant decrease of the computer time required by our MCSM methodology compared with the time required by the PHD.

2.2 Methodology

In this section, our proposed MCSM methodology is described. The formulation for the different Hausdorff distances used in this study is also presented, along with a definition of the PHD to be used as a shape similarity metric.

2.2.1 Maximum cardinality similarity metric (MCSM)

This section describes the methodology used to calculate the MCSM. The MCSM is an adaptation of the evaluation methodology proposed by Correa-Tome *et al.* [18], to use their methodology as a similarity metric.

Let A be an edge image under test, and let B be an edge image used as a reference template. For each edge pixel in B, a corresponding neighborhood of radius t in A is inspected looking for edge pixels. At this point, any edge pixel in B that has no matching edge pixels in A is simply ignored. The edge pixels in B that have one or more matching edge pixels in A are used to build a bipartite graph.

The bipartite graph is built as follows. Let U and V be the two sets of nodes that constitute the graph. The set U consists of n nodes, corresponding to the n edge pixels from B that have one or more matchings in A. Meanwhile, the set V consists of m nodes that correspond to those edge pixels in A matched by one or more edge pixels from B. The edges of the bipartite graph denote the matching nodes from U to V. It is important to note that the nodes in V that can be paired with nodes in U are restricted by the tolerance radius t. Because of this, the bipartite graph generated is always sparse, a property that may be exploited to reduce the computing time.

Finding the MCM value of the bipartite graph is equivalent to find the number of edge pixels in B that have a corresponding pixel in A, i.e. the number of correctly detected edge pixels or true positives (TP). The algo-

rithm proposed by Hopcroft and Karp [33] is used in this study to obtain the MCM value. Belongie *et al.* [8] present a similar optimization method, using the algorithm proposed by Jonker and Volgenant [45]. Nevertheless, since the Hopcroft–Karp algorithm exhibits a near-linear computational complexity $O(n \log m)$ for sparse graphs, it is more convenient to use this algorithm instead.

The Hopcroft–Karp algorithm starts by obtaining an initial configuration for the bipartite graph. Each node $u_i \in U$ is inspected looking for a free node $v_j \in V$ linked to this node by an edge in the bipartite graph. If found, the node v_j is then associated to u_i. At this stage, the node selection method is not important. After the initialization stage, a tree is built for every node $u_k \in U$ that is left unpaired. The tree is built following the association chains that start at u_k and alternate between nodes from U and V. Each node, either in U or V, can be included only once in the tree built for u_k. When it is not possible to add more nodes to the tree, a breadth-first search is used to find the first free node in V (i.e. a node that has not been previously assigned to any node in U). The path in the tree, from the root to the free node, shows how the graph edges need to be reassigned to increment the overall number of node assignations by one. This is called an *augmenting path*. The process continues until all the unpaired nodes $u_k \in U$ are evaluated. At the end of this process, the number of paired nodes is maximum. This number is the MCM value.

The MCM value obtained by the described procedure equals to TP. The Recall measure R is used here as the maximum cardinality similarity metric (MCSM), whilst the Precision measure P is not used for this application. Recall is defined as shown in Equation 2.1,

$$R = \frac{TP}{TP + FN}, \tag{2.1}$$

where FN is the number of missing edge pixels or false negatives. This value may be obtained using $FN = n_e - TP$, where n_e is the total number of edge pixels in B. Nevertheless, in this study the normalized Recall metric (Equation 2.2) proposed by Correa-Tome *et al.* [18], is used instead of Equation 2.1,

$$R_n = \left(\frac{N}{N - TP}\right)\left(\frac{TP}{TP + FN} - \frac{TP}{N}\right), \tag{2.2}$$

having N as the total number of pixels in the image. The R_n measure is preferred because, whilst $R \in [TP/N, 1]$, $R_n \in [0, 1]$ and is proportional to the amount of points actually detected. Then, the normalized Recall metric R_n is taken instead of R as the MCSM proposed in this study.

2.2.2 Hausdorff distances

In this section, the different Hausdorff distances related to this study are described. The PHD is defined in function of a neighbourhood distance, analogous to the tolerance radius of the MCSM metric.

Given two sets of points, X and Y, the HD between these sets is calculated as shown in Equation 2.3,

$$H(X, Y) = \max\big(h(X, Y), h(Y, X)\big), \tag{2.3}$$

where the function h, shown in Equation 2.4, is the directed Hausdorff distance, calculated from a set X to a set Y,

$$h(X, Y) = \max_{x \in X} \min_{y \in Y} ||x - y||. \tag{2.4}$$

Let $\mathbf{K}$ represent the K-th element of a set, ranked from the shortest to the largest distance. Then, the partial Hausdorff distance is defined as shown in Equation 2.5,

$$h_K = h_K(X, Y) = \mathop{\mathbf{K}}_{y \in Y} \min_{x \in X} ||x - y||, \tag{2.5}$$

meaning that $h_0 \leq h_1 \leq \ldots \leq h_K \leq \ldots \leq h_q$. Having q as the number of elements in the set, then $h_q = h(X, Y)$.

Let d be a distance that defines the radius of a neighborhood for a given pixel in the binary map. Then, the K value used in Equation 2.5 is determined in function of d, whereas giving the number of K points below

the distance d. Taking this into account, the PHD may be used as a shape similarity metric by calculating the quotient K/q.

Both, the K/q quotient and the R_n value, represent the proportion of hits in relation to the number of points evaluated, and are bounded in the range $[0, 1]$. Moreover, the tolerance radius t from the MCSM metric and the neighborhood distance d from the PHD accomplish the same purpose, i.e. to tolerate small point localization errors. These similarities allow both metrics to be easily compared in a set of tests.

2.3 Tests and Results

In this section, the results from different tests are presented. The goal of these tests is to compare the properties of the MCSM and to determine if it may be used as a replacement of the PHD. For this, the MCSM requires to be robust against occlusion, partial edges, edge localization errors, and noise. Furthermore, if the MCSM is to be used as an alternative to the PHD, it must provide some advantages. Our main claim is that the MCSM can be computed in a near-linear time, surpassing that of the PHD. To ascertain this claim, a computing time test is discussed. Finally, a test is presented to compare the MCSM and the PHD for template matching applications.

2.3.1 Measurement of occluded and partial edges

One of the main advantages of the HD and PHD used in image processing applications is their robustness against partial or occluded edges. In this section, a test to compare the MCSM and the PHD in this regard is described.

Because both, the MCSM and the PHD are calculated for a neighborhood around a certain edge pixel of a given edge image, the shape of the object in the edge image is negligible [18]. Firstly, the shape of a circle was chosen to build a first set of test images, because for a human being it results easier to notice the proportion of completeness of a circle, compared to that of an irregular shape. In this test, partial circle shapes are compared against the complete circle shape (taken here as a template) using both, the MCSM and the PHD. The set of images for this test consists of circle arcs of length s, varying from $s = 0$ to $s = 2\pi r$, in steps of $\Delta s = r\theta = 2\pi r/c$, where r is the

radius of the circle, θ is the arc angle (in radians) and c is the number of images in the set. Figure 2.1a shows examples from this test set, consisting of a total of $c = 100$ images.

The results obtained from this test show no significant differences between the MCSM and PHD measures. Furthermore, both measures accurately indicate the completeness of the shape in the range $[0, 1]$. The Figure 2.2a shows the similarity measurements obtained using the MCSM and PHD methodologies with the image set containing partial circle edges (Figure 2.1a). The abscissa shows the actual edge proportion and the ordinate shows the detected edge completeness.

This test was also conducted on a set of 17 edge images obtained after applying the Canny edge detector [11] on real images taken from the Berkeley Segmentation Dataset and Benchmark [59], an image database consisting of 300 natural images [3]. For each edge image in the set, 51 different versions with subtracted regions of edges were obtained. The proportion of missing edges in each image version is controlled and known in advance. The proportion of missing edge blocks varies from zero (the full edge image) to all the edges (a plain image) in steps that remove around the 2% of the edges in each version. Figures 2.1b and 2.1c show example images from this test set and some of their variations.

The results show that both, the MCSM and the PHD methodologies obtain similar results, and are close to the expected completeness value. When using real images, the results from each one of the images in the set show no appreciable differences among them. Because of this, only the mean results are shown in Figure 2.2b.

The results from the tests presented in this section show that the MCSM and the PHD methodologies obtain similar results, for partial or occluded edges without noise. Because of this, both metrics may be used interchangeably in a given application. Moreover, under the aforementioned conditions, both metrics are proportional to the completeness of the shape.

2.3.2 Measurement of edges under noisy conditions

Another advantage of the HD and PHD when used in image processing applications is their robustness against noise. In this section, several edge images containing edge noise (i.e. missing edge pixels) are compared with a

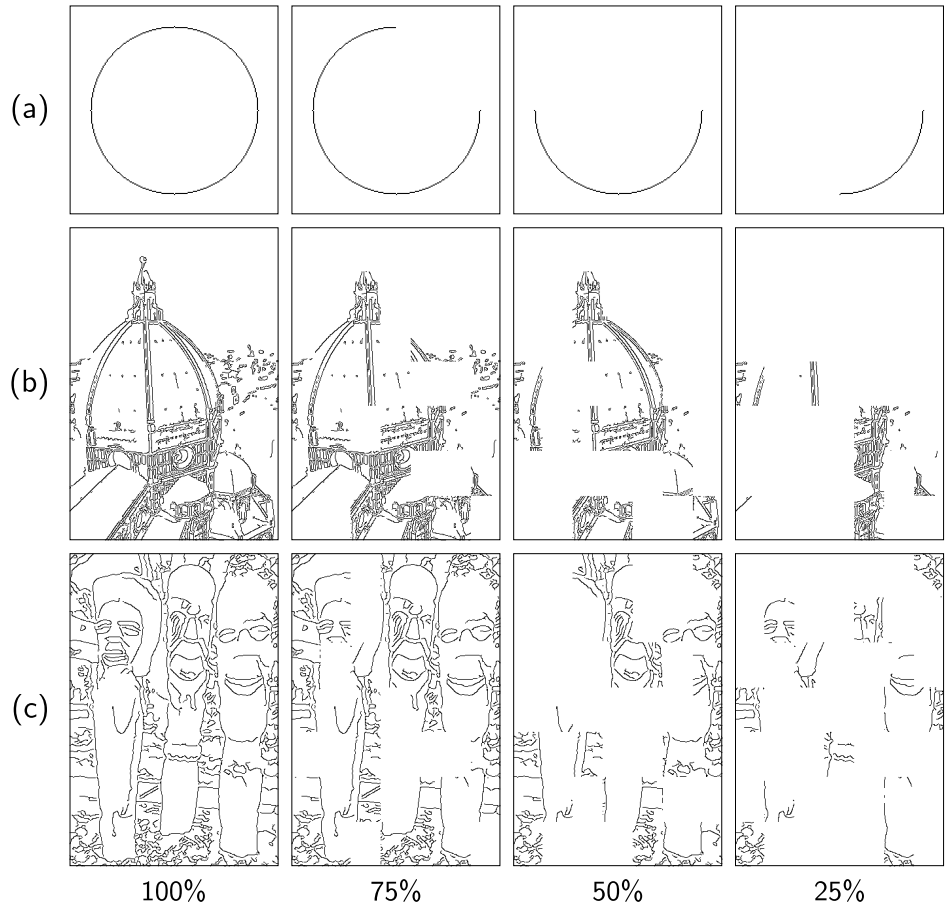


Figure 2.1. Edge image examples with partial occlusion from the circle test set (**a**), and from the test set of real images (**b**, **c**).

template, using both the MCSM and the PHD. The circle shape previously discussed was also used to build a set of images for this test. This set of edge images consists of a circle with a proportion p of missing edge pixels at randomly chosen locations along the circular shape. The proportion of missing edge pixels varies from $p = 0$ to $p = 1$, in steps of $\Delta p = 1/c$ between consecutive images. Figure 2.3a shows some examples from this test set of $c = 100$ images.

Figure 2.4a shows the similarity measure for the MCSM and the PHD, when edge pixels are randomly eliminated from the circle shape (Figure 2.3a). The abscissa shows the true edge proportion and the ordinate shows the

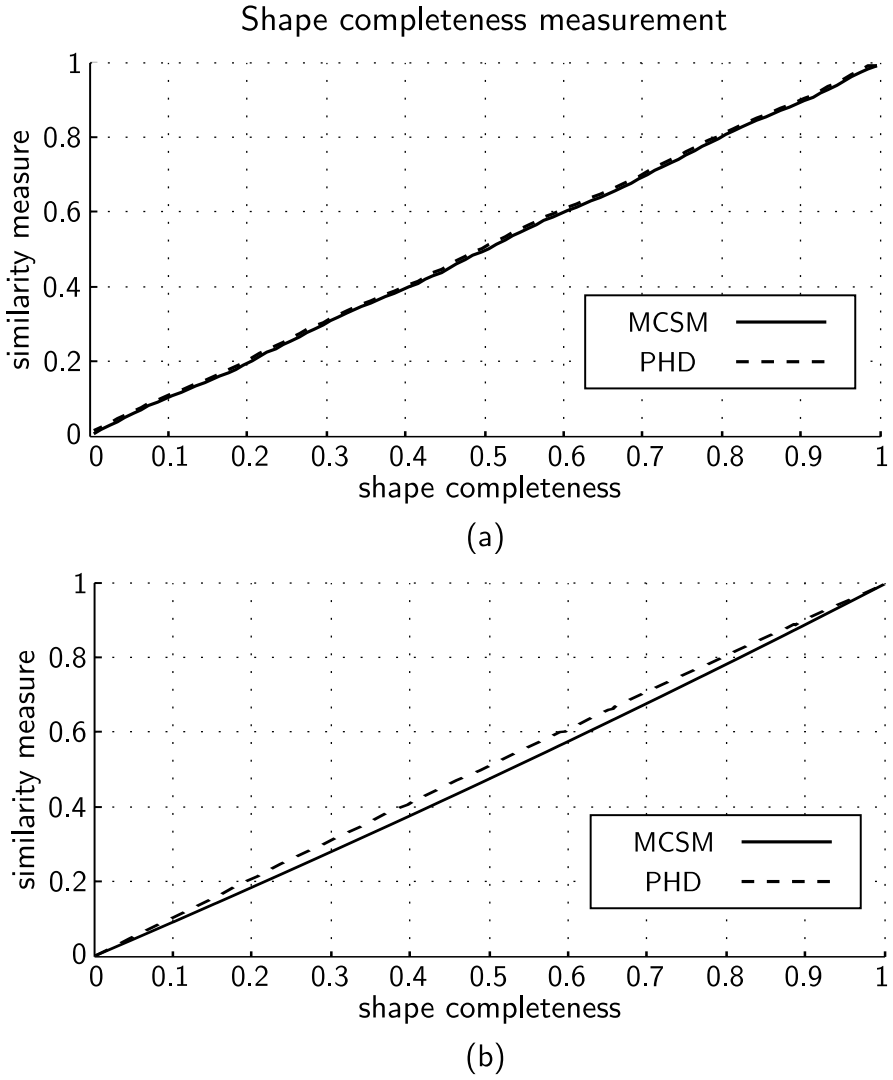


Figure 2.2. Comparison of the partial occlusion test results for circle shapes (**a**), and for mean results of real images (**b**).

detected completeness. The PHD plot shows a curve approximation to the mean results.

The results show a clear difference between the MCSM and PHD. Whereas the MCSM correctly detects edge completeness, the PHD shows a significant difference between the measured and the real completeness. This differ-

ence occurs because the PHD methodology is unresponsive to single or small groups of missing edge pixels. Nevertheless, the MCSM methodology performs a one-to-one assignation of points, being able to detect small changes.

This test was also conducted on real images. The same set of edge images described in Section 2.3.1 was used here. For each one of the 17 edge images in the set, 100 versions with different proportion of randomly eliminated edge pixels were generated. The proportion of eliminated edges varies from 0% (the original image) to 100% (a plain image). Figures 2.3b and 2.3c show some examples from this test set.

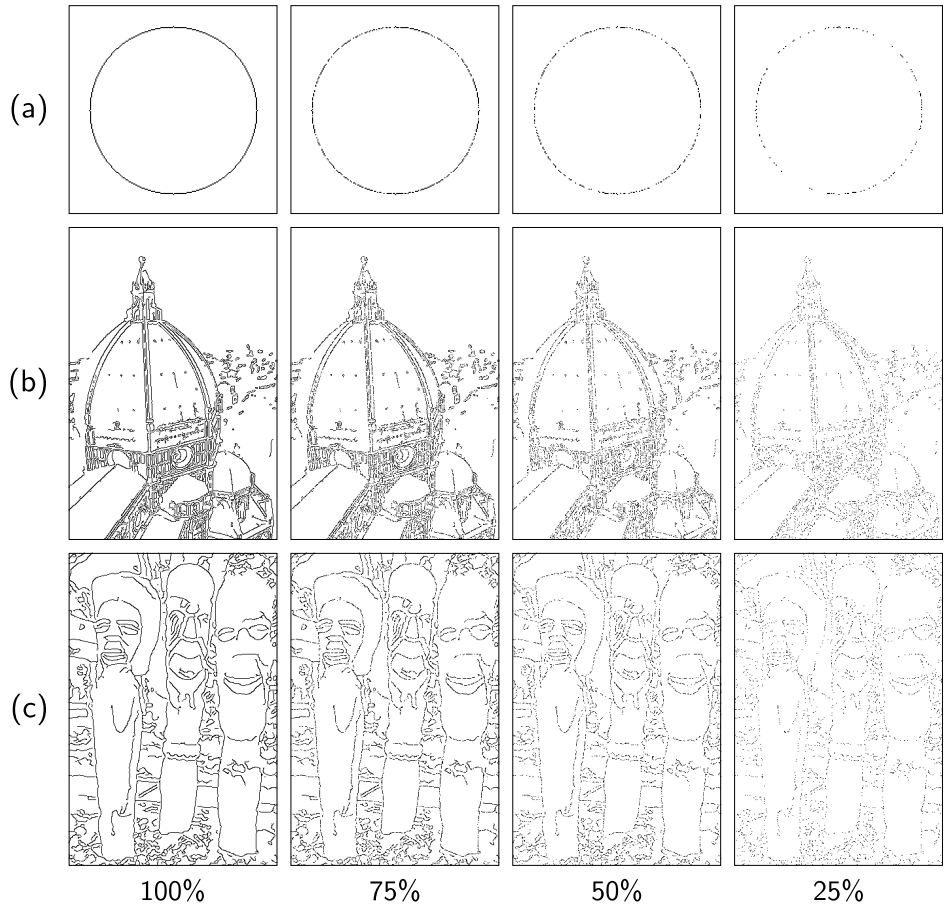


Figure 2.3. Edge image examples with noise from the circle test set (**a**), and from the test set of real images (**b**, **c**).

The results show that, whilst the MCSM obtains an accurate measure of the edge completeness, the PHD presents a significant loss of accuracy at determining the completeness of a shape under noise conditions. When using real images, the results from each one of the 17 real images show no appreciable differences among them. Because of this, only the mean results are shown in Figure 2.4b.

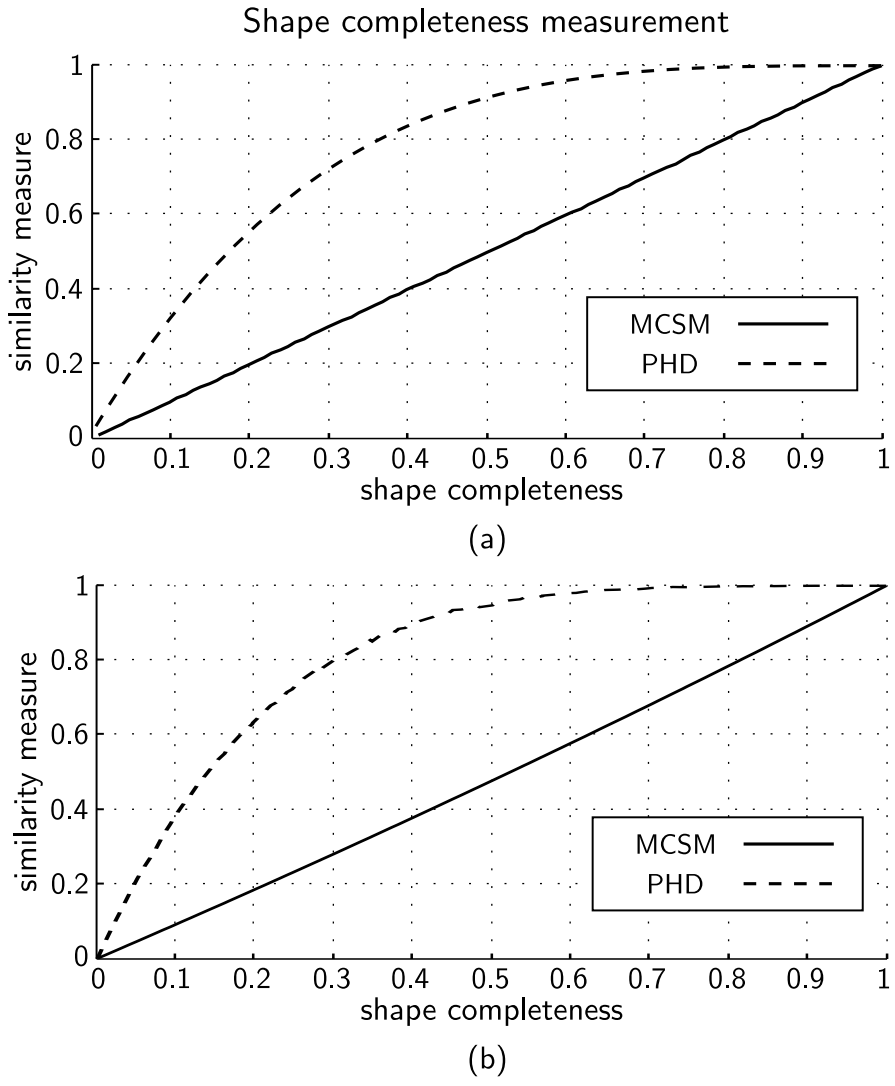


Figure 2.4. Comparison of the noise test results for circle shapes (**a**), and for mean results of real images (**b**).

The results from this test show that the PHD has certain degree of insensitivity to small missing edges. Nevertheless, the obtained MCSM value is sensitive to as few as one missing edge pixel. This property may be useful to accurately measure the proportion of noise in an edge image.

2.3.3 Distortion tolerance and distortion measurement

One of the main reasons to use PHD in edge images is because of its robustness against small edge localization errors. The proposed MCSM exhibits the same tolerance, and the conducted tests show similar results for both methodologies. The tolerance is defined in the algorithm as a radius whose distance is given in pixels. As defined in Section 2.2, the tolerance is t for MCSM, and d for PHD.

Figure 2.5 shows some examples from a set of images of a grid pattern with different degrees of distortion. The images are from the PSU Near-Regular Texture Database [55].

The image without distortion (Figure 2.5a) is used as the template, and is compared with the distorted images using different values of t and d. The results are shown in Figure 2.6a for the MCSM, and in Figure 2.6b for the PHD.

Even though the measures obtained using the MCSM and the PHD are not equivalent, the results show a similar behavior. If the variable t or d is increased, the tolerance against distortion augments in a similar proportion.

Instead of using the tolerance variables t and d to define a distortion tolerance threshold, they may be used the other way around to detect the degree of distortion in a given image. The results from Figures 2.6a and 2.6b show that the image **a** in Figure 2.5 has no distortion, **b** has a distortion radius of about 1 pixel, whereas **c** is about 2 pixels, **d** about 3 pixels, and **e** about 4 pixels. The radius of distortion for the image **f** is not clear from the figures, nevertheless the experimental results show that the radius is about 5 pixels. Furthermore, the results show that the distortion radius obtained using the MCSM (t) or the PHD (d) are equivalent.

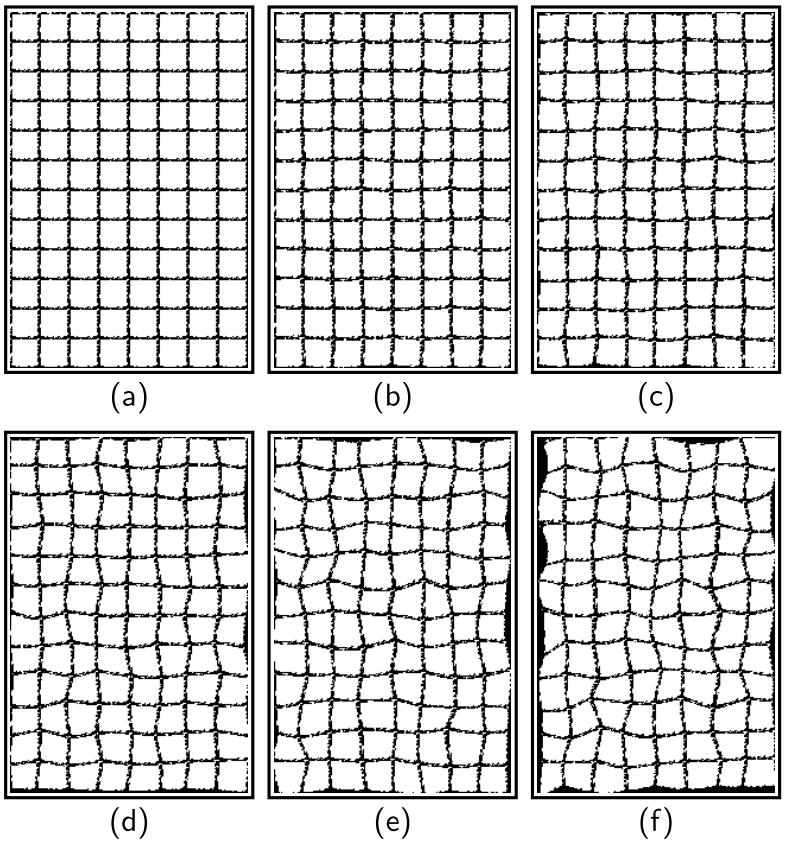


Figure 2.5. Example images showing a pattern with different degrees of distortion, from no distortion (**a**) to a large distortion (**f**).

2.3.4 Processing time

Regarding to the algorithmic complexity of the MCSM and the PHD methods, a performance test is described in this section. This test set consists of images with a different number of points to be evaluated using both, the proposed MCSM methodology and the PHD. The test is made to determine the time required by each algorithm when calculating the similarity measure between two sets of n points (i.e. edge pixels), for $0 < n \leq 50 \times 10^3$. The actual location of the points is irrelevant for the test.

Figure 2.7 shows the processing time required by each methodology as the number of points increases. An Intel Core i3–2120 computer at 3.30 GHz and 4 GB of RAM was used to obtain this graph. Even though the time

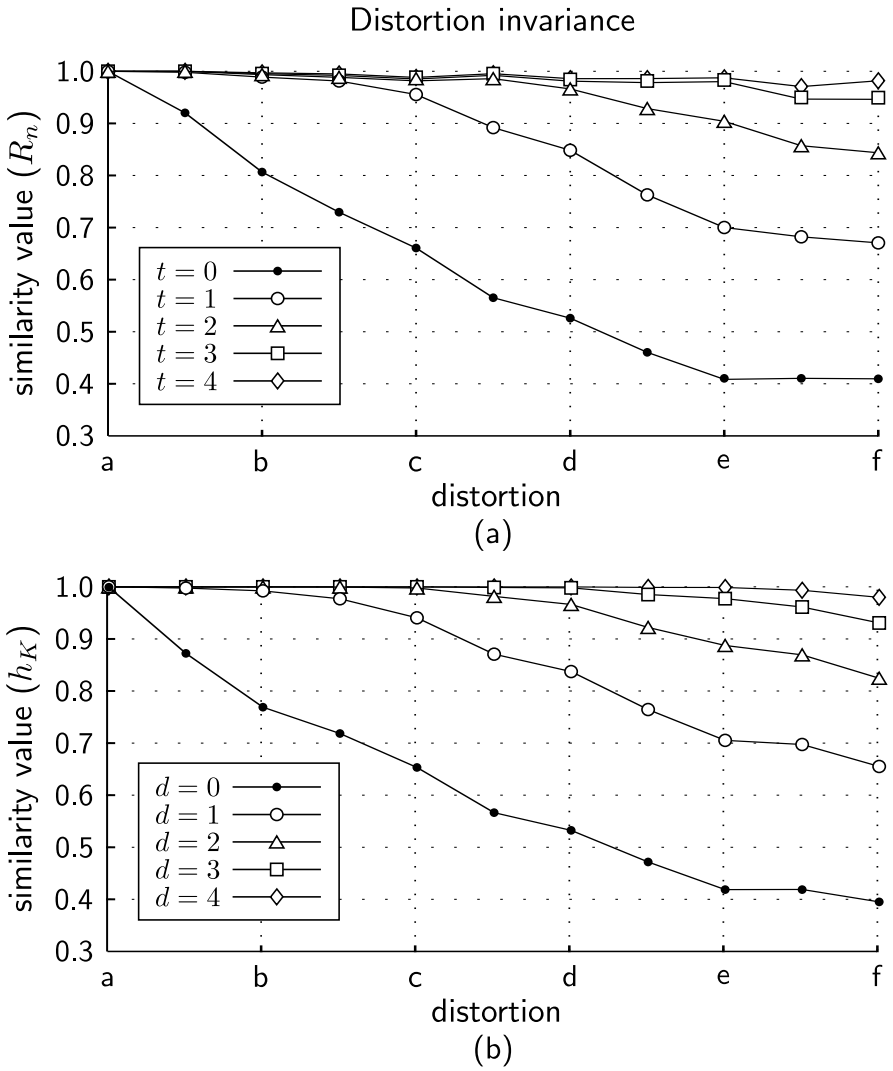


Figure 2.6. Distortion tolerance test results for the MCSM (**a**), and the PDF evaluation (**b**), using $t, d = 0, 1, 2, 3, 4$.

spent is dependent of the hardware used, the algorithmic complexity growth shown in the Figure 2.7 is independent of it.

The results show a better time performance for the MCSM compared with the PHD. For sets with few points, the time required to compute each metric is close. Nevertheless, as the number of points increases, the time required for the PHD methodology increases exponentially $O(n^2)$, while our methodology

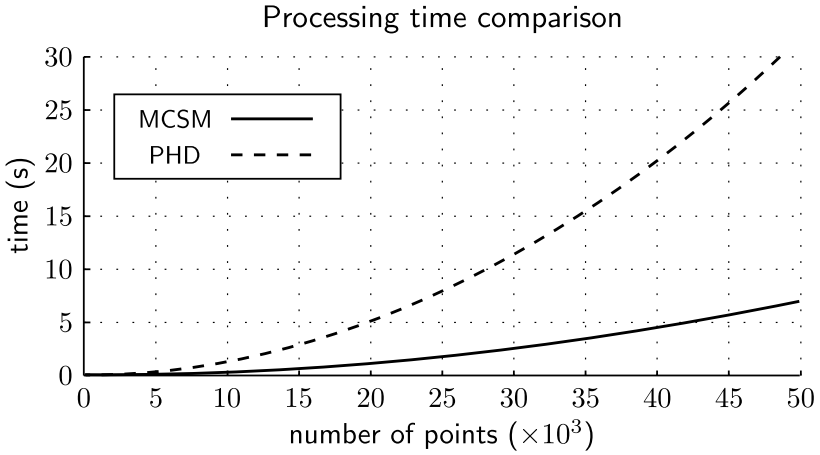


Figure 2.7. Processing time comparison between the MCSM and the PHD.

remains near-linear $O(n \log n)$. This feature makes the proposed MCSM suitable as a replacement of the PHD for demanding real-time applications.

To give an example, the MCSM methodology requires on average 20.92 ms to calculate R_n for an image of 481×321 pixels (from the real image test set), containing 13 720 edge pixels, on an Intel Core i3–2120 computer at 3.30 GHz and 4 GB of RAM. Nevertheless, the PHD methodology requires 2.44 s to calculate h_K for the same image on the same off-the-shelf computer.

2.3.5 Template matching

Template matching is a common use for the HD and PHD in edge image processing applications. As mentioned in Section 2.1, the HD is used in such applications because it is robust against the errors commonly found in edge images. To use our MCSM methodology as a HD or PHD replacement, the results obtained for template matching tasks should be similar to those obtained by the HD.

In a series of tests, the MCSM and PHD methods are used as similarity metrics for template matching. The images used for testing are also from the Berkeley Segmentation Dataset and Benchmark [3]. Each one of the 300 natural images in the dataset has from five to ten segmentations made by humans. The required templates were obtained from those human-made segmentations.

For each image in the test set, the edge detection algorithm proposed by Canny [11] is applied to a gray-scale version of the natural image. The resulting edge image is registered, comparing the template at each pixel location using both, MCSM and PHD. The maximum value obtained is taken as the location of the template in the image. The tests show similar results for both methodologies, with variations of one pixel, in average.

Figures 2.8 and 2.9 show examples from template matching tests. The top–left image of each figure (**a**) shows the template to look for and the natural image to look into. The top–right image (**b**) shows the image obtained using the Canny edge detector. The bottom–left image (**c**) shows the template location found using MCSM and the bottom–right image (**d**) shows the template location found using PHD.

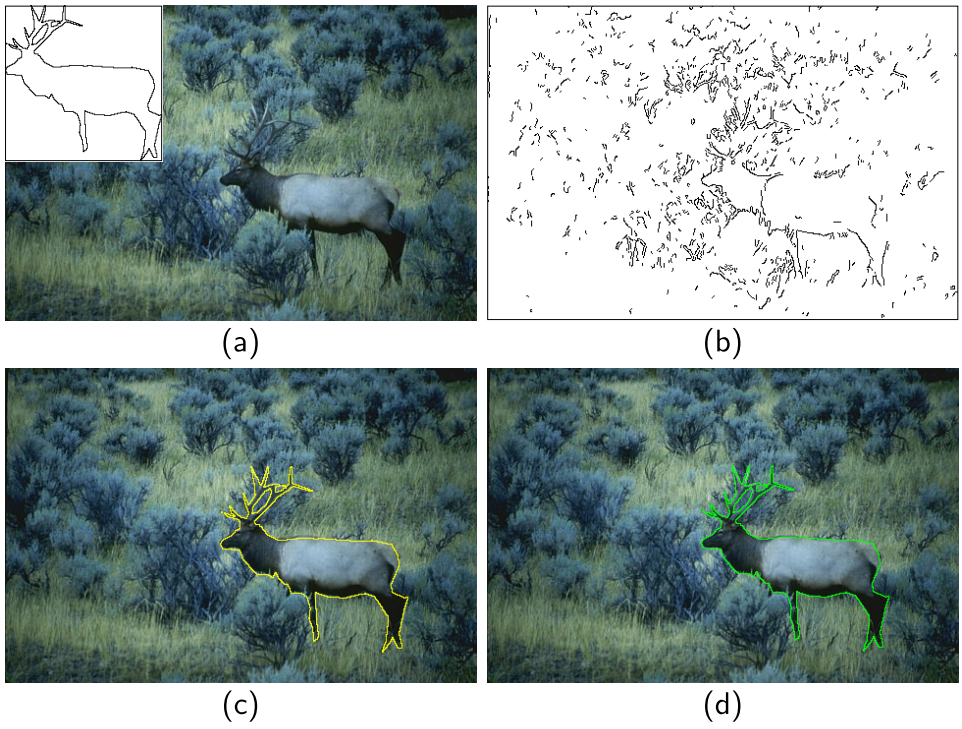


Figure 2.8. Template matching test showing the original and template images (**a**), the edge image (**b**), the MCSM detection (**c**), and the PHD detection (**d**).

Even though small discrepancies are expected from the results, the experiments show that MCSM may be actually used as a replacement of the PHD

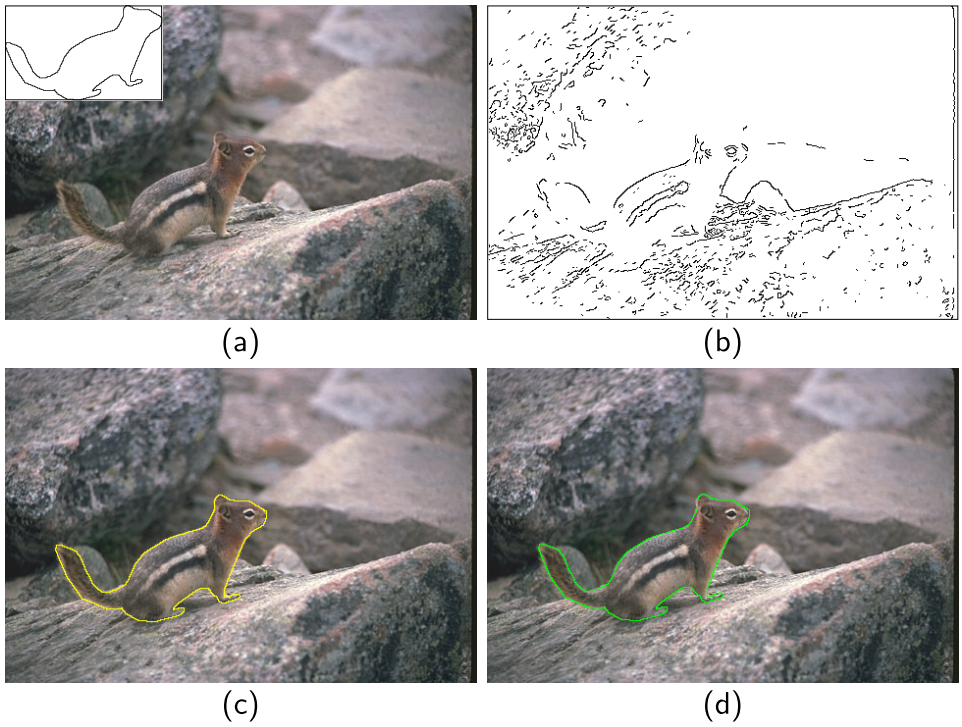


Figure 2.9. Another template matching test showing the original and template images (**a**), the edge image (**b**), the MCSM detection (**c**), and the PHD detection (**d**).

to find the localization of templates in edge images. The MCSM methodology is also robust against the same problems found in edge images. Moreover, the low algorithmic complexity of the MCSM offers a significant advantage over the PHD in execution, for applications where the processing time is a crucial factor.

Chapter 3

Integral split and merge segmentation

In this chapter, a fast image segmentation methodology is presented. This methodology uses a split-and-merge segmentation approach, combined with integral images to reduce the computing time required by the method. A general overview of the proposed segmentation methodology is presented in the following Section 3.1. Then, the details about the implementation of this methodology are discussed in Section 3.2. In order to evaluate the performance of our segmentation methodology, a series of tests were performed. These tests and the results obtained from them are discussed in Section 3.3.

3.1 Introduction

The segmentation of images is a common procedure in image analysis applications. This procedure is the first step in many processes meant to extract useful information from scenes. It consists in partitioning the image into different disjoint regions, that are homogeneous for one or more features computed from the image. Those regions are assumed to correspond to meaningful parts of the objects in the scene, regions that are easier to analyze. In their survey, Vantaram and Saber [84] classify the segmentation methodologies in three main classes: spatially blind, spatially guided, and miscellaneous methods.

As their name suggest, the segmentation of images using the spatially blind methods is not guided by the spatial relationships of the pixels in the image. These methods include approaches using clustering or histogram thresholding. The spatially blind methods have the advantage of being easy to implement, not requiring any a priori information. However, the number of clusters to use for clustering is required to be determined in advance. This has proven to be a challenging task. Moreover, the histogram thresholding methods have problems with low-contrast images, and are difficult to use in color images.

The segmentation of images using spatially guided methods, is directed by the relationships between pixels in an image region. These methods group together regions that are homogeneous regarding to a given image feature. The spatially guided methods are further subdivided into region-based, energy-based, and contour-based methods. The region-based methods include procedures such as the growing, the splitting and the merging of regions. Some algorithms from this category include the J-segmentation algorithm, proposed by Deng and Manjunath [21], the gradient segmentation algorithm, proposed by Garcia Ugarriza *et al.* [79], and a multiresolution extension of the gradient segmentation algorithm, proposed by Vantaram and Saber [85], among others. Regarding to the energy-based methods, they attempt to minimize cost functions that model regions in the image. Those cost functions may be contour or region based functions. The regions covered by the functions evolve until a given energy model is minimized. Some algorithms in this category include the active contours (a.k.a. snakes), first proposed by Kass *et al.* [46] and variants, such as the fast active contours algorithm proposed by Chan and Vese [13] or the active contours without edges, proposed by Vantaram and Saber [83]. Lastly, the contour-based methods consist of different variants of the watershed algorithm. This algorithm considers a gray-scale image as a topographic relief, where the intensity of the pixels determine the corresponding height of that particular zone. The relief is then flooded in a simulation, the water flows to local minima and forms basins, corresponding to different regions in the image. Some examples of these methods include the work of Gao *et al.* [25], where watersheds are used to segment color images, the study of Hill *et al.* [31], that uses a texture gradient to partition textured regions using watershed, and the method by Kim and Kim [47], where a multiresolution watershed segmentation using wavelets is presented.

In this study, we propose a methodology called Integral Split and Merge (ISM) segmentation. This methodology is a region-based segmentation algorithm, where the split-and-merge segmentation, and an image representation called *integral image* are combined, to achieve two main goals: to obtain acceptable segmentation outcomes, and to attain a computational complexity low enough to use the method in real-time applications (i.e. 15 fps or more). The split-and-merge segmentation was proposed by Horowitz and Pavlidis [35, 34] back in 1974. It consists in recursively partitioning an image into homogeneous parts, and then merging those parts into bigger homogeneous regions. Our ISM methodology uses the integral images to improve the computing time performance. The integral image representation was first used in the object recognition field by Viola and Jones [87], to achieve a fast feature evaluation for real-time face detection applications.

The proposed ISM method works over intensity images, where the pixels in the homogeneous regions have similar intensity values. The method consists of two main parts: the splitting and the merging steps. The splitting process divides the image into homogeneous regions. This process is performed in a single step, using a quad-tree search. The image is partitioned into regions (i.e. quads) that are evaluated for homogeneity. Because determining the homogeneity of a region is computationally expensive, we use the integral images to improve the time performance. On the other hand, the merging process combines the homogeneous regions obtained from the splitting process, into bigger areas. We propose a merge criterion that is performed in an efficient time, and exhibits three main properties. Firstly, it assigns different labels to spatially disconnected regions, independently of their similitude. This is equivalent to perform a connected-component analysis. Secondly, our method is able to follow gradients in image areas showing this property, and to group them into single regions. This process is similar to that performed by region-growing methods. Lastly, our merging procedure is able to automatically determine the number of disjoint regions in the segmentation.

Additionally to the ISM split and merge processes, a small-region elimination procedure is presented. This procedure is used to improve the visual appeal of image segmentations, if required. This step first removes the small regions obtained from the ISM segmentation. Then, the removed pixels are reassigned to bigger regions, using a region growing process. The tests performed show that the small regions have no significant impact in the overall

evaluation of a segmentation, and therefore this step may be omitted, if necessary.

The ISM methodology was tested to ascertain its performance in segmentation accuracy and computing time. The tests were performed on a collection of natural images, where texture patterns are abundant. For this kind of images, it is better to use texture information for the segmentation. Different image features (e.g. texture descriptors) may be used to obtain an intensity map that shows different homogeneity properties. In this work, we use a single texture descriptor: the standard deviation (SD) map. We calculate the SD map in a preprocessing stage. This preprocessing transforms texture features into intensity values, that are suitable to be used by our ISM methodology. The SD image describes texture as intensity levels, where each pixel in the SD image is the standard deviation of the intensity values in a neighborhood, centered at the given pixel position in the original image. The ISM methodology used on SD images produces segmentations where the regions are uniform, regarding to the standard deviation of their values. Even though there may be better texture descriptors than the SD images, their study is beyond the scope of this work. Here, the SD images are used only to present the ISM methodology. However, despite its simplicity, the SD images have obtained good results in segmentation applications, as in the work of Lizarraga-Morales *et al.* [56].

The results obtained were evaluated using the Normalized Probabilistic Random (NPR) index. The NPR index is a robust methodology, developed by Unnikrishnan *et al.* [81, 80] to evaluate the quality of image segmentations. The image segmentations obtained from a given algorithm are normally compared against one or more human-made references. To this end, we use the Berkeley Segmentation Dataset and Benchmark (BSDS) [59], a collection of 300 natural images and several human-made segmentation references for each image. The BSDS has been previously used along with the NPR index. An example is the work of Pantofaru and Hebert [68], where the BSDS and the NPR index are used to evaluate image segmentations, obtained using mean-shift, the efficient graph-based segmentation proposed by Felzenszwalb and Huttenlocher [23], and an hybrid method that combines both, in order to determine if the hybrid method improves the segmentation quality. The NPR index and the BSDS were also used by Vantaram and Saber [84] to evaluate the segmentation quality of eleven state-of-the-art algorithms. In this study, we compare in equal terms the results obtained using our ISM

methodology and the results obtained from the eleven algorithms, reported by Vantaram and Saber [84].

The results obtained from the tests performed show that our ISM methodology obtains a similar segmentation quality than the other state-of-the-art algorithms, used for comparison purposes. However, whilst the other methods combine both color and texture features, our ISM methodology only requires a single texture feature (i.e. the SD image) to achieve similar results. Furthermore, our tests show that, using integral images to calculate the statistics required by the split and merge segmentation, improves the execution time of the method. We have found that the ISM methodology is efficiently executed in a piecewise linear time. These contributions may be advantageous for real time segmentation applications.

3.2 Methodology

In this section, the proposed split and merge segmentation methodology is described. The section starts discussing the integral images, adapted to this particular application. Then, the splitting and merging procedures of our methodology are described, along with their implementation details. Additionally, two more procedures that may be used after the segmentation are discussed. The first one is a procedure that declassifies small regions, and the second one is a region growing procedure, that assigns the declassified pixels to the remaining classes.

3.2.1 Integral images

The proposed ISM methodology makes use of the integral images to improve the time required to obtain an image segmentation. This procedure is essential, because it allows the ISM method to be executed in real time.

The integral images [87] (a.k.a. summed area tables) are used to efficiently compute sums of pixel intensities in square regions of an image. The time required to compute such sums is constant, and does not depend on the region size. Moreover, the integral image may be computed in linear time.

This section describes the method to compute the integral image, and to obtain the area sums. These sums are used to compute the mean intensity,

and the variance of the intensities in an image region. These statistics are used by our splitting process to improve the execution time.

The summed area table

The integral image, or summed area table, is obtained from the original image, and has the same dimensions. This image representation is used to efficiently compute the sum of intensities in a square area of the image.

Let I be an image consisting of $W \times H$ pixels. Also, consider the pixel $(0, 0)$ to be at the top–left corner of the image, and the pixel $(W-1, H-1)$ at the bottom–right corner. The pixel at column i and row j in I is denoted by $I(i, j)$. The integral image S consisting of $W \times H$ cells, is calculated as shown in Equation 3.1. The cell at column i and row j in S is denoted by $S(i, j)$. Each cell $S(i, j)$ is the sum of all the intensities above and to the left of its corresponding image pixel $I(i, j)$, including itself. The computation of $S(i, j)$ may be optimized using the data already computed, as shown in Equation 3.2.

$$S(i,j) = \sum_{\substack{i' \leq i \\ j' \leq j}} I(i', j') \tag{3.1}$$

$$\begin{aligned} S(i,j) = {} & I(i,j) + S(i-1,j) + S(i,j-1) \\ & - S(i-1,j-1) \end{aligned} \tag{3.2}$$

The function $S(i, j) = 0$ for the values $i, j < 0$. Notice that using Equation 3.2, the integral image is obtained in linear time $O(n)$.

Sum of intensities in a region

The integral image may be used to compute the sum of pixel intensities of a square region of arbitrary size. The time required to compute this sum is constant, and it is independent of the size of the area involved.

Consider an square area defined by $a = (x_0, y_0)$, $b = (x_1, y_0)$, $c = (x_0, y_1)$ and $d = (x_1, y_1)$, where x_0, y_0, x_1, and y_1 are integer numbers representing

the coordinates of the region in the image (see Figure 3.1). Then, the sum of the pixel intensities (s) in the region is obtained using Equation 3.3.

$$\begin{aligned} s = {} & S(x_1, y_1) - S(x_0 - 1, y_1) - S(x_1, y_0 - 1) \\ & + S(x_0 - 1, y_0 - 1) \end{aligned} \tag{3.3}$$

The function $S(i, j) = 0$ for the values $i, j < 0$. Notice that the computation of the Equation 3.3 is obtained in constant time $O(1)$.

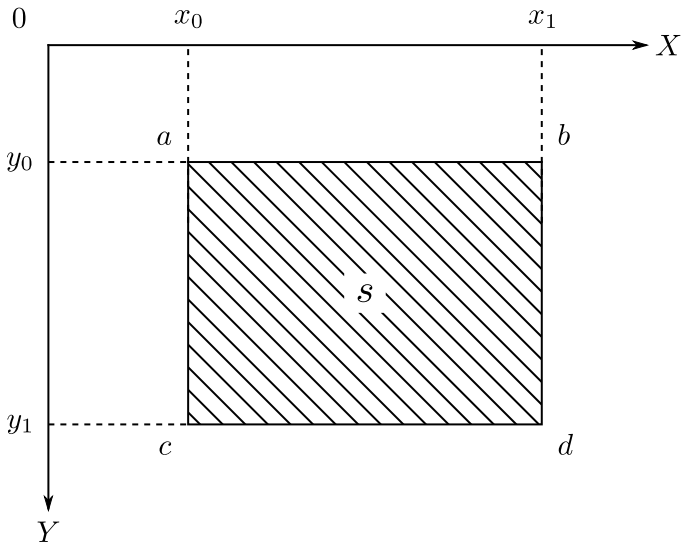


Figure 3.1. Area to sum in a summed area table.

Fast computation of statistics

In their article, Bradley and Roth [9] use the integral image to compute the mean intensity of a rectangular region of an image. In this work, a similar approach is used to calculate the mean. Additionally, we also calculate the variance of the pixel intensities, in square regions of arbitrary size, by using statistical moments.

First, consider the Equation 3.4, where s_r is the sum of all the intensity values, raised to the power of r, in a region defined by x_0, y_0, x_1, and y_1. The

mean μ and variance σ^2 of the region data may be computed in function of s_r, using Equations 3.5 and 3.6, respectively.

$$s_r = \sum_{j=y_0}^{y_1} \sum_{i=x_0}^{x_1} I^{(r)}(i,j) \tag{3.4}$$

$$\mu = \frac{s_1}{N} \tag{3.5}$$

$$\sigma^2 = \frac{s_2}{N} - \left(\frac{s_1}{N}\right)^2 \tag{3.6}$$

where N is the number of pixels in the defined region. Notice that the sum s_r may be efficiently computed using integral images. First, the summed area table of an image with intensities raised to the power of r is calculated as shown in Equation 3.7.

$$\begin{aligned} S_r(i,j) = I^{(r)}(i,j) + S_r(i-1,j) + S_r(i,j-1) \\ - S_r(i-1,j-1) \end{aligned} \tag{3.7}$$

This operation is also performed in linear time $O(n)$. Then, the sum of intensities s_r is calculated using the summed area table S_r, as shown in Equation 3.8.

$$\begin{aligned} s_r = S_r(x_1,y_1) - S_r(x_0-1,y_1) - S_r(x_1,y_0-1) \\ + S_r(x_0-1,y_0-1) \end{aligned} \tag{3.8}$$

The function $S_r(i,j) = 0$ for the values $i,j < 0$. The computation of Equation 3.8 is also performed in constant time $O(1)$.

Optimization details

The integral images discussed in this section are intended to be used for image segmentation, and because the intensity values of the image pixels are integer (in $[0,255]$), an integer integral image may be used. This data representation may be advantageous, because the integer arithmetic is faster than

the floating point arithmetic. Thus, the segmentation methodology discussed in this work may be fully implemented using only integer arithmetic.

Even though the mean and variance values obtained from pixel intensities may be real numbers, an integer approximation is acceptable. Therefore, the division operations required to calculate the mean and the variance may be performed using integer arithmetic alone. Moreover, the division operations required by the splitting process described in Section 3.2.3, may be replaced by bit shifts.

3.2.2 Image segmentation

The image segmentation is formally defined as follows [67]. Let F be the set of all the pixels in a given image. Then, the image segmentation is the partitioning of F into a set of connected regions $\{Q_1, Q_2, \ldots, Q_k\}$ that satisfy the Equations 3.9 and 3.10.

$$\bigcup_{i=1}^{k} Q_i = F \tag{3.9}$$

$$Q_i \cap Q_j = \emptyset, \quad i \neq j \tag{3.10}$$

Also, let $P(\,)$ be a homogeneity predicate, defined on groups of connected pixels. Then, $P(Q_i) = true$ for all segmented regions Q_i, and $P(Q_i \cup Q_j) = false$ for $i \neq j$. A region is homogeneous when the values of its pixels are close to each other, as described in the next section.

3.2.3 Splitting process

The first step of the proposed segmentation methodology is a splitting process. This process divides the image into regions of homogeneous intensity. First, the image is divided in quads, and each quad is tested for homogeneity. If the region is not homogeneous, that region is further subdivided in quads, and the process is repeated until the homogeneity condition is reached, or until a stop condition is met.

A region homogeneous in intensity is a region in the image that has the same intensity for all its pixels. In practice, this situation rarely occurs.

Small differences between pixel intensities are expected in perceptually homogeneous regions. Therefore, a certain tolerance to intensity variation may be allowed for homogeneous regions. The maximum intensity variation allowed is determined by a variance threshold (σ_T^2), that is required to be calculated for each image. To that end, an automatic method to detect the variance threshold is also discussed in this section.

Quad-tree building

Before starting the subdivision process, the original image should be slightly modified in order to make the subdivision easier. First, it results more convenient if the image is square, with sides that are powers of two. This assures the image to be divisible at every level of the tree, and the divided regions to be always powers of two.

If the original image does not meet the aforementioned size requirements, the image canvas is enlarged, adding empty pixels to the right and to the bottom of the image, until obtaining an image of $2^\ell \times 2^\ell$ pixels, where ℓ is a positive integer value such that $2^{\ell-1} < \max(W, H) \leq 2^\ell$. Additionally, it is recommended to assign a special value to the empty pixels. This special value has the property of being always different from any other pixel intensity, regardless of the variance threshold (σ_T^2) used. This way, the empty pixels can only be grouped together, and the space added by the canvas enlargement does not interfere with the segmentation process.

Subdivision process

The subdivision process starts with the whole image, after the canvas enlargement step. Being the side length of the image equals to 2^ℓ, consider the number of levels in the quad three to be ℓ.

The homogeneity of the level ℓ of the image is determined by obtaining the variance of the region (in this case the variance of the whole image), and comparing it against the variance threshold σ_T^2. The variance of the region is efficiently computed by our ISM methodology using integral images, applying the Equation 3.6. Notice that, for this equation, $N = (2^\ell)^2 = 2^{2\ell}$, a number that is also a power of two. Therefore, the variance calculation requires no division operations, a bit shift may be used instead.

To decide if the region (in this case the whole image) is homogeneous, its variance should be below the variance threshold level σ_T^2. Otherwise, the region is considered non-homogeneous, and it is subdivided into four square regions of side $2^{\ell-1}$.

The process is repeated for each one of the four regions. If a region results to be non-homogeneous, the region is further subdivided. The process is repeated until the homogeneity condition is met, or the level $\ell = 0$ is reached. At this point, all the image is divided into different regions that are homogeneous in intensity.

Additionally, the process may be initiated or finalized at arbitrary levels. For example, the splitting process may start at level ℓ. However, until the level $\ell - \alpha$ is reached, all the regions are considered non-homogeneous. This consideration is used to avoid segmentation errors produced by big regions, containing small areas that should be assigned to a different class. If the mean intensity of such region is below σ_T^2, the error may not be detected. However, as mentioned by Ojala and Pietikäinen [65], smaller initial regions are easily merged together again by the merging process. In this work, we use a maximum region of $2^6 \times 2^6$ pixels to avoid this kind of errors. On the other hand, the process may be also finalized before the level $\ell = 0$ is reached. This may speed up the subdivision process, because the lower levels in the quad three have more regions to process than the upper levels. However, doing this increases the error in the detection of the boundaries between regions, reducing the quality of the segmentation. For this reason, in this work we use $\ell = 0$.

In our implementation we use a data structure called *node*, to store the information of a given homogeneous region. This information is used to further reduce the operations required by the splitting process. The nodes are stored in a vector as soon as the homogeneous regions are found. Because the quad tree is explored using a breadth-first search, the resulting vector is sorted. This is required by the merging process.

Automatic variance threshold selection

It results inconvenient to define a constant variance threshold σ_T^2 for all images, because that optimum variance threshold is dependent on the image under segmentation. In their article, Garcia Ugarriza *et al.* [79] propose an

automatic approach, called *adaptive gradient thresholds*, that initializes the seeds of their region growing segmentation methodology. We adapted this methodology to determine the best value of σ_T^2 from a gradient map obtained from a given image. This process is fast and should be performed for every image before starting the segmentation. The methodology adapted from the work of Garcia Ugarriza *et al.* [79] is discussed in this section.

An estimation of σ_T^2 may be obtained using the next procedure. First, a gradient image is obtained from the original image, assumed to have only one channel. The image consists of $K = W \cdot H$ pixels of intensity $U = \{u_1, u_2, u_3, \ldots, u_K\}$, whilst the gradient image is made of $K = W \cdot H$ values $G = \{g_1, g_2, g_3, \ldots, g_K\}$, for an image of $W \times H$ pixels, where each k index from the u_k and g_k values, corresponds to the image position $k = Wj + i$.

The value for each pixel in the gradient image G is defined as $g = \sqrt{\lambda}$, where λ is obtained using the Equation 3.11 presented by Garcia Ugarriza *et al.* [79].

$$\lambda = \tfrac{1}{2}\left(q + h + \sqrt{\big(q+h\big)^2 - 4\big(qh - t^2\big)}\right). \tag{3.11}$$

The variables q, t and h are defined in Equations 3.12, 3.13 and 3.14, respectively. The variables x and y are the spatial coordinates of the image.

$$q = \left(\frac{du}{dx}\right)^2 \tag{3.12}$$

$$t = \left(\frac{du}{dx} \cdot \frac{du}{dy}\right) \tag{3.13}$$

$$h = \left(\frac{du}{dy}\right)^2 \tag{3.14}$$

The differential terms of Equations 3.12 to 3.14 are calculated in function of image intensity values as shown in Equations 3.15 and 3.16.

$$\frac{du}{dx} = I(i,j) - I(i-1,j) \tag{3.15}$$

$$\frac{du}{dy} = I(i,j) - I(i,j-1) \tag{3.16}$$

After obtaining the gradient image G, its histogram is calculated, and the threshold variance σ_T^2 is obtained from this histogram. It is required to find a threshold p in the histogram where a given percentile is reached. For convenience, the threshold p is normalized in $[0, 1]$ and not in $[0, 100]$ as is the percentile, but reflect the same proportion. The threshold p signals the maximum intensity deviation that should be considered as homogeneous, which is equivalent to σ_T (see Figure 3.2). Even though p is also a threshold value, its behavior is different from σ_T^2. Whilst σ_T^2 changes from image to image, p is a constant that may be experimentally determined. Garcia Ugarriza *et al.* [79] reported that the optimum value for their adaptive gradient threshold is near 0.8. We experimentally obtained a similar value for the gradient threshold $p = 0.82$, the protocol used for this experiment is discussed in Section 3.3.3.

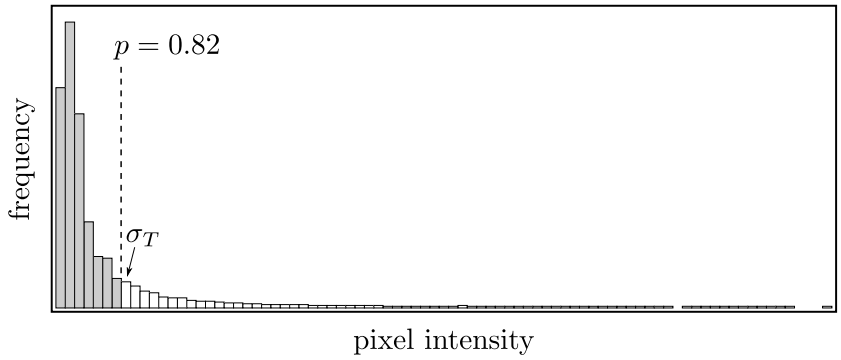


Figure 3.2. Histogram of the gradient image showing the p threshold level.

3.2.4 Merging process

The ISM merging process combines the homogeneous regions, obtained from the splitting process, into bigger regions that are assigned to a class. The regions in a class should be homogeneous in intensity, and be spatially connected. The ISM merging methodology proposed here has three main characteristics: the number of classes is automatically determined for each image, smooth gradients are considered as homogeneous regions, and different classes are assigned for disconnected regions. The details of our ISM merging algorithm are discussed next.

Conditions and cases

The homogeneous regions identified by the splitting process (nodes) may be merged together if they fulfill the next merging conditions. Firstly, an unclassified region is always merged to a classified region. Secondly, the unclassified region should be equal in size or smaller than the region to which it is going to be merged to. Thirdly, the difference of the mean intensities between the classified and the unclassified regions should not be greater than σ_T. Lastly, the regions must be adjacent, in the vertical and horizontal directions only.

If a region is not classified and has no classified neighbors that fulfill the merging conditions, a new class is created for that region. Additionally, it may happen for an unclassified region to have more than one adjacent region that fulfills the aforementioned conditions. There may be up to four suitable neighbors for each unclassified region, because of the second merging condition. If those regions share the same class label, the assignment of the region to that class is straightforward. However, if the adjacent regions have different class labels, those labels need to be reassigned to a single class.

Class assignment

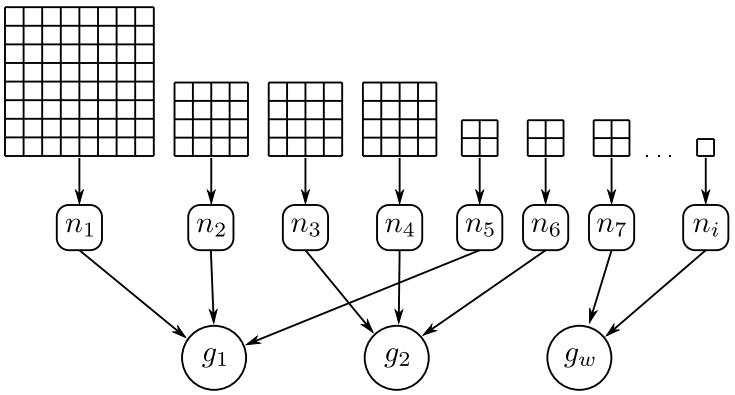


Figure 3.3. Graphic representation of the node (n_i) and group (g_w) structures. The arrows depict pointers to structures.

At the beginning of the merging process, there is no class assigned to any region. In this case, and whenever a region has no neighbors that meet

any of the merging conditions, a *group* structure (g_w) is generated. All the neighboring nodes (n_i) that fulfill the merging conditions, are pointed to a group structure (see Figure 3.3). This structure keeps a record of the sum of the pixel intensities from all the nodes.

Whenever a new node is created, a new class (C_j) structure is also defined, and the group is pointed to that class structure. The class contains a unique label, and stores a vector containing all the group structures that point to the class. If two or more regions of different classes are merged together, their corresponding groups are added to the group vector of the class. This reduces the computing time otherwise used in node reassignment. The Figure 3.4 shows the relationship between the class and group structures.

The merging process is performed using the vector of node structures, generated during the splitting process. This vector is sorted by the size of the regions, where the first nodes correspond to the biggest homogeneous regions in the image. This means that all the classes are started using the biggest region possible, and smaller regions are added later.

The use of intermediate group structures between the node and the class structure is useful for the implementation only. It considerably reduces the number of operations that are required to merge multiple classes together. At the end of this process, the remaining classes are labeled from 1 to L, consecutively.

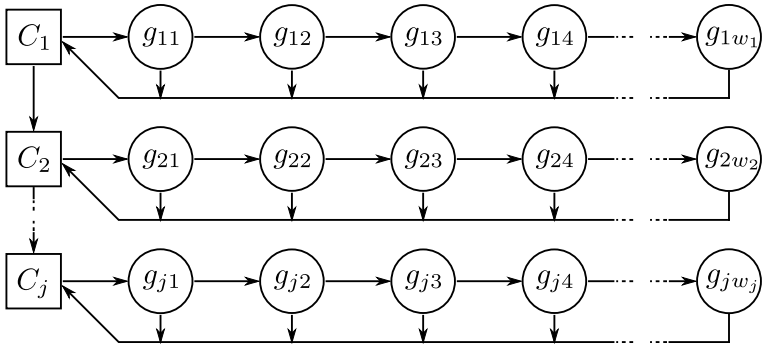


Figure 3.4. Graphic representation of the group and class structures. The arrows depict pointers to structures.

3.2.5 Small-region elimination

At the end of the split and merge segmentation process, all the pixels in the image are assigned to a class in the resulting segmentation. However, there are many classes among them associated only to a small group of pixels.

Normally, it is expected for an image segmentation to have a reduced number of classes, and those classes are expected to group the important areas in the image. For this reason, the excess of small classes may suppose a problem for some applications. It may result convenient to eliminate those small classes, and later reassign their pixels to bigger classes. The elimination process is straightforward. All the class structures are inspected and, if the number of pixels associated to that class is below M pixels, the class is eliminated along with the structures used by it. All the pixels affected are declassified.

If the elimination of small classes is used, it results more convenient to perform the class labeling after this process.

3.2.6 Region growing

The declassified pixels by the small region elimination procedure are reassigned to the class that results more convenient by a region growing procedure. Considering that most of the pixels are already classified, the time required to perform this operation should be short, because the region growing is applied only to a small number of pixels.

The process starts by introducing all the unclassified pixels into a FIFO stack. Each element of the stack is tested in order to determine if that particular pixel may be assigned to a neighboring class. There are two conditions to meet in order to perform that assignment. Firstly, the pixel should be a neighbor of a pixel that is already classified. Secondly, the pixels recently assigned to a class by this process are not considered as classified pixels.

Those pixels that do not meet the assignment conditions are pushed into another FIFO stack, to start a second cycle. The pixels that were assigned in the previous step are now considered as classified pixels. This distinction is made in order to make the regions to grow in layers. The same process is repeated until no unclassified pixels remain.

To summarize the ISM segmentation algorithm presented in this section, a series of examples obtained from each step of the methodology are shown in Figure 3.5. The Figure 3.5a shows the original image. Then, Figure 3.5b shows the intensity values obtained from the original image, used here as a descriptor. Figure 3.5c shows the homogeneous regions obtained from the splitting process. Their boundaries are shown in black. Figure 3.5d shows the homogeneous regions obtained after the merging process. The boundaries are also shown in black. Figure 3.5e shows in red the small regions, containing less than M pixels. Finally, Figure 3.5f shows the resulting segmentation, obtained after the region growing procedure. This image shows each class label in a different color.

3.3 Tests and Results

This section presents the results obtained from of a series of tests, divided in three categories. The first set of tests is made to determine the optimum segmentation parameters for the proposed algorithm. The second one compares the outcome of the proposed methodology with other state-of-the-art segmentation algorithms. The third set is made to experimentally ascertain the execution time of the method.

It is desirable that the proposed methodology reaches similar results than other state-of-the-art methods. Also, the methodology proposed is expected to be fast enough to be used in real-time applications (i.e. 15 fps or more). Our algorithm is compared with the eleven segmentation methods presented in the survey made by Vantaram and Saber [84]. These algorithms are: the Edge Detection and Image Segmentation (EDISON) system by Christoudias *et al.* [14], the Compression-based Texture Merging (CTM) by Yang *et al.* [89], the J-Segmentation (JSEG) algorithm by Deng and Manjunath [21], the Dynamic Color Gradient Thresholding (DCGT) by Balasubramanian *et al.* [6], the Gradient Segmentation (GSEG) algorithm by Garcia Ugarriza *et al.* [79], a multiresolution extension of the GSEG methodology called MAPGSEG by Vantaram *et al.* [85], the Level Set-based Segmentation (LSS) by Sumengen [77], the Gibbs Random Field (GRF) algorithm by Vantaram and Saber [83], the Graph-based Segmentation (GS) algorithm by Felzenszwalb and Huttenlocher [23], the Ultra-metric Contour Map (UCM) segmentation by Arbeláez and Cohen [4], and a Color Texture Segmentation

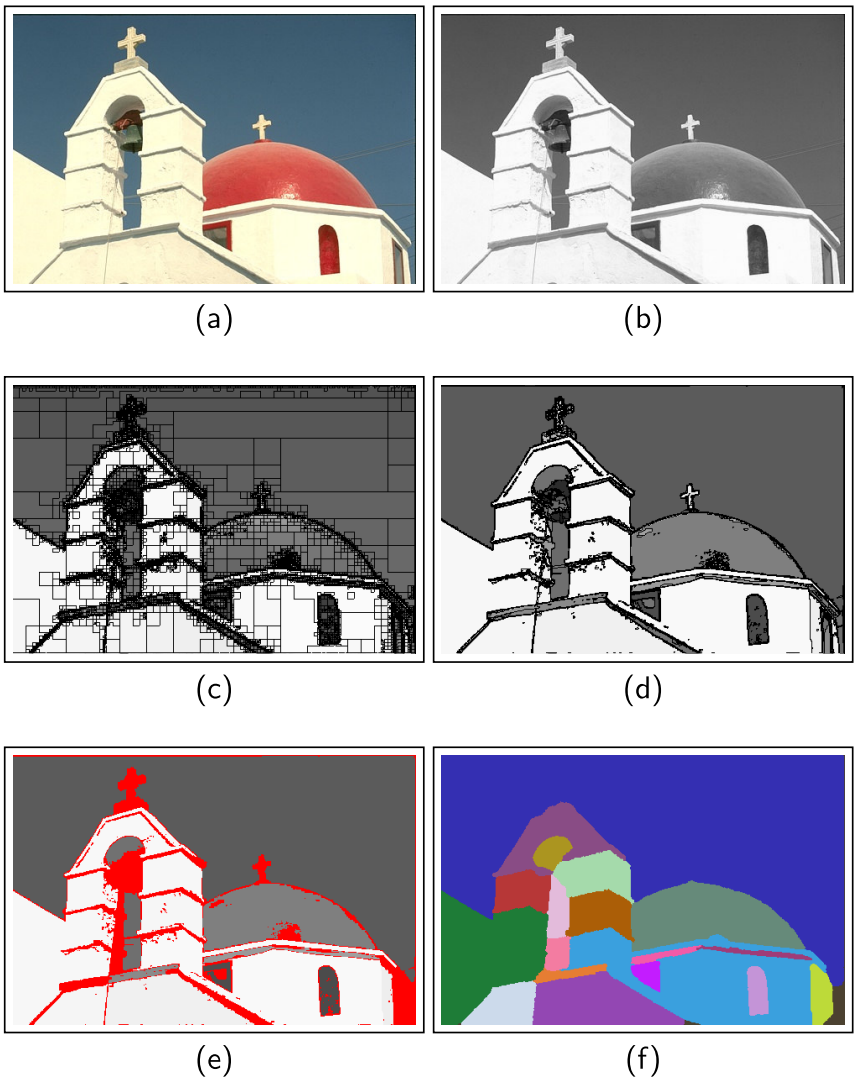


Figure 3.5. Segmentation steps: Original image (a), feature description image (b), image after the splitting process (c), image after the merging process (d), elimination of small regions (e), and the resulting segmentation classes obtained after the region growing procedure (f).

(CTS) by Hoang *et al.* [32]. These segmentation algorithms are evaluated using the Normalized Probability Rand (NPR) index proposed by Unnikrishnan *et al.* [81]. The NPR index requires reference segmentations to determine the evaluation measure, and the Berkeley Segmentation Dataset and Bench-

mark (BSDS) proposed by Martin *et al.* [59] is used for that purpose. This is a collection of 300 natural images that includes from 5 to 10 human-made segmentations for each image.

Our ISM algorithm is also evaluated using the NPR index and the BSDS images, in order to compare our results with those from the eleven algorithms of the survey, under equal conditions. Additionally, to ascertain the significance of the results obtained, a statistical t-test was performed. The test protocols explaining the different experiments are presented next, along with the discussion of the obtained results.

3.3.1 Evaluation of results, using the NPR index

The segmentations resulting from the tests performed were evaluated using the Normalized Probabilistic Rand (NPR) index, a segmentation evaluation methodology proposed by Unnikrishnan *et al.* [81]. This methodology compares an image segmentation against a reference segmentation (i.e. the ideal outcome) and determines their similitude. The NPR index has four desirable properties for segmentation evaluation: Firstly, the method is able to compare segmentations that have a different number of regions. This property is important, because the ISM segmentation methodology automatically determines the number of labels of a segmentation, and may differ from the references or the results from other algorithms. Secondly, the NPR index presents no degenerate cases, in which bad segmentations obtain abnormally good results. Thirdly, the NPR index is able to use multiple segmentation references to obtain a more objective result, and it is able to accommodate the refinement of regions found in the references. Finally, the measure obtained from the NPR index allows the comparison between segmentations of different images, or between segmentations from different algorithms.

In this work, the Matlab toolkit provided by Yang *et al.* [89] is used to compute the PR index. The required human-made references are provided in the BSDS. The NPR index is obtained using Equation 3.17,

$$NPR = \frac{PR - E[PR]}{\max[PR] - E[PR]} \tag{3.17}$$

where $\max[PR] = 1$, and $E[PR] = 0.6064$, according to Vantaram and Saber [84], for the set of 300 natural images in the BSDS.

3.3.2 Statistical tests

For all the tests performed, the segmentation results obtained using the NPR index were averaged. This result is the mean performance of the ISM algorithm, over the 300 images in the BSDS. This mean result was compared with the results from the eleven algorithms in the survey of Vantaram and Saber [84], in order to determine their differences in quality. However, a simple comparison of the mean performance is not enough to provide a conclusion from such comparisons. In cases where the results are too close, the difference may not be significant. In order to identify such cases, statistical significance tests need to be performed for all the comparisons made. In this study, we use a two-sample t-test, where different variances are assumed. Our null hypothesis is that the mean results of two test sets are the same. On the other hand, our alternative hypothesis is that the mean results of two test sets are actually different.

The results obtained are reported in the following sections. Additionally, different tests were performed to optimize the parameter p required by the ISM methodology, the parameter M used for small-region elimination, and the parameter R used in the preprocessing stage to calculate the SD image.

3.3.3 Parameter optimization

This section presents the tests made in order to determine the best segmentation parameters for the proposed methodology. The parameters to optimize are three. The first one is the parameter p of the ISM methodology, used to determine the variance threshold σ_T^2 of the image. This parameter specifies the percentile of the image intensities that should be below σ_T^2 in an adaptation of the method proposed by Garcia Ugarriza *et al.* [79]. The second parameter is the minimum size M allowed for a class in the resulting segmentation. This parameter is used by the small-region elimination process. All pixels of classes below the value of M are declassified and reassigned to bigger classes. The last parameter R defines the size of the window $w = 2R + 1$. This window size is required to calculate the SD image of the preprocessing stage. The SD image is used here as a texture descriptor.

Gradient histogram threshold

The threshold p used to determine the value of σ_T^2, is adapted from the work of Garcia Ugarriza *et al.* [79]. They report a value of 0.8 for their adaptive gradient thresholds. However, because the adaptation of this method may lead to differences in the results, we conducted a test to independently determine the value of p.

For this test, we used the 300 images from the BSDS proposed by Martin *et al.* [59], as a training set. All the images in this set were segmented using different values in $0 \leq p \leq 1$. The segmentation evaluations, obtained using the NPR index show that, the best value is $p = 0.82$. Even though this is the best average evaluation value, the statistical significance test performed shows that the differences of the segmentation results are not significant for values in $0.49 < p < 0.88$, meaning that the method is robust for different values of p. This means that the value of $p = 0.82$ and the value of 0.8 reported by Garcia Ugarriza *et al.* [79] for their adaptive gradient thresholds have no significant differences. From now on, we use the value of $p = 0.82$ only as a preference.

Size of a small region

We conducted a test to determine the optimal value for the minimum class size M, used in the small-region elimination step. The 300 testing images from the BSDS were segmented, this time using different values of M, that vary from zero to 500 pixels. The results were evaluated using the NPR index, and the next results were found.

First, we found that the optimal value obtained is $M = 400$. However, the statistical significance test performed showed that, there is no significant difference in $0 \leq M \leq 500$. Therefore, the elimination of the small classes is not relevant to ascertain the quality of the image. However, it may be used for display purposes. We choose the value of $M = 400$ for the comparison tests, described in Section 3.3.4.

Window size to obtain a deviation image

Another test was conducted to determine the optimal size of the inspecting window, used to obtain the SD image in the preprocessing stage. We used the 300 images in the BSDS to obtain different segmentation sets for different values of R. We average the results from the 300 segmentations for a given R value, and then compare the different average results obtained using different R values. The values of R vary in the range $0 \leq R \leq 20$. The result obtained shows a maximum for $R = 8$. However, the statistical significance test shows that the differences are not significant for values in $5 \leq R \leq 12$. In our implementation, we also used the integral images in the preprocessing stage, to calculate the SD image. This has two advantages: firstly, the SD image is obtained in linear time, and secondly, that time is independent of the size of the window that is used, i.e. R. Therefore, we can use any value of R without performance losses. We choose the of $R = 8$ for the performance tests, only as a preference.

3.3.4 Segmentation performance

This section presents the results obtained from the proposed ISM methodology, using the SD images as input, after the small-region elimination process. The Figure 3.6 shows a diagram of the segmentation process performed using the ISM methodology. The results were compared with other state-of-the-art segmentation methods, and were also evaluated using the NPR index.

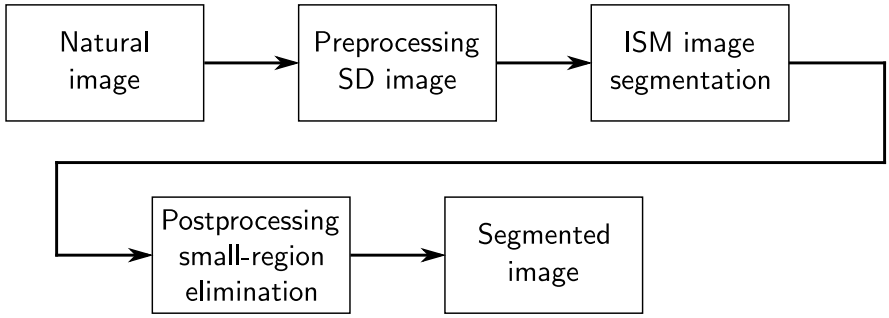


Figure 3.6. Diagram of the segmentation process using the ISM methodology.

The evaluation of the segmentations, obtained from the 300 images from the BSDS using the ISM methodology, obtains an average evaluation $\mu = 0.390$, and a standard deviation $\sigma = 0.636$. The parameters used were $R = 8$ for the SD image in the preprocessing stage, $p = 0.82$ for the ISM segmentation, and $M = 400$ for the small-region elimination step.

The results from our ISM methodology were compared with the eleven algorithms reported by Vantaram and Saber [84]. Some image examples obtained from these results are shown in Figure 3.7. The eleven algorithms used for comparison were applied to the same 300 images from the BSDS than our ISM methodology, and their segmentation results were also evaluated using the NPR index. Therefore, the tests were conducted under the same evaluation conditions.

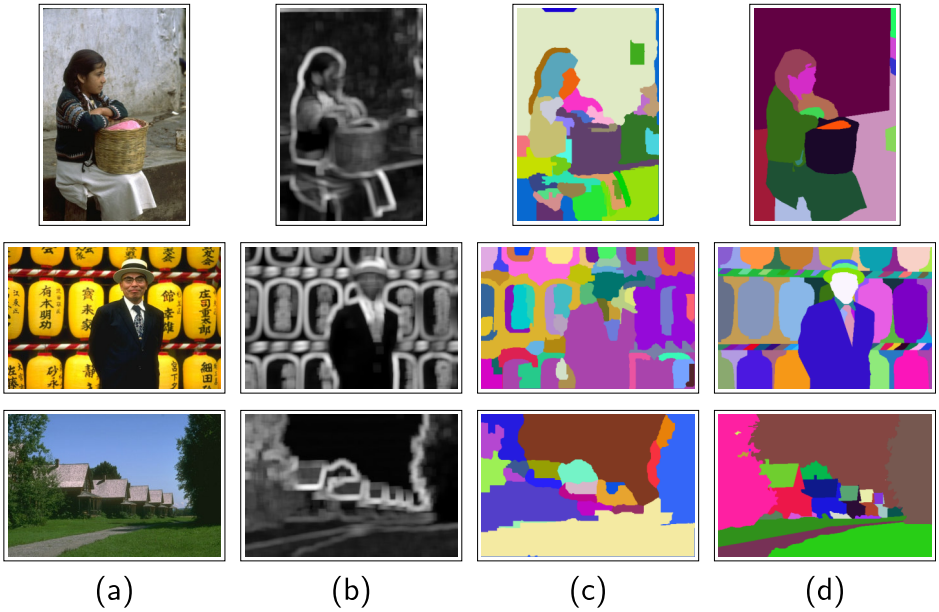


Figure 3.7. Example results: the input image (a), the SD image (b), the segmentation of the SD image using ISM (c), and a human-made reference segmentation (d).

The Table 3.1 shows a comparison of the results obtained by our algorithm and the eleven algorithms. Each algorithm was applied to the 300 images from the BSDS. The resulting segmentations were evaluated using the NPR index. The table shows the mean result for each algorithm (μ_i), and the standard deviation of the results (σ_i). Additionally, the table shows the

Alg.	i	μ_i	σ_i	t-test
ISM		0.390	0.636	
EDISON	1	0.377	0.383	$\mu = \mu_1$
CTM	2	0.386	0.368	$\mu = \mu_2$
JSEG	3	0.440	0.318	$\mu = \mu_3$
DCGT	4	0.394	0.375	$\mu = \mu_4$
GSEG	5	0.496	0.306	$\mu \neq \mu_5$
MAPGSEG	6	0.495	0.312	$\mu = \mu_6$
LSS	7	0.329	0.344	$\mu = \mu_7$
GRF	8	0.488	0.309	$\mu = \mu_8$
GS	9	0.457	0.324	$\mu = \mu_9$
UCM	10	0.507	0.322	$\mu \neq \mu_{10}$
CTS	11	0.214	0.419	$\mu \neq \mu_{11}$

Table 3.1. Results from the evaluation of the 300 image segmentations, using the NPR index. The table shows the algorithm names (Alg.), a numerical label (i), the mean result (μ_i), the standard deviation of the sample (σ_i), and the significance test decision (t-test).

conclusions obtained from the statistical significance t-test. The mean result of each algorithm used for comparison (μ_i), was tested against the mean result from our ISM methodology (μ). The t-test tells if the difference between mean values is significant or not. The table shows $\mu = \mu_i$ when the differences are not statistically significant, and $\mu \neq \mu_i$ otherwise.

The results from the t-test show that, there are no significant differences between the results obtained from our methodology and the algorithms used for comparison, with the exception of the GSEG and the UCM algorithms that obtain better results, and the CTS algorithm that obtains worse results. Therefore, our ISM methodology using the SD image as a texture descriptor, obtains results comparable to most of the eleven algorithms tested. However, the use of a feature descriptor other than the SD image may lead to better results.

3.3.5 Execution time and algorithmic complexity

An execution time test was performed to determine the complexity of the ISM methodology, regarding to the number of pixels processed. For the test,

150 scaled sets of the BSDS were used. The ISM segmentation was applied to all the 300 images in each scaled set. The segmentation of each scaled set was repeated 100 times, just to increase the accuracy of the registered time T. This process was repeated for all the 150 scaled versions. The Figure 3.8 shows as dots the time (ms) obtained experimentally for different number of pixels (i.e. n inputs), corresponding to different scaled versions of the BSDS. The average segmentation time for a typical n-pixel image from the database is obtained by doing $T/30000$.

The results show a piecewise linear behavior, $O(n)$ in Big-O notation. The discontinuities between segments are related to the canvas enlargement step of the ISM methodology. For non-square $W \times H$ image sizes, having $\min(W, H) < 2^\ell/2$, two quads are filled only by empty pixels. The processing required for those quads is negligible. However, for images having $\min(W, H) \geq 2^\ell/2$, there are no empty quads, and the number of quad divisions increases significantly. The time *jump* occurs for image sizes having $\min(W, H) = 2^\ell/2 + 1$. Even though these jumps seem to increase the time consumption, their occurrence decrease in frequency as n increases, because the occurrence of the jumps is in function of powers of two.

The results of the different experiments shown as dots in Figure 3.8 were adjusted to lines, using the least squares fitting methodology. The tests performed on the 300 images from the BSDS for images of 481×321 pixels show that, a single image is segmented in an average time of 32.6 ms, using a microprocessor 4th generation Intel Core i7 at 3.40 GHz. Because of its piecewise linear time complexity, the ISM methodology may achieve real-time segmentation, using the adequate hardware.

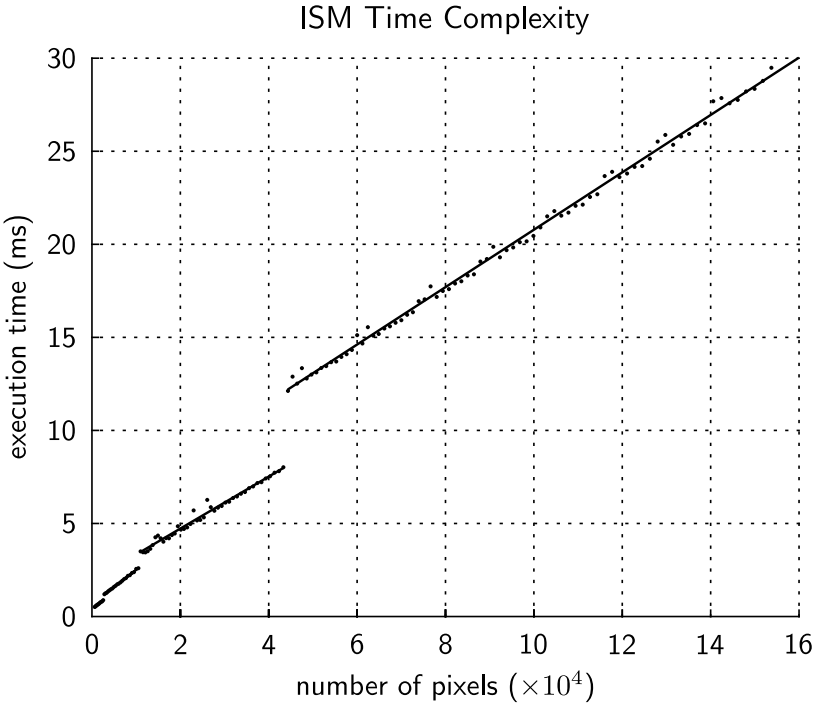


Figure 3.8. Time complexity of the ISM methodology.

Chapter 4

Object recognition using feature histograms

This chapter presents the object recognition system developed for this thesis work. The object recognition methodology uses histograms that combine texture and color features, extracted from the images used for both learning and testing. The Section 4.1 shows an overview of the object recognition field, and of the proposed methodology. Then, the methodology used for our object recognition system is described in Section 4.2. The tests performed using our object recognition methodology are presented in Section 4.3, along with the results obtained from these tests.

4.1 Introduction

The recognition of objects is a task in which a computer algorithm tries to identify an object in an image, regardless of its orientation, scale, illumination, occlusion, and other factors that affect the image. Moreover, the recognition may include not just a particular object, but a family of objects that share some characteristics.

The object recognition methods allow a computer program to identify specific objects of interest in images, and to obtain information about them, like their location, orientation, distance, among others. However, the object recognition is still an open research area, and there are several approaches

to the problem, where the results obtained depend on the specific constrains of a particular application.

In their survey, Roth and Martin [72] classify the object recognition approaches into two main categories: the generative models and the discriminative models. The generative models try to find a suitable representation of the original data. These methods try to approximate the original data to a model, while keeping as much information as possible. A likelihood metric is calculated for a given a sample, and the sample is then assigned to the most likely class. Examples of generative models include principal component analysis (PCA) [44], independent component analysis (ICA) [42], and non-negative matrix factorization (NMF) [50], among others. On the other hand, the discriminative models try to find optimal decision boundaries in the data for a given label. An unknown sample is classified by directly assigning a label, based on the estimated decision boundary. Examples of discriminative models include linear discriminant analysis (LDA) [22], support vector machines (SVM) [86], and boosting [24], among others.

The generative methods are further divided into three subcategories: model-based, shape-based, and appearance-based approaches. The model-based approaches try to approximate the objects as a collection of geometrical primitives, such as spheres, cones, cylinders, boxes, etc. On the other hand, the shape-based approaches try to represent the object by the shape of its edges (i.e. their contour). Regarding to the appearance-based approaches, these methods try to model the object using only its appearance properties, commonly captured using different views of the object.

Regarding to the features used, these methods can be further subdivided into local and global approaches. The local approaches calculate many features at single points or for small neighborhoods in the image. These features by themselves are normally not useful, because they are not invariant to changes in illumination, noise, scale and orientation. Therefore, additional processing is often required to obtain features that are invariant to the aforementioned problems. These processed features are commonly known as descriptors. In contrast, the global approaches obtain information from the whole image, using all the pixels to obtain the model. The goal is to project the original data onto a subspace that optimally represents the data. This allows the global methods to reconstruct the original data to some ex-

tent. This property gives the global methods more robustness than the local methods against partial occlusions of the object of interest.

In this chapter, we present an object recognition methodology belonging to the category of the generative appearance-based methods, using a global approach. Our methodology uses texture and color features, and histograms to model the objects. Example images of different objects are presented to the system in the learning stage. The system then builds a knowledge database from those examples, to be used during the evaluation process. The identification of an object is done by assigning a label to the evaluated image. This label describes a particular object, associating the image under evaluation with a previously learned example.

The object recognition is normally performed in two stages: the feature extraction and the classification [38]. During the learning stage, images of several views from different objects are presented to the system, along with a label that identifies each object shown. The system then extracts the distinctive features from each image, and builds an histogram that describes the object. The given label is then associated to its corresponding histogram in order to build the knowledge database, and it is later used by the classifier to identify unseen views of the trained objects.

The main objective of the system described in this chapter is to distinguish a particular object among previously learned objects. In order to test the system, an image database is needed. This image database should meet certain requirements. First, the database should be a collection of images of many different objects. Also, several different views of each object should be included. Finally, images without background are preferred, because the system here presented is not able to isolate the background from the object of interest, at this stage of development.

We have found an image database that meets the aforementioned requirements in the Amsterdam Library of Object Images (ALOI) [1]. This library contains several databases, and one of those databases is a collection of images of 1000 small objects, from 72 different points of view. The objects in the images are at the same distance from the camera, and the background of all the images is black. We use this database to train and test the object recognition system, using different sets of images for each task.

The system uses four different image features, consisting of a texture feature, the color hue, the color saturation, and the luminance of the image.

These features are extracted from color images, previously transformed to the HSLuv color space. This color space is only a different representation of the CIELuv, made to express its color channels in terms of the image properties: hue, saturation and luminance.

The HSLuv (CIELuv) color space is used instead of simple RGB because it is a perceptual color space, meaning that the color representation emulates the visual response of the human eye. This has advantages over other representations such as the RGB, because color distinctions in the perceptual color spaces are similar to the way the human eye distinguishes different colors. In this regard, the frontiers between colors found by an algorithm are closer to those frontiers identified by the human eye. We have found that the use of perceptual color spaces improves image segmentation [17], and may also improve the methodology presented here. The use of CIELuv instead of other color spaces, such as the widely known CIELab color space, is preferred because CIELuv is faster to calculate, whilst producing similar results than CIELab in segmentation applications.

In addition to the color features, we also included a simple texture feature: the standard deviation (SD) image, previously discussed in Chapter 3. This texture feature registers the intensity variations in the neighborhood of each pixel of the image, meaning that the noisy or textured regions are grouped together into regions of homogeneous intensity in the feature image. We choose this feature because of its simplicity and because it has previously obtained good results, such as in the work of Lizarraga-Morales *et al.* [56].

The features that are extracted from the image in the HSLuv color space are the hue, saturation, luminance, and the SD texture. Each feature is saved into a single image or feature map. These four feature maps are then analyzed in order to obtain a compact representation that describes the image trough histograms. An histogram is calculated for each image feature and then, a combined histogram is assembled, holding the statistics of each image feature. The histograms obtained in this way are used both for learning and testing.

The learning methodology used in this work is as follows. A combined histogram is calculated for each image in a learning set, and is associated to its corresponding label. The knowledge database is then built as a collection of histogram–label entries, containing all the feature histograms obtained from the learned examples.

The testing procedure consists in extracting the combined histogram from an image under test. Then, the histogram is compared against all the histograms in the knowledge database. A distance metric is used to compare the histograms, and the label of the histogram in the knowledge database, that is more alike to the histogram under test, is assigned to the tested image.

For testing the system, we used the image database from the ALOI, and divided the images into two sets. The first set is used for training, and the second set is used for testing. We compare the number of correct classification outcomes, and the number of wrongly classified images, to obtain a classification rate. Our results show a classification rate above 90%, depending on how many examples are used for training.

4.2 Methodology

In this section, the methodology used for the object recognition system is presented. An image dataset containing different views of several objects is used, both for learning and classification. First, each image is transformed to the CIELuv color space, from which the visual features of the object are extracted as feature maps. Then, combined feature histograms are obtained from those feature maps. In the learning stage, the combined histograms obtained from a training set are saved in a knowledge database, along with the labels assigned to their corresponding images. For the classification stage, a combined feature histogram is obtained from an image under test, and is compared against all the histograms stored in the knowledge database. The label of the histogram in the knowledge database that is closer to the tested histogram is then assigned to the image under test. Details of each step of the object recognition methodology are presented next.

4.2.1 Color features from the HSLuv color space

To obtain the image features, we use a transformation of the CIELuv color space called HSLuv. The HSLuv color space consists of three channels: hue, saturation, and luminance. Each channel is obtained from the original CIELuv color space, using the Equations 4.7, 4.8, and 4.2, respectively.

The components u^* and v^* of the CIELuv color space are obtained from the CIEXYZ color space, using the Equations 4.3 to 4.6.

The Equation 4.1 is used to transform the sRGB color representation to the CIEXYZ color space. We assume that the images used in this work are all in the sRGB color space, whose components are defined as $r^*, g^*, b^* \in [0, 255]$. The Equation 4.1 requires normalized components of the sRGB color space, defined as $r, g, b \in [0, 1]$.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.4124564 & 0.3575761 & 0.1804375 \\ 0.2126729 & 0.7151522 & 0.0721750 \\ 0.0193339 & 0.1191920 & 0.9503041 \end{bmatrix} \begin{bmatrix} r \\ g \\ b \end{bmatrix} \tag{4.1}$$

The CIELuv color space is derived from the color components in the CIEXYZ color space. The CIELuv color space consists of three components: L^*, u^*, and v^*; where the component L^* is the luminance, whilst u^* and v^* are chromatic components. The luminance (L^*) is calculated using the Equation 4.2, and the chromatic components (u^* and v^*) are calculated using the Equations 4.3 to 4.6.

$$L^* = \begin{cases} \left(\frac{29}{3}\right)^3 \frac{Y}{Y_n} & \frac{Y}{Y_n} \leq \left(\frac{6}{29}\right)^3 \\ 116\left(\frac{Y}{Y_n}\right)^{\frac{1}{3}} - 16 & \frac{Y}{Y_n} > \left(\frac{6}{29}\right)^3 \end{cases} \tag{4.2}$$

$$u' = \frac{4X}{X + 15Y + 3Z} \tag{4.3}$$

$$v' = \frac{9Y}{X + 15Y + 3Z} \tag{4.4}$$

$$u^* = 13 \cdot L^* \cdot (u' - u'_n) \tag{4.5}$$

$$v^* = 13 \cdot L^* \cdot (v' - v'_n) \tag{4.6}$$

where Y_n is the Y component of the color used as reference white, required to perform color balance on the image. We do not require this color balance for our aplication, and for practical purposes we asume $Y_n = 1$, which produces no color balance.

The HSLuv color representation is then obtained from the components of the CIELuv color space. The luminance component is the same calculated

for the CIELuv space (Equation 4.2). The hue component (H^*) is calculated using the Equation 4.7, and the saturation (S^*) is obtained using the Equation 4.8.

$$H^* = 2\tan^{-1}\frac{\sqrt{(u^*)^2+(v^*)^2}-u^*}{v^*} \tag{4.7}$$

$$S^* = \frac{\sqrt{(u^*)^2+(v^*)^2}}{L^*} \tag{4.8}$$

4.2.2 Normalization of the HSLuv color space

The HSLuv components are defined in different ranges. In order to generate the required feature maps, a normalization stage is required. The HSLuv color space is normalized linearly for the components H^* and L^*, as shown in the Equations 4.9 and 4.10, respectively. However, the component S^* required a special normalization criteria.

$$H = \frac{1}{2\pi}H^* \tag{4.9}$$

$$L = \frac{1}{100}L^* \tag{4.10}$$

The sRGB colors transformed to HSLuv show, in general, a low saturation value. The exception are few colors from all the representable sRGB colors. This shows a non-linear distribution for the saturation component of the HSLuv color space. Whilst most colors have low saturation values, few colors exhibit high saturation values.

This distribution of the saturation values is not ideal for the classification methodology used in this work, because it makes difficult to determine levels in the saturation component that allow the identification of objects using this component. The range of values where the distinction of object features should be performed is very narrow. This should be not a problem if real numbers of enough resolution were used. However, the methodology uses integer images as input, where the intensity values are in the range $[0, 255]$. Therefore, the saturation component (S^*) should be adjusted in order to make a better use of the intensity values available.

To do this adjustment, we use a sigmoid function. This function moves away the low saturation values, and brings near the high saturation values. This does not make the distribution linear, but makes a better use of the intensity range available. The Equation 4.11 shows the sigmoid function used to do this adjustment.

$$S = \frac{2}{1 + \exp(-k \cdot S^*)} - 1 \tag{4.11}$$

where $k = m/S^*_{\max}$ is a constant, being m the slope given to the sigmoid function, and $S^*_{\max}$ the maximum saturation value for the color space used.

By transforming to HSLuv all the representable sRGB colors, we have found that the maximum saturation (S^*) value that can be obtained from a given sRGB color is $S^*_{\max} = 4.0462337$. The slope m of the sigmoid function was determined by trying different integer values, and checking the resulting distribution. The objective here was not to obtain an optimal distribution, but to separate apart the low saturation levels to aid the histogram representation. We have found that using $m = 4$ obtains a good adjustment. Therefore, we use $k = 0.988774 \approx 1$ in the Equation 4.11 to normalize the saturation component of the HSLuv color space.

4.2.3 Texture feature from the Luminance (L) channel

The luminance component of the HSLuv color contains only the light intensity information of the image. No chromatic information is contained in this component. Because of this, the luminance component is used to obtain the textural feature used by the methodology.

In this work, we use a simple texture descriptor: the standard deviation (SD) map, or SD image as described in Chapter 3, Section 3.1. We use the neighborhood $R = 8$ (a window of 17×17 pixels) to obtain the SD image, as determined in the tests described in Chapter 3.

Along with the HSLuv components H^*, S^* and L^*; we include an additional feature T^*, that is the standard deviation obtained from the luminance component of the image. The normalized version of T^* is calculated using Equation 4.12,

$$T = \frac{T^*}{T^*_{\max}}, \tag{4.12}$$

where $T^*_{\max}$ is the maximum value found for the SD images in the data set. The normalized versions of each feature (T, H, S, and L) are used to build the required histograms, and from now on are referred as THSL features.

4.2.4 THSL feature histogram

After obtaining the feature images (Texture, Hue, Saturation, and Luminance), an histogram is calculated for each feature. In order to obtain better results, the number of bins for the histograms should be determined experimentally, and may be different for each feature. Finally, the feature histograms are concatenated to build the THSL histogram. In this work, we chose to concatenate the feature histograms in the order THSL. However, this arrangement has no impact in the final results. In this work, each THSL histogram is obtained from a single image, either from the training or from the testing data set.

4.2.5 Learning methodology: Histogram collection

The learning process is performed by examples. Each example consists of a single image, and a label that identifies the object in the image. The learning process is made using as many images as required.

From each image, a THSL histogram is obtained. All the histograms are saved in a set of labeled histograms, describing the different objects in the image set. This collection of histograms and labels is our knowledge database, and is used by the classification step to identify objects.

4.2.6 Histogram Intersection distance

The classification process requires the comparison of a THSL histogram obtained from an image under test, and the multiple THSL histograms stored in the knowledge database.

To compare histograms, we tried several distance metrics commonly used to compare histograms. The tested distances were: Correlation, Chi-Square, Intersection, and Bhattacharyya distances. In an early design stage we tried a simple one-to-one histogram comparison using each of these algorithms. The results obtained were poor for all distances, except for the Intersection distance.

The intersection distance is defined as follows. Let C be an histogram containing B bins defined as $C = \{c_1, c_2 \ldots, c_B\}$, where $C(i) = c_i$. The intersection distance d between an histogram under test C_t and a reference histogram C_r, both of size B, is then calculated using the Equation 4.13.

$$d(C_t, C_r) = \sum_{i=1}^{B} \min\left(C_t(i), C_r(i)\right) \tag{4.13}$$

A more detailed study of this distance shows properties that are useful for the classification methodology designed for this work. The intersection distance ignores those bins that are present in the reference, but absent in the histogram under test. This property of the intersection distance is useful for the object recognition task described in this work, because a single object may have different visual properties, depending on the point of view of the object that is presented to the classifier. If a given feature bin is not present in the histogram under test, that bin is just ignored by the distance metric.

This distance is the sum of the minimum bins from each histogram. For the intersection distance, the comparison is better the larger is the distance. In order to obtain high distance values, the difference between bins from the histogram under test and from the reference histogram should be small. The maximum distance value is equal to the sum of the bin values of the reference histogram.

4.2.7 Histogram classification using the intersection distance

In order to classify a given histogram, obtained from an image containing an object to identify, we use the next procedure. The distances between the histogram obtained from the image under test, and all the histograms in the learning database are calculated, using the intersection distance. Then, the

label of the histogram in the knowledge database with the largest distance to the histogram tested, is assigned to the image under test.

This is a simple classification method that is not very efficient in the use of memory, and requires the optimization of the number of bins to use for each feature histogram. However, this classification methodology obtains high classification rates in our experimental tests. Several tests were performed in this regard, and are described in the next Section 4.2, along with the results obtained.

4.3 Tests and Results

In this section, the results obtained from the object recognition system, described in Section 4.2, are presented. We also present the tests performed to determine the optimal number of bins to use for the THSL feature histograms, and the proportion of bins for each THSL feature. For all the tests, we used the ALOI 1000 image database, described next.

4.3.1 Image Database: ALOI 1000

The objective of the methodology presented in Section 4.2 is to learn the most important features that describe an object, when different views of the object are presented to the system. Therefore, an image database providing different views of several objects was required for testing the method. In this regard, we used an image database from the Amsterdam Library of Object Images (ALOI) [26]. The ALOI provides different sets of object images, with varying illumination direction, different illumination color, different object viewpoints, and wide-baseline stereo images. Also, the ALOI provides different versions of the aforementioned image sets at different resolutions: full (768×576), half (384×288), and quarter (192×144). All the objects in the ALOI images are small objects, and are placed at the same distance from the camera. Also, the background of the images in the dataset is always black. At this stage, we need to test our classifier without the adverse effect of an arbitrary background, and the ALOI database provides images that meet our requirements. Some examples of the objects found in the ALOI 1000

Figure 4.1. Examples of the object images found in the ALOI dataset.

database are shown in Figure 4.1. The Figure 4.2 shows examples of the images taken from a single object, from different points of view.

For this work, we use a data set containing 72 000 images, consisting of 72 different points of view of 1000 different objects (taken in steps of 5°), with constant illumination of the scene, and at half resolution (384×288). From the 72 000 images in the database, we randomly select images to obtain two sets, a training set and a testing set. The random selection was performed without replacement, so that no image is contained in both sets, and no image is left out. In this regard, different views of the same object are used to train the classifier. The views not used for training were used to test the generalization capability of the classifier.

4.3.2 Optimal number of bins for the THSL histogram

During the experimentation with histogram lengths, we have found that the number of bins used for each feature of the THSL histogram affects the classification result. In order to determine the optimal number of bins for

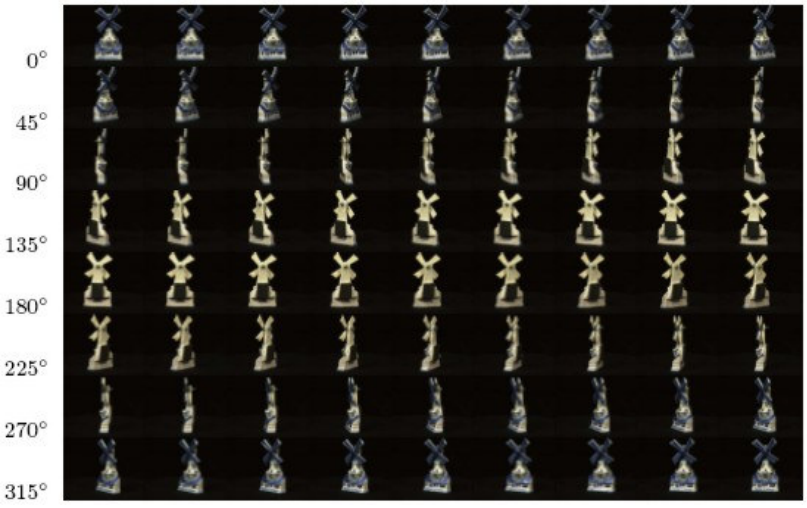


Figure 4.2. Image examples from different points of view of a single object from the ALOI 1000 database.

each feature histogram, several tests were made. We performed training and classification of the image database for different histogram configurations.

We tried different histogram sizes, varying from 20 bins to 250 bins, for each THSL feature. The number of bins were adjusted independently for each feature, using a hill climbing methodology based on successive approximation. At first, the combinations tried were modified using large steps. Then, the steps were gradually reduced as the solution improved. At the end of this optimization process we obtained the following THSL histogram configuration: The best size for the texture histogram is $B_T = 163$ bins, for the hue histogram is $B_H = 155$ bins, for the saturation histogram $B_S = 52$ bins, and for the luminance histogram $B_L = 170$ bins. The resulting THSL histogram consists of the combination of the four feature histograms, resulting in an histogram of 540 bins.

The classification results obtained using this histogram configuration are presented in the next section.

4.3.3 Classification performance

In order to test the classifier, we randomly divided the image dataset into two image sets, one used to train the classifier and the other for testing. However, the optimal number of examples required for training was unknown.

In order to determine an acceptable size for the training set, different training sets were generated, each one of a different size, varying from 10% to 90% of the original image set. For each training set, a testing set was generated, containing the rest of the images of the database not used in the training set.

The classifier was trained and tested, using a different pair of training and testing sets each time. The results obtained are shown in Table 4.1, where the column 'Pair' specifies a label for the train–test set pairs used for classification. The column 'Training' shows the percentage of the images from the full database used to build the training set. The column 'Testing' shows the percentage of images from the full dataset used for testing. Finally, the column 'C.Rate' shows the classification rate achieved using the corresponding train–test set pair.

Pair	Training	Testing	C.Rate
1	10%	90%	89.58%
2	20%	80%	94.69%
3	30%	70%	96.26%
4	40%	60%	96.96%
5	50%	50%	97.66%
6	60%	40%	97.93%
7	70%	30%	98.24%
8	80%	20%	98.27%
9	90%	10%	98.61%

Table 4.1. Classification rate (%) for different training and testing set sizes, measured in a percentage of the full image dataset.

The results obtained are also shown in Figure 4.3. The graphic contains a curve approximated from the data, showing the performance of the classifier for each training set size, from 10% to 90% of the full image database.

The results show that, a classification ratio of 89.58% was achieved using only the 10% of the full image database for training. These results are

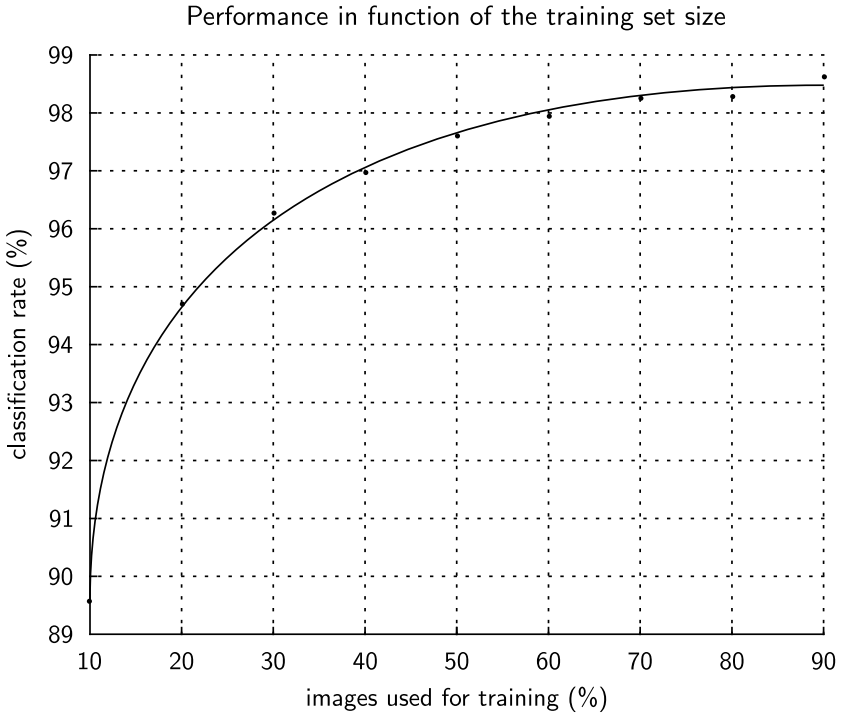


Figure 4.3. Curve approximation of the classification performance, in function of the percentage of images from the full dataset used for training.

improved for larger training sets, reaching a classification rate of 98.61% for a training set consisting of the 90% of the full image database.

The results obtained show that a good classification performance may be achieved using a portion of the original database. Even though the classification performance increases with the number of examples, the graph in Figure 4.3 shows that this increment is smaller as the number of examples increase. The shoulder of the graph provides a good balance between the number of learning examples required and the classification performance, being that balance point close to the 30% of the images in the database.

It is important to note that the ALOI database contains some irregularities, assumed to be on purpose. For example, there are images of different objects with the same label. Also, there are different labels assigned to images of the same object. These special cases in the image database forbid the classifier to reach the 100% classification rate.

Chapter 5

Concluding remarks

In this chapter, the concluding remarks of the studies presented in this thesis are given in Section 5.1. Also, the perspectives of future work following the research lines discussed in this thesis are presented in Section 5.2. As a personal note, I include some thoughts about the work developed during the doctorate in Section 5.3. Finally, references to the published work derived from this study are given in Section 5.4.

5.1 Conclusions

Conclusions: Real-time template matching

In this study, the maximum cardinality similarity metric (MCSM) is presented as a replacement of the partial Hausdorff distance (PHD) for real-time image processing applications. The conducted tests show that the proposed MCSM methodology shares the same robustness against occlusion, partial edges, and localization errors than the PHD. Nonetheless, for noisy edge images, the MCSM and the PHD methodologies show differences: whilst the PHD is insensitive to few missing edge pixels, the MCSM accurately detects the proportion of missing edge pixels. This property may be advantageous for some applications where edge completeness is required to be determined. Applied to template matching, both the MCSM and the PHD methodologies show similar results at finding the target shape in natural images, indicating that the MCSM may be used for edge image registration. Finally, the com-

puting time test shows that the MCSM is much faster to compute than the PHD for a large set of points, a property that enables the MCSM methodology to be used in real-time image processing applications.

Conclusions: Integral split and merge segmentation

In this study, we propose the Integral Split and Merge (ISM) segmentation methodology, a split-and-merge segmentation algorithm that uses integral images to achieve real-time segmentation, through the fast computation of statistics. Additionally, the ISM methodology performs connected component analysis. Therefore, regions showing equal features are labeled as different regions if they are not spatially connected. Also, the method is able to follow the gradients present in some image areas, and group those areas into a single region. Lastly, the ISM methodology automatically determines the number of regions in each image segmentation. The comparison of results between our ISM methodology and other state-of-the-art algorithms, show that our ISM methodology obtains results as good as most of the methods evaluated. Even though the results from the GSEG and the UCM algorithms obtained a better performance, the outcomes from the ISM method using a simple SD image as a texture descriptor, are comparable to those of the rest of the algorithms. However, this quality is achieved by the ISM method using a single texture feature, whilst the other methods use a combination of both color and texture features. Additionally, better results may be achieved using different feature descriptors, or by the combination of two or more descriptors. Regarding to the execution time, the execution tests show that the ISM methodology is executed with a piecewise linear algorithmic complexity. To our knowledge, none of the comparison algorithms achieves this execution time. Our methodology is able to obtain image segmentations in a rate about 32 fps, for images of 481×321 pixels, that depending on the application may be considered real-time. The results show that the ISM methodology may be a good alternative for applications that need a fast segmentation using few image features, whilst still achieving an acceptable segmentation quality.

Conclusions: Object recognition using feature histograms

In this study, we presented an object recognition system that is able to learn from examples given to the system in real time. The method uses an ap-

proach based on feature histograms. The feature histograms are obtained from four different perceptual features extracted from images, that combine texture and color features. Regarding to the classification stage of the system, a knowledge database is built, containing the histograms extracted from each learned image, and a label that describes the object in the image. The classification is then performed by comparing a new histogram, extracted from the image under test, using the intersection histogram distance metric. We tested this system on an image database from the Amsterdam Library of Object Images, a collection of objects captured from different points of view. The results obtained from a series of tests show a classification success rate of 98.61%, using a set containing the 90% of the images in the dataset for training. However, we found that a set with only the 30% of the images provides a good trade-off between learned examples and performance, obtaining a classification success rate of 96.26%. Even though the results are close to the 100% classification rate, some issues remain. The system still needs to be made robust against different image conditions, e.g. isolate the object from the background, insensitivity to different illumination conditions, identification of objects regardless of their scale. Moreover, a test against state-of-the-art object recognition systems is still pending. However, the system described in this study fulfills the original purposes of this work.

5.2 Future perspectives

The study presented in this thesis document opens several lines of research that may be explored in the near future. Each methodology discussed in this thesis may be improved and used in different applications. In this section, we present some of the ideas that we have for future research.

Some improvements may be still made on the MCSM methodology, by improve object detection by adding invariance to scale and rotation. Also, one of the strong points of the MCSM methodology is the ability to handle edges with missing pixels, and still obtain a metric that correctly reflects the degree of similitude between a template and the edge image being tested. We may exploit this property to reduce the data that is processed, improving the speed by using only a small fraction of the edge points in the comparison.

Regarding to the ISM segmentation methodology, a combined texture–color segmentation test is pending. In this work, we presented the ISM

segmentation using a single texture descriptor: the SD image. We compared our results against state-of-the-art methodologies using both color and texture features. This test will tell if the ISM methodology can benefit from combining texture and color features, such is the case of the segmentation methods used for comparison in this work.

With respect to the object recognition system, there is still some work to be done before this study can be published. There is still room for speed improvements. It would be better if the method is refined in order to improve the description of the objects. Also, the classifier may be optimized using some heuristic search. These improvements may save memory and improve the execution time. Additionally, the method requires some robustness against scale and illumination changes. The results obtained should be then compared against other object recognition methodologies.

The following step would be to identify the trained objects in complex environments. This task is more complex and computationally expensive, and will require the aforementioned improvements to the method.

5.3 Final thoughts

In this section, I would like to include some final thoughts about the knowledge and experience acquired during the development of this doctoral work. The study presented in the previous chapters of this thesis is the result of this knowledge and experience, that I consider an essential part of my training.

Regarding to the problem at hand, the recognition of objects in images, I understand this as a problem that requires to find a description of an object, given one or more image examples of that object. This description is in essence a data construction, containing the essential information required to distinguish the object of interest from everything else. The problem lies in how to model this object description, and how to extract the essential information from the image examples provided. There are many ways to approach this problem. In this work, we present a method that performs extraction of information and description building of the objects presented in images. The solution is simple, but obtains good results for the cases of study.

Another important aspect of the thesis work regards to the execution of the system, that should be performed in real time. In order to achieve this, the implemented software requires to be optimized for speed. In my experience, this optimization implies two important aspects: the use of optimal algorithms, and the use of efficient techniques to write the source code. These two aspects are combined to reduce the number of CPU instructions required to obtain the desired results.

Good programming practices recommend to keep a source code that is clear and easy to read. Also, it is recommended to use only enough memory to run the algorithm at hand. However, in order to gain execution speed, sometimes it is required to use algorithms that are more complex, reducing the clarity of the code. Moreover, for some cases higher execution speeds can be reached by using more memory, basically used to hold pre-calculated data. Because the speed is critical for a real time system, sometimes we had to break those rules by sacrificing clarity and memory, in order to gain speed for the system. Fortunately, the memory is not a big concern for the computing equipment currently available.

Finally, there is a key aspect of the research work that I would like to mention: publication. Publication is essential for the scientific labor, because the results obtained from the research work are only useful if known by other researchers, that may benefit from them. However, this is a challenging endeavor that requires the development of writing skills and thorough attention to details. I have found that constant practice and care are necessary in order to obtain a manuscript that meets the criteria for publication. I believe that the experience gained in this regard is one of the most important and valuable skills developed during the doctorate.

5.4 Scientific results

As a final note, we would like to mention that the development of this thesis study produced three publications in scientific journals with JCR impact factor. The references to these works are presented next.

- F.E. Correa-Tome and R.E. Sanchez-Yanez, Integral split-and-merge methodology for real-time image segmentation, Journal of Electronic Imaging, vol. 24, no. 1, p. 013007, 2015, DOI: 10.1117/1.JEI.24.1.013007.

- R.A. Lizarraga-Morales and R.E. Sanchez-Yanez and V. Ayala-Ramirez and F.E. Correa-Tome, Integration of color and texture cues in a rough set-based segmentation method, Journal of Electronic Imaging, vol. 23, no. 2, p. 023003, 2014, DOI: 10.1117/1.JEI.23.2.023003.

- F.E. Correa-Tome and R.E. Sanchez-Yanez, Fast similarity metric for real-time template-matching applications, Journal of Real Time Image Processing, pp. 1–9, 2013, DOI: 10.1007/s11554-013-0363-0.

Bibliography

[1] Amsterdam library of object images (ALOI). http://aloi.science.uva.nl/, accessed on February 2014.

[2] M. Ali, M. Ghafoor, I.A. Taj, and K. Hayat. Palm print recognition using oriented Hausdorff distance transform. In *Frontiers of Information Technology*, pages 85–88, 2011.

[3] P. Arbelaez, C. Fowlkes, and D. Martin. The berkeley segmentation dataset and benchmark. http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/, accessed on Dicember 2012.

[4] P.A. Arbeláez and L.D. Cohen. A metric approach to vector-valued image segmentation. *Int. J. Comp. Vis.*, 69(1):119–126, 2006.

[5] A. Argyriou, T. Evgeniou, and M. Pontil. Convex multi-task feature learning. *Machine Learning*, 73(3):243–272, 2008.

[6] G.P. Balasubramanian, E. Saber, V. Misic, E. Peskin, and M. Shaw. Unsupervised color image segmentation using a dynamic color gradient thresholding algorithm. In *SPIE XIII Human Vision and Electronic Imaging 6806*, page 68061H, 2008.

[7] D.A. Bell and H. Wang. A formalism for relevance and its application in feature subset selection. *Machine Learning*, 41(2):175–195, 2000.

[8] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(4):509–522, 2002.

[9] D. Bradley and G. Roth. Adaptive thresholding using the integral image. *Journal of Graphics, GPU, and Game Tools*, 12(2):13–21, 2007.

[10] M.C. Burl and P.G. Wetzler. Onboard object recognition for planetary exploration. *Machine Learning*, 84(3):341–367, 2011.

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

[12] C. Cardie. A cognitive bias approach to feature selection and weighting for case-based learners. *Machine Learning*, 41(1):85–116, 2000.

[13] T.F. Chan and L.A. Vese. Active contours without edges. *IEEE Trans. Image Proc.*, 10(2):266–277, 2001.

[14] C.M. Christoudias, B. Georgescu, and P. Meer. Synergism in low-level vision. In *16th IEEE Int. Conf. Pattern Recogn.*, volume 4, pages 150–155, 2002.

[15] F.E. Correa-Tome and R.E. Sanchez-Yanez. Fast similarity metric for real-time template-matching applications. *J. of Real Time Image Proc.*, pages 1–9, 2013.

[16] F.E. Correa-Tome and R.E. Sanchez-Yanez. Integral split-and-merge methodology for real-time image segmentation. *J. Elec. Imaging*, 24(1): 013007, 2015.

[17] F.E. Correa-Tome, R.E. Sanchez-Yanez, and V. Ayala-Ramirez. Comparison of perceptual color spaces for natural image segmentation tasks. *Opt. Eng.*, 50(11):117203–117203–11, 2011.

[18] F.E. Correa-Tome, R.E. Sanchez-Yanez, and V. Ayala-Ramirez. Measuring empirical discrepancy in image segmentation results. *IET Comput. Vis.*, 6(3):224–230, 2012.

[19] M. Dash and H. Liu. Feature selection for classification. *Intell. Data Anal.*, 1(3):131–156, 1997.

[20] H. Dastmalchi, J. Jafaryahya, R. Najafi, and A. Daneshkhah. Averaged segmental partial Hausdorff distance for robust face recognition. In *2nd International Conference on Intelligent Systems*, pages 35–39, 2011.

[21] Y. Deng and B.S. Manjunath. Unsupervised segmentation of color-texture regions in images and video. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(8):800–810, 2001.

[22] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*. John Wiley & Sons, 2000.

[23] P.F. Felzenszwalb and D.P. Huttenlocher. Efficient graph-based image segmentation. *Inter. J. Comp. Vis.*, 59(2):167–181, 2004.

[24] Y. Freund and R.E. Shapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comp. Sys. Sciences*, 55 (1):119–139, 1997.

[25] H. Gao, W.C. Siu, , and C.H. Hou. Improved techniques for automatic image segmentation. *IEEE Trans. Circuits Sys. Video Tech.*, 11(12): 1273–1280, 2001.

[26] J.M. Geusebroek, G.J. Burghouts, and A. W. M. Smeulders. The Amsterdam library of object images. *Int. J. Comput. Vision*, 61(1):103–112, 2005.

[27] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3:1157–1182, 2003.

[28] I. Hanniel, A. Krishnamurthy, and S. McMains. Computing the Hausdorff distance between NURBS surfaces using numerical iteration on the GPU. *Graph. Models*, 74(4):255–264, 2012.

[29] K. Haris, S.N. Efstratiadis, N. Maglaveras, and A.K. Katsaggelos. Hybrid image segmentation using watersheds and fast region merging. *IEEE Trans. Image Proc.*, 7(12):1684–1699, 1998.

[30] A. Hartigan and M.A. Wong. A k-means clustering algorithm. *Appl. Statist.*, 28(1):100–108, 1979.

[31] P.R. Hill, C.N. Canagarajah, and D.R. Bull. Image segmentation using a texture gradient based watershed transform. *IEEE Trans. Image Proc.*, 12(12):1618–1633, 2003.

[32] M.A. Hoang, J.M. Geusebroek, and A.W.M. Smeulders. Color texture measurement and segmentation. *Elsevier Sig. Proc.*, 85(2):265–275, 2005.

[33] J.E. Hopcroft and R.M. Karp. A $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. In *IEEE Annual Symposium on Foundations of Computer Science*, pages 122–125, 1971.

[34] S.L. Horowitz and T. Pavlidis. Picture segmentation by a directed split-and-merge procedure. In *Proc. 2nd Int. Joint Conf. on Pattern Recognition IJCPR*, pages 424–433, 1974.

[35] S.L. Horowitz and T. Pavlidis. Picture segmentation by a tree traversal algorithm. *Journal of the ACM*, 23(2):368–388, 1976.

[36] M.J. Hossain, M.A.A. Dewan, K. Ahn, and O. Chae. A linear time algorithm of computing Hausdorff distance for content-based image analysis. *Circ. Syst. Signal. Process.*, 31(1):389–399, 2011.

[37] M.J. Hossain, M.A.A. Dewan, and O. Chae. A flexible edge matching technique for object detection in dynamic environment. *Appl. Intell.*, 36 (3):638–648, 2011.

[38] G.-S. Hsu, T.T. Loc, and S.-L. Chung. A comparison study on appearance-based object recognition. In *Proc. 21st International Conference on Pattern Recognition (ICPR 2012)*, pages 3500–3503, 2012.

[39] Y. Hu and Z. Wang. A similarity measure based on Hausdorff distance for human face recognition. In *18th International Conference on Pattern Recognition*, pages 1131–1134, 2006.

[40] D.P. Huttenlocher and W.J. Rucklidge. A multi-resolution technique for comparing images using the Hausdorff distance. In *Computer Vision and Pattern Recognition*, pages 705–706, 1993.

[41] D.P. Huttenlocher, G.A. Klanderman, and W.J. Rucklidge. Comparing images using the Hausdorff distance. *IEEE Trans. Pattern Anal. Mach. Intell.*, 15(9):850–63, 1993.

[42] A. Hyvärinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. John Wiley & Sons, 2001.

[43] Y.H. Ji, J.B. Song, and J.H. Choi. Outdoor mobile robot localization using Hausdorff distance-based matching between COAG features of elevation maps and laser range data. In *11th International Conference on Control, Automation and Systems*, pages 686–689, 2011.

[44] I.T. Jolliffe. *Principal Component Analysis*. Springer, 2002.

[45] R. Jonker and A. Volgenant. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, 38:325–340, 1987.

[46] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *Int. J. Comp. Vis.*, 1(4):321–331, 1998.

[47] J.B. Kim and H.J. Kim. Multiresolution-based watersheds for efficient image segmentation. *Pattern Recogn. Lett.*, 24(1–3):473–488, 2003.

[48] C. Knauer, M. Löffler, M. Scherfenberg, and T. Wolle. The directed Hausdorff distance between imprecise point sets. *Theor. Comput. Sci.*, 412(32):4173–4186, 2011.

[49] A. Krishnamurthy, S. McMains, and I. Hanniel. GPU-accelerated Hausdorff distance computation between dynamic deformable NURBS surfaces. *Comput.-Aided Design*, 43(11):1370–1379, 2011.

[50] D.D. Lee and H.S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788–791, 1999.

[51] Y. Li and R.L. Stevenson. A similarity metric for multimodal images based on modified Hausdorff distance. In *9th International Conference on Advanced Video and Signal-Based Surveillance*, pages 143–148, 2012.

[52] K.H. Lin, B. Guo, K.M. Lam, and W.C. Siu. Human face recognition using a spatially weighted modified Hausdorff distance. In *International Symposium on Intelligent Multimedia, Video and Speech Processing*, pages 477–480, 2001.

[53] P.L. Lin, Y.H. Lai, and P.W. Huang. Dental biometrics: Human identification based on teeth and dental works in bitewing radiographs. *Pattern Recogn.*, 45(3):934–946, 2012.

[54] H. Liu and L. Yu. Toward integrating feature selection algorithms for classification and clustering. *IEEE Trans. Knowledge Data Eng.*, 17(4): 491–501, 2005.

[55] Y. Liu and S. Lee. PSU near-regular texture database. http://vivid.cse.psu.edu/, accessed on Dicember 2012.

[56] R.A. Lizarraga-Morales, R.E. Sanchez-Yanez, V. Ayala-Ramirez, and F.E. Correa-Tome. Integration of color and texture cues in a rough set-based segmentation method. *J. of Elec. Imaging*, 23(2):023003, 2014.

[57] K.V. Mardia and T.J. Hansworth. A spatial thresholding method for image segmentation. *IEEE Trans. Pattern Anal. Machine Intell.*, 10(6): 919–927, 1988.

[58] S. Markovitch and D. Rosenstein. Feature generation using general constructor functions. *Machine Learning*, 49(1):59–98, 2002.

[59] 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 *8th International Conference on Computer Vision*, volume 2, pages 416–423, 2001.

[60] D.R. Martin, C.C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(5):530–549, 2004.

[61] B. Mirkin. Concept learning and feature selection based on square-error clustering. *Machine Learning*, 35(1):25–39, 1999.

[62] D.S. Modha and W.S. Spangler. Feature weighting in k-means clustering. *Machine Learning*, 52(3):217–237, 2003.

[63] L.P. Niu, X.H. Jiang, W.H. Zhang, and D.X. Shi. Image registration based on Hausdorff distance. In *International Conference on Networking and Information Technology*, pages 252–256, 2010.

[64] M.S. Nixon and A.S. Aguado. *Feature Extraction and Image Processing*. Newnes Elsevier, 2002.

[65] T. Ojala and M. Pietikäinen. Unsupervised texture segmentation using feature distributions. *Pattern Recognition*, 32(3):477–486, 1999.

[66] C. Owens. Integrating feature extraction and memory search. *Machine Learning*, 10(3):311–339, 1993.

[67] N.R. Pal and S.K. Pal. A review on image segmentation techniques. *Pattern Recogn.*, 26(9):1277–1294, 1993.

[68] C. Pantofaru and M. Hebert. A comparison of image segmentation algorithms.

[69] J. Pauli. Learning to recognize and grasp objects. *Autonomous Robots*, 5(3–4):407–420, 1998.

[70] P. Perona and J. Malik. Scale-space and edge detection using anisotropic difussion. *IEEE Trans. Pattern Anal. Machine Intell.*, 12(7):629–639, 1990.

[71] R. Polikar. Ensemble based systems in decision making. *IEEE Circ. Sys. Mag.*, 6(3):21–45, 2006.

[72] P.M. Roth and M. Winter. Survey of appearance-based methods for object recognition. Technical report, Institute for Computer Graphics and Vision, Graz University of Technology, 2008.

[73] W.J. Rucklidge. Locating objects using the Hausdorff distance. In *5th International Conference on Computer Vision*, pages 457–464, 1995.

[74] W.J. Rucklidge. Efficiently locating objects using the Hausdorff distance. *Int. J. Comput. Vis.*, 24(3):251–270, 1997.

[75] L.G. Shapiro and G.C. Stockman. *Computer Vision*. Prentice–Hall, 2001.

[76] D.G. Sim and R.H. Park. Two-dimensional object alignment based on the robust oriented Hausdorff similarity measure. *IEEE Trans. Image Process.*, 10(3):475–483, 2001.

[77] B. Sumengen. A Matlab toolbox implementing level set methods, 2005. http://barissumengen.com/level_set_methods/.

[78] N. Tsapanos, A. Tefas, N. Nikolaidis, and I. Pitas. Shape matching using a binary search tree structure of weak classifiers. *Pattern Recogn.*, 45(6):2363–2376, 2012.

[79] L. Garcia Ugarriza, E. Saber, S.R. Vantaram, V. Amuso, M. Shaw, and R. Bhaskar. Automatic image segmentation by dynamic region growth and multiresolution merging. *IEEE Trans. Image Proc.*, 18(10):2275–2288, 2009.

[80] R. Unnikrishnan, C. Pantofaru, and M. Hebert. A measure for objective evaluation of image segmentation algorithms. In *Proc. CVPR Workshop Empirical Evaluation Methods in Computer Vision*, 2005.

[81] R. Unnikrishnan, C. Pantofaru, and M. Hebert. Toward objective evaluation of image segmentation algorithms. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(6):929–944, 2007.

[82] C.J. van Rijsbergen. *Information retrieval*. Butterworths, London, 2 edition, 1979.

[83] S.R. Vantaram and E. Saber. An adaptive bayesian clustering and multivariate region merging-based technique for efficient segmentation of color images. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1077–1080, 2011.

[84] S.R. Vantaram and E. Saber. Survey of contemporary trends in color image segmentation. *Journal of Electronic Imaging*, 21(4):040901–1, 2012.

[85] S.R. Vantaram, E. Saber, S.A. Dianat, M.Q. Shaw, and R. Bhaskar. Multiresolution adaptive and progressive gradient-based color image segmentation. *J. Elect. Imag.*, 19(1):013001, 2010.

[86] V.N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.

[87] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 511–518, 2001.

[88] D. Xu. A unified approach to autofocus and alignment for pattern localization using hybrid weighted Hausdorff distance. *Pattern Recogn. Lett.*, 32(14):1747–1755, 2011.

[89] A.Y. Yang, J. Wright, Y. Ma, and S.S. Sastry. Unsupervised segmentation of natural images via lossy data compression. *Comput. Vis. Image Und.*, 110(2):212–225, 2008.

[90] C.H.T. Yang, S.H. Lai, and L.W. Chang. Hybrid image matching combining Hausdorff distance with normalized gradient matching. *Pattern Recogn.*, 40(4):1173–1181, 2007.

[91] L. Yu and H. Liu. Efficient feature selection via analysis of relevance and redundancy. *J. Mach. Learn. Res.*, 5:1205–1224, 2004.

[92] G.P. Zhang. Neural networks for classification: A survey. *IEEE Trans. Sys. Man Cyber.*, 30(4):451–462, 2000.