ANALYSIS OF IMAGE THRESHOLDING METHODS FOR THEIR APPLICATION TO AUGMENTED REALITY ENVIRONMENTS

Daniel Martín Carabias

MÁSTER EN INVESTIGACIÓN EN INFORMÁTICA. FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

Trabajo Fin Máster en Sistemas Inteligentes

Junio de 2012

Calificación: Sobresaliente (10)

Directores:

Gonzalo Pajares Martinsanz
María Guijarro Mata-García

Autorización de difusión

Daniel Martín Carabias

21 de junio de 2012

El abajo firmante, matriculado en el Máster en Investigación en Informática de la Facultad de Informática, autoriza a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a su autor el presente Trabajo Fin de Máster: "Analysis of image thresholding methods for their application to augmented reality environments", realizado durante el curso académico 2011-2012 bajo la dirección de Gonzalo Pajares Martinsanz y María Guijarro Mata-García en el Departamento de Ingeniería del Software e Inteligencia Artificial (ISIA), y a la Biblioteca de la UCM a depositarlo en el Archivo Institucional E-Prints Complutense con el objeto de incrementar la difusión, uso e impacto del trabajo en Internet y garantizar su preservación y acceso a largo plazo.

Resumen en castellano

La realidad aumentada es una reciente disciplina que superpone contenido virtual en entornos reales usando para ello diferentes tipos de información externa, siendo la geolocalización y la detección de marcadores los elementos más comunes para aportar dicha información. En este trabajo se presenta un método de conversión de imágenes de escala de grises a blanco y negro (un proceso conocido como umbralización, o binarización de imágenes) basado en la conservación de determinados momentos de naturaleza estadística. El método propuesto ha sido evaluado frente a otros algoritmos de la literatura en el contexto de las aplicaciones de realidad aumentada, en las que la precisión dentro del marcador y el rendimiento en tiempo son cruciales.

Palabras clave

Umbralización, binarización, realidad aumentada, procesamiento de imágenes, visión por computador

Abstract

Augmented reality is a recent discipline that superimposes virtual content over real environments using many different kinds of external information, geo-positioning and marker detection being the most common methods. In this work, a method to convert gray images into black and white images (a process known as image thresholding) based on moment preservation is presented and evaluated against other state-of-the-art algorithms in the context of augmented reality applications, in which in-marker accuracy and running time performance are crucial.

Keywords

Thresholding, binarization, augmented reality, image processing, computer vision

Contents

Index i

Acknowledgements iii

Dedication iv

1 Introduction 1
- 1.1 Problem Description . . . 2
- 1.2 Overview . . . 3
- 1.3 Objectives . . . 3

2 State of the Art 5
- 2.1 Histogram-shape-based Methods . . . 5
- 2.2 Clustering-based Methods . . . 7
  - 2.2.1 Iterative-based methods . . . 7
  - 2.2.2 Clustering thresholding . . . 8
  - 2.2.3 Minimum error thresholding . . . 9
  - 2.2.4 Fuzzy clustering thresholding . . . 10
- 2.3 Entropy-based Methods . . . 10
- 2.4 Attribute Similarity Methods . . . 11
  - 2.4.1 Moment preserving thresholding . . . 11
- 2.5 Locally Adaptive Thresholding . . . 12
  - 2.5.1 Local variance methods . . . 12
  - 2.5.2 Local contrast methods . . . 13
  - 2.5.3 Surface-fitting thresholding . . . 13
  - 2.5.4 Kriging method . . . 14

3 Proposed Solution 15
- 3.1 Mathematical Background . . . 17
- 3.2 Moment-preserving in Computer Vision . . . 18
- 3.3 Addition of Locality . . . 20
- 3.4 Heuristics . . . 21
- 3.5 Algorithm Steps . . . 21

4 Evaluation 23
- 4.1 Dataset . . . 23
- 4.2 Histogram Analysis . . . 24
- 4.3 Region Uniformity . . . 27

4.4 Misclassification Error 32
4.5 Computational Cost 45

5 Conclusions and Future Work **50**

Bibliography **55**

Acknowledgements

First of all, I would like to thank my supervisors Gonzalo Pajares and María Guijarro for their contribution in the elaboration of this work with countless hours of answering questions, suggestions, and corrections that have definitely improved the overall quality of this work. I would also like to thank my friends and family for their support and understanding when my free time gradually decreased to be able to complete this work on time. I extend my deepest gratitude to any other people that I could have forgotten and can identify with these words.

Dedication

To my family, for their encouragement to pursuit my dreams...

Chapter 1

Introduction

Augmented reality is a recent discipline, derived from virtual reality, that consists of superimposing computer-generated images (2D or 3D) over a live video generated by a camera. It is currently gaining popularity as portable devices such as smartphones and tablets are progressively replacing older forms of consumer computing and coincidentally are also equipped with hardware that facilitates geo-positioning and image tracking, two of the main components of augmented reality. Augmented reality software must compute real world coordinates from those images captured by the camera, so it needs to run elaborated computer vision algorithms to extract the position and rotation of a particular object ("marker") in the real scene. Among the computer vision processing that augmented reality performs, there is an important step, called "image thresholding", that segments the image in two regions: marker itself (or foreground), and the rest of the image (or background).

Image thresholding is a very important problem in automated image analysis. Many image processing and computer vision applications usually require binary images (i.e. black and white) as an introductory step in order to do further processing. By choosing a particular intensity value as "threshold", images can be segmented by setting those pixels whose original intensity is above the threshold as "white pixels", and setting the other pixels as "black pixels." Thresholding is one of the easiest methods to automatically segment an image using a computer.

1.1 Problem Description

Despite the apparent easiness of the thresholding process, it is actually a complex procedure given the vast array of circumstances and environments that are unique to each image. In fact, most applications of image thresholding are restricted to a particular domain where conditions are not as variable as in the general case. Lighting conditions are one of those factors that are unpredictable and difficult to model accurately. Resulting shadows, glitter, and related artifacts suppose a challenge for computer vision algorithms, in particular for image thresholding techniques, the ground of this research.

These difficulties are more aggravating in real-time applications with strict delay requirements because they impose an upper bound on the time complexity of the algorithms involved. Not only they should be fast, but they also need to be low on memory requirements, because augmented reality applications usually run on hand-held devices that do not have as much main memory as a desktop computer has.

This research focuses on the augmented reality applications that use fiduciary trackers to help in positioning. In imaging systems, a fiducial, or fiduciary marker, is an object placed in the field of view of an imaging system for use as a point of reference. Fiduciary markers have been extensively used in fields like geography until the introduction of the GPS, where this kind of trackers helped in the measurement of surfaces and other terrestrial features by using aerial photography. Other prominent applications of fiducials are in medicine, where markers are placed in the area captured by two image systems to help correlate them. In radiology, markers are landmarks in tumours that help establish correct targets for treatment.

Recently, two disciplines have used fiducials with success. Virtual reality and augmented reality can recognize objects in the scene by placing in it a fiduciary tracker that is extracted by sophisticated computer vision libraries. One of those libraries for augmented reality is ARToolkit [1], available as a free open source library and also as a commercial product. This

---

[1] http://www.hitl.washington.edu/artoolkit/

library has been selected in this work as a reference for multiple reasons: There is a free and open source version available for everyone to explore and use without restriction. It also has a simple application programming interface (API) in C and it is one of the most popular augmented reality libraries available, commonly used in desktop and mobile applications.

1.2 Overview

This document is organized in the following chapters:

Chapter 2 presents a comprehensive study of the state-of-the-art regarding thresholding techniques.

Chapter 3 shows the proposed solution with a detailed explanation of the involved methods and techniques.

Chapter 4 documents the evaluation and testing of the proposed method, comparing it with current alternatives available in the literature.

Chapter 5 reveals the conclusions that are drawn from this work and proposes future lines of work for further research.

1.3 Objectives

Given the aforementioned description of the problem, this research focuses on the analysis and evaluation of a proposed image thresholding technique and a comparison against other techniques available in the literature, in the context of augmented reality applications.

The main requirement that is expected from the solution is a reasonably good quality in the results, where the next chapters will explain what can be understood by “good quality” in image thresholding. This is expected to imply an accurate recognition of fiducials by the library when they are used in real-world situations.

Another requirement, also very important, is a good running time performance for its successful application to streams of video.

The above two requirements are in contraposition so there must be a sensible analysis and consideration of multiple parameters. Thus, the algorithm must be flexible enough to deal with both requisites.

Chapter 2

State of the Art

Image thresholding, a method which extracts objects in an image from the background, is one of the most common operations in image processing and, as such, it has been extensively researched by computer vision experts. In Sezgin and Sankur [25], thresholding methods are categorized according to the information they obtain from the data:

- Histogram-shaped-based
- Clustering-based
- Entropy-based
- Attribute similarity methods
- Object attribute-based
- Spatial approaches
- Local methods

2.1 Histogram-shape-based Methods

This group of thresholding methods is based on the form and shape properties of image histograms. Rosenfeld and de la Torre [22] and Lee et al. [10] used histogram concavity analysis to derive the optimal threshold value for a given image. In their paper, a convex hull of the image histogram is calculated and the deepest concavity points are selected as candidates to be the threshold value. In order to make computations faster, ad-hoc hardware units are designed that are able to compute a value for a standard-sized frame in about 10 milliseconds. In Seazan [24], the histogram function is convolved with a smoothing kernel. The gray levels where the peaks start, end and attain a maxima are estimated. To add some data reduction, the algorithm sets gray-level thresholds between the peaks, and the gray levels at which the peaks attain a maximum are chosen as quantization levels.

In Chang et al. [3], a multi-modal histogram thresholding method is proposed based on a combination of regularization and statistical approaches. The original histogram is decomposed in several non-overlapping distributions by modeling it with a mixture of Gaussian density. Although the histogram is contaminated by contiguous distributions, experiments shown on the paper with simulated data demonstrate that this method outperforms similar ones at the task of finding the best threshold values and the parameters of predefined distributions. It is also a good option for real-world images as they generally do not come as a Gaussian mixture of densities. The only caveat in this paper is the heuristic nature of the smoothness factor, which would generate different thresholding values depending on the particular election. There is opportunity for future work regarding smoothness factors for different gray-level images.

Ramesh et al. [21] were among the first who used a simple two-step approximation function to the normalized histogram. Thus, the sum of squares between the function and the histogram is minimized and the optimal threshold value can be obtained by performing an iterative search.

Prewitt and Mendelsohn [20], in their analysis of cell images, proposed a method that iteratively smoothed the histogram using a running average of size 3, until there is two local maxima, j and k. The final threshold t is computed as the average, $(j + k)/2$.

2.2 Clustering-based Methods

These methods rely on generating clusters from gray-level information. Because the end result is a binary image, there are only two classes, or clusters. Each cluster corresponds to a lobe of the histogram. There are several clustering approaches which can be further subclassified as follows:

2.2.1 Iterative-based methods

Iterative schemes are based on the mixture of two Gaussian-based models. The basic theory behind iterative thresholding can be summarized as follows:

1. Choose an initial threshold (T), either randomly or using another method.
2. Segment the image as background and foreground according to this initial threshold.
    (a) $G_1 = \{f(x,y) : f(x,y) > T\}$
    (b) $G_2 = \{f(x,y) : f(x,y) <= T\}$
3. Compute the average of each set. Let m_1 be the average of G_1 and m_2 be the average of G_2.
4. A new threshold, T', is calculated from the average of m_1 and m_2.
5. Repeat step 2 until the difference between T and T' is small enough.

This method is known as Calvard et al. [2], and has undergone several modifications and improvements from the research community. One of those modifications was developed by Yanni and E. [29]. In their work, the proposed iterative algorithm is initialized to the midpoint between the two peaks of the histogram, the highest gray level and the lowest one.

2.2.2 Clustering thresholding

In this category we can fit in the Otsu's method (Otsu [17]), one the most referenced thresholding methods in the literature.

Otsu's method is based on selecting a threshold for separating the image into two classes so that the variance within each class is minimized. For obvious reasons, the distributions cannot be changed, but the selection of a threshold value modifies the spread of the two parts of the distribution. The goal is to select a threshold that minimizes the combined spread.

The *within-class* variance can be defined as the weighed sum of the variances of each cluster:

$$\sigma^2_{within}(T) = n_B(T)\sigma^2_B(T) + n_F(T)\sigma^2_F(T) \tag{2.1}$$

where:

$$n_B(T) = \sum_{i=0}^{T-1} p(i) \tag{2.2}$$

$$n_F(T) = \sum_{i=T}^{N-1} p(i) \tag{2.3}$$

$$\sigma^2_B(T) = \text{The variance of background pixels.} \tag{2.4}$$

$$\sigma^2_F(T) = \text{The variance of foreground pixels.} \tag{2.5}$$

The above equations require the computation of *within-class* variance for each class and for each possible thresholding value, resulting in a very expensive computation that must be avoided. The key observation to reduce computation cost is that the calculation of *between-class* variance is a less expensive step and it can be defined as the *within-class* variance subtracted from the total variance.

$$\sigma^2_{Between}(T) = \sigma^2 - \sigma^2_{Within}(T) = n_B(T)[\mu_B(T) - \mu]^2 + n_O(T)[\mu_O(T) - \mu]^2 \tag{2.6}$$

where σ^2 is the combined variance and μ is the combined mean. The *between-class* variance is the weighted variance of the cluster means around the overall mean. If we substitute $\mu = n_B(T)\mu_B(T) + n_O(T)\mu_O(T)$ and simplify the result, we get

$$\sigma^2_{Between}(T) = n_B(T)n_O(T)[\mu_B(T) - \mu_O(T)]^2 \tag{2.7}$$

So, for each potential threshold, the algorithm separates the pixels into two clusters according to the value. Then, it finds the mean of each cluster and square the difference between the means. Lastly, it multiplies the number of pixels in one cluster times the number in the other.

It turns out that the computations for each threshold are not independent as we iterate over the potential thresholds, so the algorithm is efficient.

2.2.3 Minimum error thresholding

Another way to minimize the error in classification is to suppose that each group or cluster is Gaussian-distributed with a mean and variance independent of the chosen threshold.

Whereas the Otsu's method separates the image into two clusters according to the threshold and then try to optimize some statistical measures, minimum error thresholding methods suppose there is a distribution and we have to estimate its parameters. If the two distributions are well separated (there could be a little overlap), we can reasonably assume that if we choose an arbitrary threshold, the mean and standard deviation of each group should approximate the mean and standard deviation of the underlying populations.

The optimal threshold can be characterized as the one that causes the mixture of the two Gaussians to better approximate the real histogram. To reduce the solution space of this problem, a gradient descent method is used from an initial guess. Otsu's method can be a good technique for estimating that initial guess.

Kittler and Illingworth [9] formulated one of the principal minimum error methods. In recent papers, Cho et al. [5] have suggested some improvements by detecting a bias that affects the threshold calculation.

2.2.4 Fuzzy clustering thresholding

Ramesh et al. [21] approach the thresholding problem by assigning fuzzy clustering memberships to pixels in an image depending on the differences between the mean of the two classes. In their work two thresholding schemes are proposed. The first one is based on minimizing the variance of the approximated histogram. The second one is based on minimizing the sum of square errors. Experimental results show that the former scheme gives better results on average than the latter one, at the expense of a small additional computational cost.

2.3 Entropy-based Methods

In this category we can enclose methods that use the entropy of the distribution of gray levels in the picture. The entropy is a concept that comes from the second law of Thermodynamics and measures spontaneous dispersal of energy. It was later introduced to communications theory by Shannon as a measure of the efficiency in data transmission over a noisy channel. A high entropy is indicative of a great information transfer. The opposite consideration, preservation of information, can be achieved by minimizing cross-entropy between the input, gray level image and the output, thresholded image.

Kapur et al. [8] used Shannon's concept of entropy, from a different point of view. They considered the background and the foreground as two different image signals. Each of those classes have their entropy calculated and summed, so that when the sum is maximum the threshold is considered optimal. The probability distribution of the gray levels over the black part of the image is

$$p_0/P_B, p_1/P_B, ..., p_s/P_B \tag{2.8}$$

and for the white part:

$$(p_{s+1})/1 - P_B, (p_{s+2})/(1 - P_B), ..., (p_{n-1})/(1 - P_B) \tag{2.9}$$

where s is the threshold, p_i is the probability of pixels with gray level i and P_B is the probability of gray level less than or equal to the threshold.

$$P_B = \sum_{i=0}^{s} p_i \tag{2.10}$$

The entropy of the object of the image is

$$H_B = -\sum_{i=0}^{s} p_i/P_B log(p_i/P_B) \tag{2.11}$$

The entropy of the background is

$$H_W = -\sum_{i=s+1}^{n-1} p_i/(1 - P_B) log(p_i/(1 - P_B)) \tag{2.12}$$

The threshold s is selected such that the total entropy, $H_B + H_W$, is maximized.

Li and Tam [12] approached the thresholding problem by minimizing cross entropy, a measure of information theoretic distance. It is minimized under the constraint that observed and reconstructed images have identical average intensity in their foreground and background.

2.4 Attribute Similarity Methods

These methods extract a threshold value based on similarity between the original image and the binarized one using some attribute quality or similarity measure. Some of those measures are gray-level moments and fuzzy measures.

2.4.1 Moment preserving thresholding

In Tsai's work (Tsai [26]), the notion of similarity between source and target images is suggested by considering the source, gray-level image, a blurred version of an ideal binary image. The thresholding is computed by matching the first three gray-level moments with the first three moments of the binary image. Further improvements by Cheng and Tsai [4] used neural networks as an aid to the algorithm.

2.5 Locally Adaptive Thresholding

This class of thresholding algorithms calculates a threshold value for each pixel, depending on some local parameters like range, variance or surface-fitting in the neighborhood.

Among the first adaptive thresholding techniques was the algorithm developed by Nakagawa and Rosenfeld [14], who proposed a variation on the original method of variable thresholding by Chow and Kaneko [6]. In their study, the image is divided into several windows, or subsets of the original image with the locality property. Those windows with bimodal histograms are selected, and a threshold is calculated. Threshold from different windows are interpolated to calculate a threshold for the whole image.

The above method was successfully applied to TV images of machine components, with results substantially better than those that applied a fixed threshold to the whole image. A further extension allowed for trimodal histograms in the computation of the threshold, which yielded better results with the negative point of being more sensitive to shadows.

2.5.1 Local variance methods

Niblack [15] developed an algorithm that adapts threshold selection to local mean $m(i,j)$ and standard deviation $\sigma(i,j)$ and a local window of size bxb. This method proved to work well for optical character recognition (OCR) tasks. Further improvements introduced by Sauvola and Pietikäinen [23] changed the impact of the standard deviation in the algorithm to better recognize letters in stained or documents with bad illumination.

2.5.2 Local contrast methods

White and Rohrer [28] suggested comparing the gray level of a pixel with the average of gray levels of their neighbor pixels. For their experiments they used a window which approximates roughly to the size of a printed character. If a pixel is significantly darker than the average, it can be classified as a character pixel; otherwise, it is a background pixel. Venkateswarlu and Boyle [27] are a great reference for performance comparison of local adaptive methods.

Huang's method (Huang and Wang [7]) first smooths the image by averaging the gray level of a pixel with their local neighbors, provided that the range (difference between maximum and minimum gray levels within the window) is below a given threshold T_1. Next, an adaptive threshold is applied, which sets a pixel to the maximum value if it is greater than the local average, or if the local range is below a threshold T_2.

2.5.3 Surface-fitting thresholding

In Yanowitz's method (Yanowitz and Bruckstein [30]), gray-level information is combined with edge information to build a threshold surface. This method that exploits geometric features of blueprint images. It works under the assumption that the objects have a "c" shape, and the area is small. Multiple window sizes are proposed in the work, which effectively reduces computation time and helps distinguish thin lines from thick lines. Although the method is targeted at blueprint images, results indicate that it is also good for a wide range of images. The threshold surface is constructed by interpolation with potential surface functions and it is obtained iteratively using a discrete Laplacian on the surface.

Other clever methods, specially used for badly illuminated images, include the one proposed by Parker [18]. This method involves a first step where objects are located in the scene by using an intensity gradient. Next, levels that correspond to the objects in the various parts of the image are used as initial guesses for the threshold. Overall, Parker's method outperforms many classical adaptive threshold algorithms when applied on images produced with variable illumination.

2.5.4 Kriging method

Oh's method (Oh and Lindquist [16]) is a two-pass algorithm that firstly uses a non-local thresholding algorithm such as Kapur et al. [8] to classify the majority of the pixel population in one of two classes (object and background). Then, a variation of Kapur's method is applied, in which a lower threshold is established, below which gray levels are assigned to the first class (for example, the object class). Then, a second, larger, threshold value is found such that any pixel with a greater gray level is automatically assigned to the second class (for example, the background class). The remaining pixels whose gray level values lie between the two thresholds are left to the second pass. In the second pass, also known as the indicator kriging stage, pixels are assigned one of the two classes using local covariance of the class indicators and the constrained linear regression technique called kriging, within a region.

Chapter 3

Proposed Solution

As described in the first chapter, image thresholding is far from being considered a solved problem. The research community is tackling this problem for small and concrete domains, chiefly because an optimal solution for every environment is regarded as very unlikely. One of those small domains that resembles the augmented reality domain is optical character recognition (OCR). OCR software makes use of sophisticated computer vision algorithms that depend on the computer being able to separate characters (objects) from the paper (background). Complicated illumination and extreme conditions, usual in ancient documents, makes up an environment that is as hard to analyze and as unpredictable as the environment that augmented reality software has to deal with.

In the explanation of the current state-of-the-art in thresholding techniques we can argue that every algorithm can be roughly classified as a global method, i.e. one that extracts a single threshold value for the whole image, or a local -or dynamic- method, i.e. one that extracts several thresholding values that segment parts of the input image.

There have been several studies of the performance of the existing fiducial-trackers used in augmented reality. One recent study was conducted by Naimark and Foxlin [13], where it was pointed out that an important source of problems during the tracking of fiduciary markers is the lack of robustness of global thresholding methods when lighting conditions cause shadows and reflections off the marker's surface. More recently, a paper by Pintaric [19] proposed a new dynamic method based on regions of interest (ROIs) around the marker once it has been detected. Results from their work show that it outperforms global thresholding specially when markers experiment little movement relative to the camera.

Given the aforementioned studies and the special conditions of augmented reality, for this work it was decided to think about a variation of a dynamic method. They offer a better performance in many situations because the non-homogeneous gray-levels of pictures with real-world lighting are amenable to different threshold values depending on the gray-levels of a neighborhood around a pixel. The commercial augmented reality library ARToolkit implements several global thresholding algorithms, and recently also a local adaptive method based on Pintaric's paper. Empirically, the adaptive method gives more consistent results and is also able to react to feedback from the environment by recalculating a new global threshold value.

Despite the good results of local thresholding methods, calculation of threshold values for regions of an image is a computational expensive procedure: First of all, the algorithm must loop through the image matrix and, for every pixel in it, loop through a sub-image, or window around it, performing some additional calculations that in the best case are in O(1) time. This gives an algorithm that is bounded by $\Omega(N * M * P)$, where N is the number of rows in the input image, M is the number of columns, and P is the number of pixels inside the window. As the number of pixels inside a particular window is relatively small compared to the total number of pixels in the image, the lower bound complexity of this particular algorithm is quadratic in the minimum between the image width and the image height. For small images captured by old VGA-based cameras still available in mobile devices, this complexity is not crucial given the availability of ad-hoc GPUs inside modern hand-held devices that deals with matrix operations with efficiency.

Even if regions of interest are used, there is still at least one frame for which the entire image must be analyzed, so improvements at this stage are worthy. The arising problem that the computer vision community faces is that mobile cameras are getting better each year, and users expect a high level of realism and fidelity while interacting with an augmented reality application. In this situation, local thresholding methods can slow down the application to the point that it is unable to produce images at a 24 fps rate, so lag becomes noticeable. In conclusion, there is a trade-off between this two groups because global methods are quick but results are not as good as those produced by local methods.

In the proposed solution for the thresholding problem in augmented reality applications, we suggest considering an "hybrid" approach that uses the concept of "window" around a particular pixel; but at the same time calculates some values depending on the image histogram, in order to improve the robustness of the process. Because of time restrictions, calculating a threshold for every pixel in the image is not affordable, so the image is also partitioned in classes with each class assigned a particular threshold value. The theoretical background and details of this process are explained in the following sections.

3.1 Mathematical Background

In mathematics, the shape of a set of points can be described as a quantitative measure known as *moment*. More precisely, given a function f(x) and a real value c, we can define its moment as

$$\mu_n = \int_{\infty}^{\infty} (x-c)^n f(x)\,\mathrm{d}x \tag{3.1}$$

The first moment, μ_1, the mean, is usually simply denoted μ. A statistical distribution P(x) is not uniquely specified by its moments, and they are most usually taken about the mean. These *central moments* are denoted μ_n and defined by

$$\mu_n = \int_{\infty}^{\infty} (x-\mu)^n f(x)\,\mathrm{d}x \tag{3.2}$$

The second moment about the mean is equal to the variance, that is,

$$\mu_2 = \sigma^2 \tag{3.3}$$

The third central moment is the skewness, a measure of the symmetry of the distribution.

The fourth moment is a measure of whether the distribution is tall and skinny compared to a normal distribution of the same variance. As it is a fourth power, it is always positive. A relative concept, kurtosis, is defined as the normalized fourth moment minus 3.

3.2 Moment-preserving in Computer Vision

In computer vision, a moment is a weighted average of the image pixels' intensities, usually chosen because they are believed to have interesting and attractive properties. In moment-preserving thresholding, values are chosen so that gray-level moments of an input image are preserved in the output image. The aim of this moment-preserving transformation is to group different pixel intensities into a number of classes. Each class of values can be represented by a single gray-level value. Given an image f whose pixels can be represented as f(x,y), the ith moment m_i of f is defined as

$$m_i = 1/n \sum_x \sum_y f^i(x, y) \tag{3.4}$$

Where n is the number of pixels in the image.

Gray moments can be computed from the histogram of an image f as follows:

$$m_i = 1/n \sum_{j=0}^{255} h(j) * j^i \tag{3.5}$$

Where n is the total number of pixels and *h(j)* is the number of pixels with gray level equal to j.

For bilevel thresholding, a single T threshold is used to segment the image into two classes: the background class and the foreground class. If z_0 and z_1 are the mean gray levels of each of those classes and p_0 and p_1 are the the fraction of pixels below T and above T, respectively, then the moments for the thresholded image are given by:

$$m'_i = \sum_{k=0}^{1} p_k (z_k)^i \tag{3.6}$$

The moments of the original image are preserved by equating m_i and m'_i (for i=0,1,2,3). This yields four equalities:

$$p_0 + p_1 = 1 \tag{3.7}$$

$$p_0 z_0 + p_1 z_1 = m_1 \tag{3.8}$$

$$p_0 (z_0)^2 + p_1 (z_1)^2 = m_2 \tag{3.9}$$

$$p_0 (z_0)^3 + p_1 (z_1)^3 = m_3 \tag{3.10}$$

Equations 3.7, 3.8, 3.9 and 3.10 have to be solved in order to obtain p_0 and p_1. The solution explained here is adapted from the one presented in the appendix of "Introduction to digital image processing", by Niblack [15].

The threshold T can be calculated by determining the p_0-tile from

$$p_0 = \sum_{j=0}^{T} P_j \tag{3.11}$$

Expressions for solving p_0 and p_1 can be obtained by rewriting equations 3.7 and 3.8 in matrix format. By solving those expression we get that

$$p_0 = (z_1 - m_1)/(z_1 - z_0) \tag{3.12}$$

Tsai showed that the moment preserving equations can be solved by following two steps:

First step: Using moment values m_0, m_1, m_2 and auxiliary values c_0, c_1, c_2 a set of linear equations can be established:

$$c_0 m_0 + c_1 m_1 + c_2 m_2 = 0 \tag{3.13}$$

$$c_0 m_1 + c_1 m_2 + c_2 m_3 = 0 \tag{3.14}$$

Second step: Setting c_2 to 1 and applying Cramer's rule, unknowns c_0 and c_1 can be obtained.

The gray levels z_0 and z_1 can be obtained by solving this second-order equation:

$$z^2 + c_1 z + c_0 = 0 \tag{3.15}$$

yielding two values for z.

3.3 Addition of Locality

As it was commented before, a simple threshold value applied to the entire image does not give high quality results when the distribution of gray-level values is highly non-homogeneous. The next chapter will delve deep into the available thresholding techniques performance, but, in short, some local algorithms define a neighborhood around the pixel (a window) that influence the threshold value for that particular pixel. For performance reasons, a window around a pixel is used in the proposed algorithm, but it computes a threshold value for every pixel on that window. This hybrid approach was chosen to balance a good running time with the best possible results.

A window of size equal to the entire image would be equivalent to a global thresholding algorithm, in particular one of those that are based on moment preservation (see chapter 2). Experimentation showed that a very small window does not imply better results in every case; principally because small windows reduce the amount of histogram information and, consequently, the statistical significance of the measures proposed. Chapter 4 analyses the impact of window size on the thresholding performance and discusses this observation.

In addition to moment preserving calculation, each window produces another threshold value based on the sum of gray levels on it and the sum of squares of gray-level values. This is an approach similar to some available dynamic methods, like Niblack's method (Niblack [15]).

3.4 Heuristics

Once the local threshold value based on moment preservation and the local threshold value based on gray-level mean and variance are calculated, a weighted sum heuristic is used to compute the final threshold value:

$$threshold = k_g * threshold\_moments + k_l * threshold\_local \tag{3.16}$$

The impact of the above constant weights (k_g, k_l) in the results is analyzed in chapter 4.

3.5 Algorithm Steps

To sum up, in the proposed algorithm a moment-preserving technique is applied to subsets of the input image pixels. At the same time, local mean and local variance are computed for each window to derive an hybrid threshold value. The specific steps of the algorithm are:

1. Loop through the image and loop through a neighborhood around the center pixel of a previously defined window.

2. Compute the normalized histogram of that window, the sum of gray levels, and the sum of squares of gray level values.

3. Compute the first three gray-level moments according to equations 3.2 and 3.3 (the zeroth moment is assumed to be 1.0) and calculate p_0, the fraction of object pixels in the target image according to equation 3.12. Then a threshold value is computed using p_0, such that the fraction of image pixels are below this value.

4. Compute the sum of local mean, and the square root of the local variance, times a constant k, an input to be provided to the algorithm (0.2 is a suggested value according to Niblack's work).
5. Compute the final threshold value for the window as the weighted sum of threshold values in steps 3 and 4.

In the following chapter, a comprehensive study and evaluation of current thresholding techniques and the comparison with the proposed algorithm are presented.

Chapter 4

Evaluation

In this chapter we are going to assess the performance of some thresholding techniques in the context of augmented reality. Previous analysis and evaluation of image thresholding showed that thresholding techniques can encounter difficulties when the foreground object in the image occupies a large or small area or when gray levels are overlapping, resulting in a near unimodal distribution. Ideal testing criteria should take into account both the noisiness of the picture as well as the fidelity of shapes, a factor that is very important because augmented reality algorithms that use markers usually perform a line detection step that helps in the estimation of marker position and orientation with respect to the camera. In order to assess and compare the performance of different thresholding techniques, a set of experiments has been conducted and their results are explained and discussed in this chapter.

4.1 Dataset

The test set of images were 30 images containing specific markers used on augmented reality that have been trained using the ARToolkit library. Figure 4.1 shows some examples of the images used in the experiment. Markers were placed in real-world situations that may be classified as dark background images, where the marker was placed on a theater for an augmented reality show; images with backgrounds similar to the marker itself, where the marker was placed on patterned blankets, and images with an uneven lightning, where the marker was placed under the influence of a direct light source, near a window.

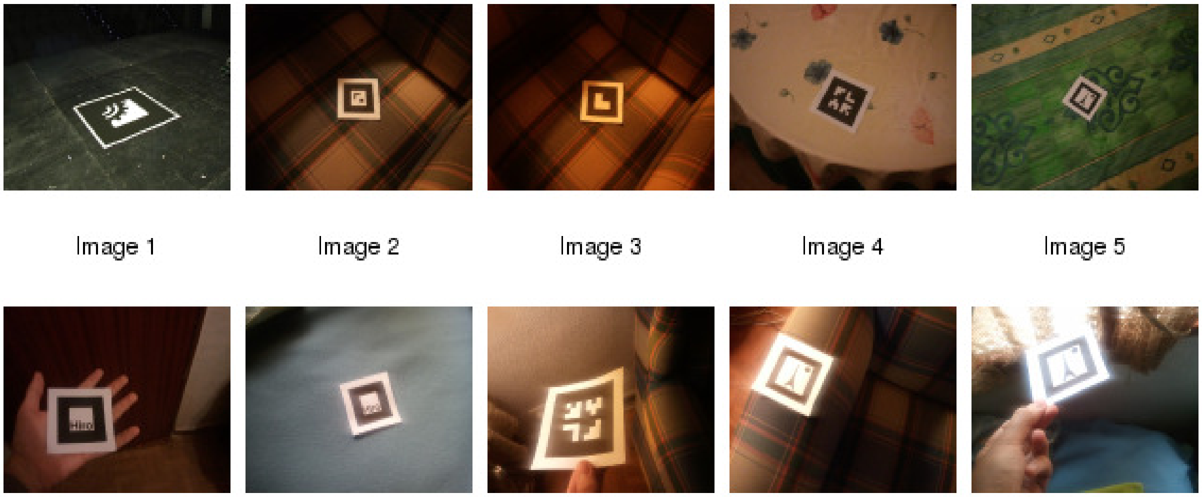


Image 1 Image 2 Image 3 Image 4 Image 5

Image 6 Image 7 Image 8 Image 9 Image 10

Figure 4.1: *Sample of 10 of the 30 ARToolkit-trained markers in real-world situations that were used as input set in the evaluation.*

4.2 Histogram Analysis

As a preliminary step to a more in-depth thresholding analysis, the histograms of the image set were extracted. Image histograms are a useful tool to help discover some properties from images, and even directly obtain thresholds from them. For example, given a simple image with a light object over a dark background, its histogram would be conformed by two predominant peaks (i.e. a bimodal histogram), so one thresholding value would be enough to correctly segment the image. A morphological analysis of histograms can help predict which thresholding method may offer the best performance for a particular image by reasoning about some assumptions of the thresholding method itself. Figure 4.3 shows the gray-level images and their associated histograms.

However, morphological analysis alone is often not sufficient for discovering useful properties in a gray-level image. It has been proposed several histogram metrics in the image

	Mean	**Variance**	**Skewness**	**Entropy**	**Kurtosis**
Image 1	45.175	1505.349	207928.992	6.232	20.087
Image 2	36.661	1299.116	86713.207	6.486	7.949
Image 3	36.351	1270.613	80484.589	6.445	7.432
Image 4	90.061	1440.517	-59725.406	6.527	2.644
Image 5	62.410	503.700	13352.520	6.283	6.480
Image 6	37.092	1238.962	68290.513	6.194	5.080
Image 7	111.712	1616.260	-9095.314	6.966	4.991
Image 8	85.682	3992.031	151729.104	7.591	2.811
Image 9	65.396	4735.375	460919.413	7.006	4.150
Image 10	107.616	5864.732	367695.639	7.218	2.313

Table 4.1: *Histogram statistical properties for a subset of images in the test set.*

processing literature. As histograms are a statistical concept, it makes sense to refer to the average, variance, skewness, entropy and kurtosis of them. Table 4.1 sums up this metrics for the gray-level input images in the experiment. A brief description of these concepts and their significance in image processing follows.

The *mean* of a data set, and in particular of an image histogram, is the arithmetic average of the values in the set, obtained by summing all values and dividing by the number of them. The mean is, thus, a measure of the center of the distribution. The mean is a weighted average where the weight factors are the relative frequencies. It was calculated for the histograms of this experiment because it provides information about the brightness level of an image. In the case of images 10 and 7, with the highest mean value of the test set as figure 4.2, it is indicative that predominant white gray levels exist.

The *variance* of a data set is the arithmetic average of the squared differences between the values and the mean. The *standard deviation* is the squared root of the variance. The variance and the standard deviation are both measures of the spread of the distribution around the mean.

The *skewness* is a measure of the asymmetry of the probability distribution of a real-valued random variable. Skewness can be a positive or a negative value. Images like 4 and 7 with a negative skewness indicate that the bulk of values lie to the right of the mean.

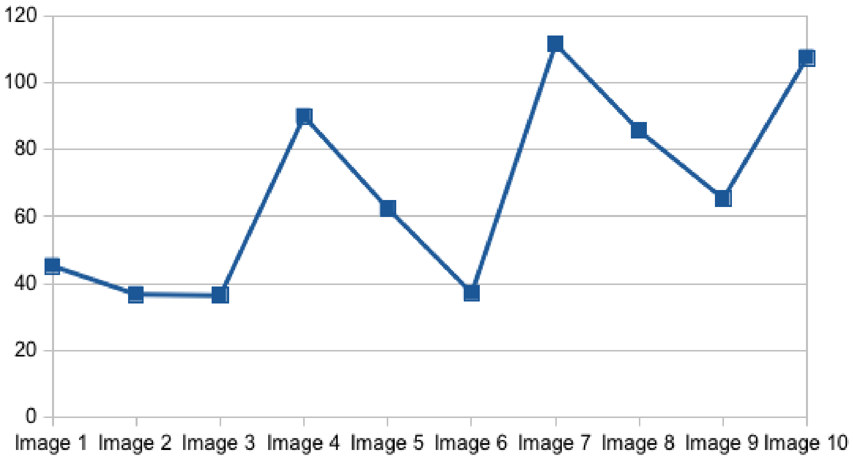


Figure 4.2: *Graphic of mean values of the image set histograms.*

Therefore, most pixels in the image have gray-level values close to white (i.e. the image is brighter). In contrast, images with a positive skewness, indicate that the bulk of values lie to the left of the mean, so the image is darker.

The *entropy* of a grayscale image is a statistical measure of randomness that can be used to characterize the texture of the input image. It can defined as:

$$-\sum(p * log_2(p)) \tag{4.1}$$

where p is equal to the count of pixels for a particular gray level divided by the total number of pixels.

Therefore, if the image is a single grayscale, entropy is 0; if it is an uniform gradient including all values from 0 to 255 equally populated in the histogram, entropy is 1.

Once the test image histograms were analyzed, a modification to the algorithm was considered in which histogram properties were extracted and one of multiple thresholding methods was applied depending on those characteristics. This approach was ruled out because the running time of the algorithm was too high. From the results point of view, performance tests were conducted and their conclusions are summarized in the following

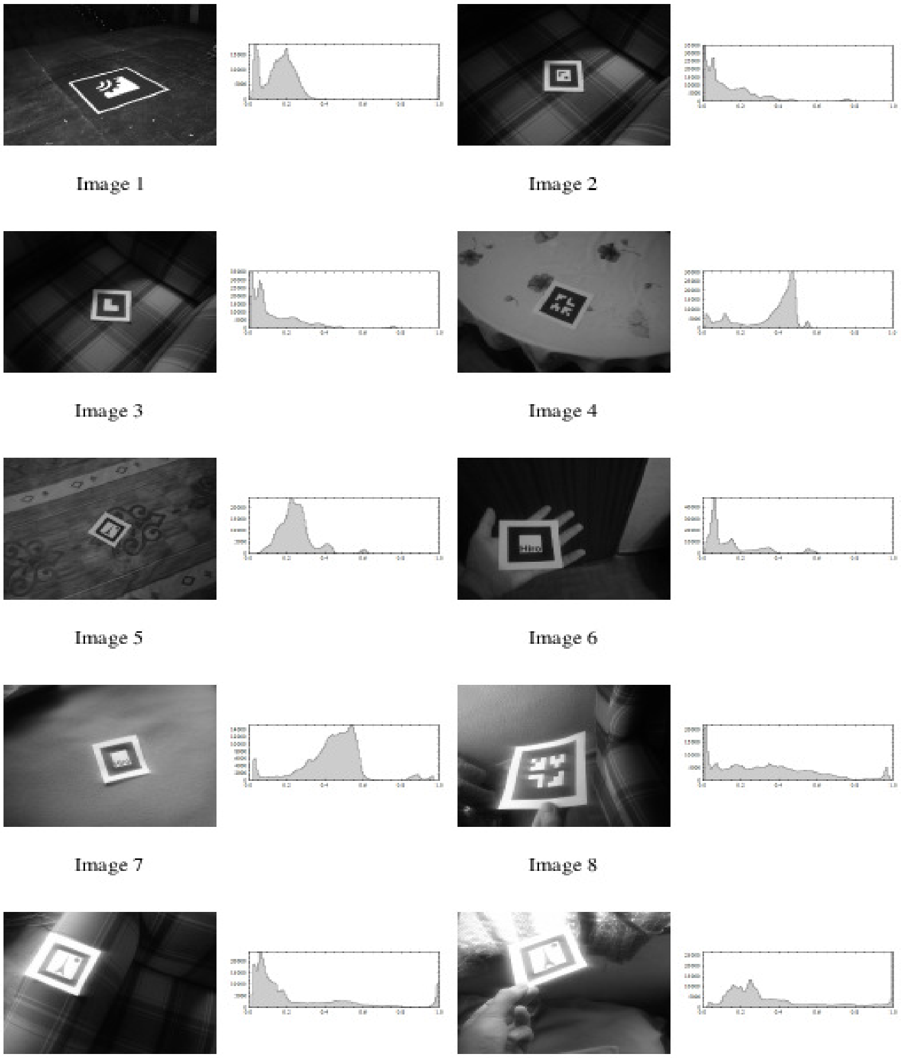


Image 1
Image 2
Image 3
Image 4
Image 5
Image 6
Image 7
Image 8
Image 9
Image 10

Figure 4.3: *Image histograms of the sample images in Fig 4.1.*

sections.

4.3 Region Uniformity

In the first evaluation of thresholding techniques performance, region uniformity was considered. Uniformity of a region is a concept that is inversely proportional to the variance of the values of that feature at every pixel belonging to that region. A large variance is indicative of a great spread from the mean of the feature value across the region. Region uniformity is considered by [11] as a method to assess thresholding performance. More precisely, considering $f(x, y)$ a function that represents, for each pixel (x, y), its gray-level value; R_i represents the segmented region i (where i can be 1 or 2, namely, object or background). For a given threshold value t, the non-uniformity measure $U(t)$ is

$$u(t) = (\sigma_1^2 + \sigma_2^2)/c \tag{4.2}$$

where

$$\sigma_i^2 = \sum_{(x,y)r_i} (f(x, y) - \mu_i)^2 \tag{4.3}$$

$$\mu_i = \sum_{(x,y)r_i} f(x, y)/a_i \tag{4.4}$$

and c is a normalization factor.

According to the previous definition, a binary (thresholded) image is more uniform the lower u(t) is.

In the experiment, 16 global thresholding methods and 5 local thresholding methods that were explained in detail in chapter 2 were selected: Huang, Prewitt, Ridler-Calvard, Li, Kapur, Mean, Kittler-Illingworth, Minimum, Tsai, Otsu, Doyle, Renyi, Shanbhag, Triangle, and Yen as global methods, and Bernsen, Mean, Median, MidGrey, Niblack, and Sauvola as local methods. There are many efficient implementations of those algorithms in almost every programming language. As region uniformity evaluation does not require an optimal implementation, standard MATLAB programs were developed and the resulting binary image was used as input to an algorithm responsible for computing region uniformity according to Levine et al. criteria explained above. The results are shown in Tables 4.2 and 4.3 for global methods, and Table 4.4 for local ones.

Uniformity measures range from 0 for a perfect uniform image to 100 for a completely heterogeneous image. From the set of images used for evaluation, the most uniform binary

	Huang	**Prewitt**	**Ridler**	**Li**	**Kapur**	**Mean**	**Kittler**	**Minimum**
Image 1	12.688	32.197	32.177	17.567	34.445	20.878	14.968	31.557
Image 2	23.949	39.384	28.745	25.281	38.538	25.281	21.950	38.198
Image 3	22.848	37.293	27.062	24.565	36.873	24.079	21.421	36.483
Image 4	25.917	28.858	28.440	30.541	25.791	25.808	25.808	30.618
Image 5	19.515	30.509	28.572	22.832	30.674	23.237	26.819	28.845
Image 6	23.605	31.383	29.412	27.012	27.012	23.605	22.340	28.316
Image 7	35.473	55.687	37.075	37.811	56.284	35.813	42.432	55.678
Image 8	42.957	42.384	43.804	43.042	47.150	43.226	41.754	41.707
Image 9	32.872	32.477	33.121	32.757	32.668	32.714	31.114	32.948
Image 10	44.360	44.152	44.152	44.532	47.524	44.815	44.815	34.443

Table 4.2: *Region uniformity U(t) for a subset of test images (global methods)*

	Tsai	**Otsu**	**Doyle**	**Renyi**	**Shanbhag**	**Triangle**	**Yen**
Image 1	34.458	32.177	19.040	26.021	19.040	12.313	19.124
Image 2	34.031	29.037	21.767	38.538	34.031	26.175	38.538
Image 3	31.857	27.062	21.580	33.103	33.103	22.071	32.781
Image 4	27.186	28.440	36.116	26.032	30.356	27.186	24.739
Image 5	29.577	30.076	21.455	30.674	23.594	30.813	30.674
Image 6	29.987	29.412	22.935	27.327	31.102	21.500	26.103
Image 7	36.556	35.823	37.226	56.284	35.638	39.444	56.284
Image 8	45.462	44.210	43.175	46.869	46.172	40.060	47.171
Image 9	33.366	33.121	30.166	33.483	33.551	32.381	31.971
Image 10	44.532	44.152	47.531	46.791	41.702	33.124	41.702

Table 4.3: *Region uniformity U(t) for a subset of test images (global methods)*

	Bernsen	**Mean**	**Median**	**MidGrey**	**Niblack**	**Sauvola**	**Proposed**
Image 1	29.617	27.367	25.469	28.279	27.111	25.620	**23.523**
Image 2	34.453	37.164	36.688	36.497	37.298	36.981	**32.234**
Image 3	33.034	36.892	36.414	35.314	37.003	36.986	**21.554**
Image 4	65.912	65.601	69.161	61.586	65.786	67.659	**62.231**
Image 5	29.220	27.552	26.553	29.061	28.842	27.020	**33.454**
Image 6	28.017	31.297	30.186	31.766	31.125	29.827	**23.656**
Image 7	47.832	49.914	50.806	46.146	50.605	51.007	**54.867**
Image 8	52.336	51.175	48.958	52.654	51.401	48.866	**45.756**
Image 9	36.423	37.223	36.826	37.354	37.531	36.842	**33.789**
Image 10	42.739	40.210	39.697	42.006	40.303	39.255	**35.342**

Table 4.4: *Region uniformity U(t) for a subset of test images (dynamic methods)*

image was produced by Triangle's method in a 34% of cases; Huang's method in a 21%; Doyle's in a 17%, Kapur's and Yen's in a 3% and the rest with less than 3% have no significative statistical relevance.

From these results we can conclude that fuzzy thresholding methods produced very uniform binary images in our experiments but, as pictures 2 and 3 show, objects placed on a shady background are not correctly thresholded. In fact, the object cannot be detected in the thresholded image at all, and this discourages the use of fuzzy thresholding algorithms and the Triangle method when there is shady and patterned backgrounds.

Otsu's method did not achieve a significant ranking according to uniformity measures. Examining the result images, we can say that this method does not have the same problems as entropy methods when there is a patterned background, but it is still susceptible to irregular brightness. Results show that regions extracted from images with irregular or direct brightness are less uniform "intra-region" compared to the same images thresholded using Kittler-Illingworth's algorithm. This can be explained taking into account the optimal inter-class variance achieved by Otsu, which makes the thresholding process more reliable to undesired noise and brightness. If we examine the histograms of the involved images, we can notice that histograms with two peaks are more uniformly thresholded by Kittler-Illingworth's method, and this observation can be supported by the fact that the algorithm supposes that an image can be represented as the mixture of two Gaussian distributions.

Inspecting the results for local thresholding methods, we can see that Bernsen's method provided the most uniform thresholded image in a significant number of cases. This can be explained from the fact that the algorithm uses the local mid-grey value of a given window when the pixel's local contrast is greater than a given contrast threshold, that in the experiment it was set to 15. This tends to produce homogeneous classes except in image 10, where Bernsen's method produced the least uniform image.

My proposed method was invoked with default parameters (window size of 25x25 pixels and $k_g = 0.6$ and $k_l = 0.4$ parameters, equation 3.16) and produced uniform results compared with other local methods with an uniformity average of 36.641, but results are worse in comparison with global methods. In particular, the method achieved the most non-homogeneous regions in image 4. Examining the results, image 4 is a complicated one for region segmentation because background has a high contrast compared to foreground (object). Figure 4.4 represents graphically the uniformity performance of the proposed method against the set of global methods tested. Examining the figure, some significant peaks can be observed in image 4 for the proposed algorithm and Doyle's method and in image 7 for Ridler's and Triangle's methods.

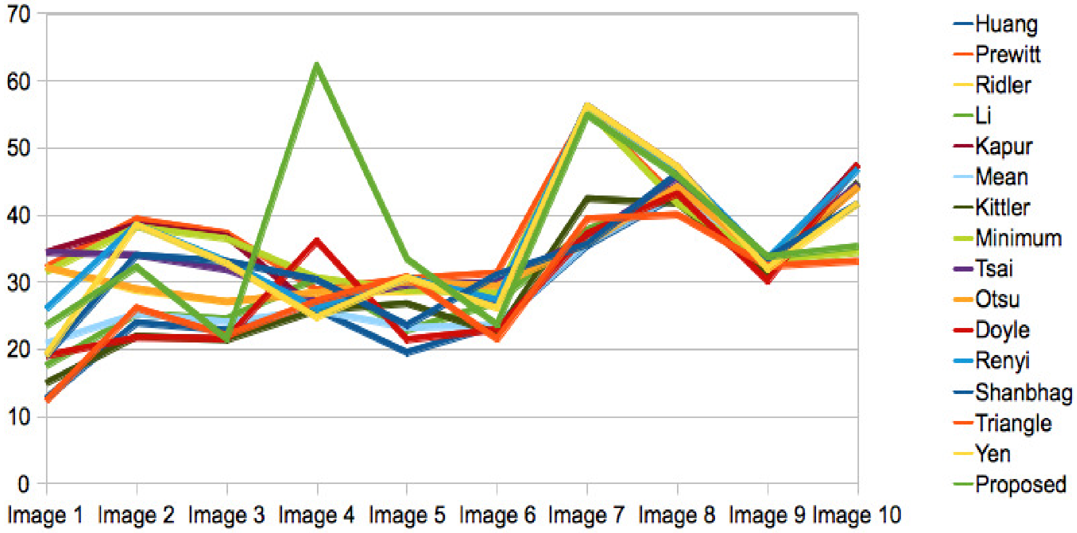


Figure 4.4: *Comparison of the region uniformity of global methods compared with the proposed algorithm (in green).*

There is a list of methods examined in this experiment that provided non-homogeneous regions for the majority of tested images. As it happened with Otsu, it does not necessarily indicate a bad thresholding, according to a visual inspection of the results. To avoid deriving wrong conclusions, region uniformity cannot be the only performance metric applied to thresholding algorithms.

4.4 Misclassification Error

The second experiment consisted of the evaluation of binary images according to the similarities with a human-provided set of correctly thresholded, or ground-truth, images. More precisely, pixels in the resulting binary image can be classified as pixels correctly assigned to the foreground, pixels correctly assigned to the background, pixels incorrectly assigned to the foreground, and pixels incorrectly assigned to the background. From this definitions we can define a metric called the *misclassification error* (ME) of a given image, that is, the percentage of background pixels wrongly assigned to foreground; and conversely, foreground pixels wrongly assigned to background. For the two-class segmentation problem of image thresholding, we can express ME as:

$$ME = 1 - (|B_O \cap B_T| + |F_O \cap F_T|)/(|B_O| + |F_O|) \tag{4.5}$$

where B_O and F_O denote, respectively, the background and foreground of the original ground-truth image and B_T and F_T denote the background and foreground pixels in the test image. The ME varies from 0 for a perfectly thresholded image, to 1 for a totally wrongly thresholded image.

Ground-truth images were produced using Adobe Photoshop by painting in black those pixels that were expected to be considered as background by the thresholding algorithm, and painting in white those that were expected to be considered foreground. Inside forms and typography were reproduced as accurately as possible in order to test both the precision of thresholding techniques globally (i.e. marker is correctly segmented from the background), and inside the marker (i.e. marker's inside drawings and text are preserved).

In the first test, the whole ground-truth image was evaluated against the manually-produced results, so as to test the accuracy in object/background segmentation. Tables 4.5, 4.6, 4.7 sum up the misclassification error results for each analyzed thresholding algorithm.

From the evaluation results, Minimum's method got the overall best accuracy with an average misclassification error of 0.090. This is a very low error rate, and it can be confirmed

	Huang	**Prewitt**	**Ridler**	**Li**	**Kapur**	**Mean**	**Kittler**	**Minimum**
Image 1	0.719	0.002	0.002	0.493	0.052	0.420	0.003	0.003
Image 2	0.412	0.022	0.252	0.361	0.029	0.361	0.002	0.002
Image 3	0.375	0.026	0.248	0.337	0.029	0.344	0.004	0.004
Image 4	0.713	0.716	0.723	0.742	0.680	0.647	0.737	0.737
Image 5	0.665	0.022	0.295	0.407	0.107	0.429	0.000	0.000
Image 6	0.272	0.078	0.150	0.181	0.181	0.272	0.002	0.002
Image 7	0.563	0.002	0.697	0.768	0.010	0.525	0.002	0.002
Image 8	0.493	0.507	0.345	0.498	0.271	0.401	0.554	0.554
Image 9	0.287	0.247	0.182	0.245	0.232	0.259	0.185	0.185
Image 10	0.205	0.208	0.208	0.242	0.356	0.281	0.102	0.102

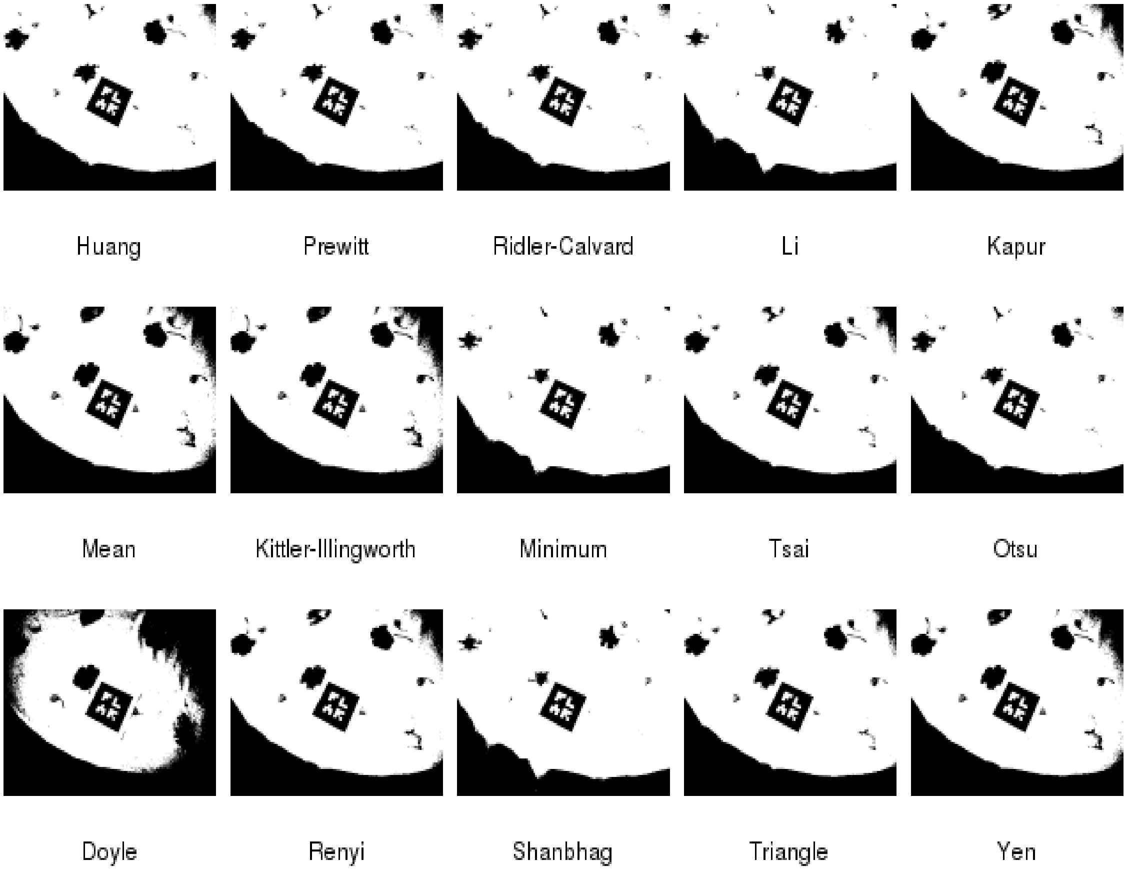
Table 4.5: *Misclassification error (ME) for a subset of test images.*

	Tsai	**Otsu**	**Doyle**	**Renyi**	**Shanbhag**	**Triangle**	**Yen**
Image 1	0.031	0.002	0.239	0.252	0.475	0.758	0.457
Image 2	0.127	0.235	0.029	0.012	0.127	0.354	0.029
Image 3	0.133	0.248	0.101	0.105	0.101	0.400	0.104
Image 4	0.695	0.723	0.675	0.687	0.743	0.695	0.660
Image 5	0.192	0.113	0.107	0.123	0.452	0.082	0.107
Image 6	0.124	0.150	0.178	0.187	0.092	0.398	0.190
Image 7	0.472	0.689	0.010	0.021	0.653	0.845	0.010
Image 8	0.304	0.340	0.273	0.232	0.288	0.793	0.275
Image 9	0.138	0.182	0.249	0.212	0.178	0.277	0.297
Image 10	0.228	0.208	0.356	0.363	0.585	0.050	0.585

Table 4.6: *Misclassification error (ME) for a subset of test images.*

	Bernsen	**Mean**	**Median**	**MidGrey**	**Niblack**	**Sauvola**	**Proposed**
Image 1	0.246	0.423	0.453	0.273	0.325	0.910	**0.431**
Image 2	0.304	0.401	0.423	0.333	0.353	0.944	**0.301**
Image 3	0.285	0.401	0.454	0.331	0.349	0.950	**0.430**
Image 4	0.379	0.391	0.398	0.494	0.393	0.946	**0.321**
Image 5	0.497	0.449	0.443	0.455	0.412	0.968	**0.444**
Image 6	0.220	0.386	0.387	0.371	0.358	0.897	**0.345**
Image 7	0.552	0.442	0.445	0.501	0.386	0.958	**0.512**
Image 8	0.320	0.409	0.423	0.322	0.350	0.925	**0.654**
Image 9	0.290	0.421	0.443	0.314	0.313	0.929	**0.478**
Image 10	0.381	0.423	0.423	0.369	0.324	0.925	**0.565**

Table 4.7: *Misclassification error (ME) for a subset of test images.*



Huang Prewitt Ridler-Calvard Li Kapur

Mean Kittler-Illingworth Minimum Tsai Otsu

Doyle Renyi Shanbhag Triangle Yen

Figure 4.5: *Image 5 results using global thresholding methods.*

by examining the good results this method provided, where the only images where it could not correctly threshold the marker were images 8, 9 and 10. Figure 4.6 shows Minimum's method results.

Local methods did not get specially good results according to misclassification metrics, mostly because considering the entire image is a handicap for these methods because by applying multiple thresholding values, it is unlikely that a background with a highly variable gray level values gets labeled as background when the background area is high.

From the results of the experiment, we can determine that, overall, misclassification error is not particularly high for global methods except for some subset of images (for example, images 4.7 and 4.8 from Tables 4.8 and 4.9). This is significant because that means that

	Huang	**Prewitt**	**Ridler**	**Li**	**Kapur**	**Mean**	**Kittler**	**Minimum**
Image 1	0.691	0.014	0.013	0.623	0.029	0.561	0.703	0.027
Image 2	0.653	0.012	0.407	0.612	0.014	0.612	0.712	0.024
Image 3	0.472	0.033	0.277	0.447	0.029	0.452	0.526	0.074
Image 4	0.030	0.031	0.032	0.036	0.028	0.028	0.028	0.035
Image 5	0.460	0.005	0.253	0.319	0.076	0.331	0.818	0.005
Image 6	0.264	0.108	0.188	0.215	0.215	0.264	0.488	0.011
Image 7	0.472	0.001	0.664	0.776	0.005	0.421	0.892	0.001
Image 8	0.732	0.745	0.517	0.737	0.417	0.601	0.697	0.754
Image 9	0.118	0.094	0.070	0.092	0.087	0.100	0.520	0.071
Image 10	0.100	0.103	0.103	0.132	0.192	0.154	0.154	0.004

Table 4.8: *Misclassification error (ME) around the marker for a subset of test images.*

	Tsai	**Otsu**	**Doyle**	**Renyi**	**Shanbhag**	**Triangle**	**Yen**
Image 1	0.023	0.013	0.158	0.138	0.612	0.700	0.600
Image 2	0.126	0.378	0.014	0.056	0.126	0.601	0.014
Image 3	0.066	0.277	0.041	0.067	0.041	0.483	0.043
Image 4	0.029	0.032	0.028	0.023	0.036	0.029	0.028
Image 5	0.173	0.086	0.076	0.086	0.342	0.026	0.076
Image 6	0.161	0.188	0.213	0.243	0.126	0.422	0.218
Image 7	0.351	0.651	0.005	0.003	0.595	0.840	0.005
Image 8	0.453	0.511	0.419	0.423	0.435	0.754	0.421
Image 9	0.051	0.070	0.095	0.089	0.069	0.110	0.125
Image 10	0.120	0.103	0.196	0.187	0.354	0.015	0.354

Table 4.9: *Misclassification error (ME) around the marker for a subset of test images.*

	Bernsen	**Mean**	**Median**	**MidGrey**	**Niblack**	**Sauvola**	**Proposed**
Image 1	0.079	0.394	0.334	0.082	0.082	0.290	**0.076**
Image 2	0.179	0.335	0.377	0.167	0.187	0.628	**0.328**
Image 3	0.055	0.237	0.213	0.051	0.060	0.342	**0.203**
Image 4	0.563	0.499	0.478	0.363	0.438	0.132	**0.306**
Image 5	0.299	0.363	0.334	0.284	0.272	0.712	**0.321**
Image 6	0.168	0.370	0.398	0.226	0.238	0.624	**0.323**
Image 7	0.535	0.461	0.434	0.486	0.389	0.969	**0.705**
Image 8	0.422	0.477	0.486	0.406	0.439	0.752	**0.441**
Image 9	0.223	0.387	0.324	0.282	0.283	0.886	**0.291**
Image 10	0.197	0.416	0.485	0.131	0.143	0.844	**0.038**

Table 4.10: *Misclassification error (ME) around the marker for a subset of test images.*

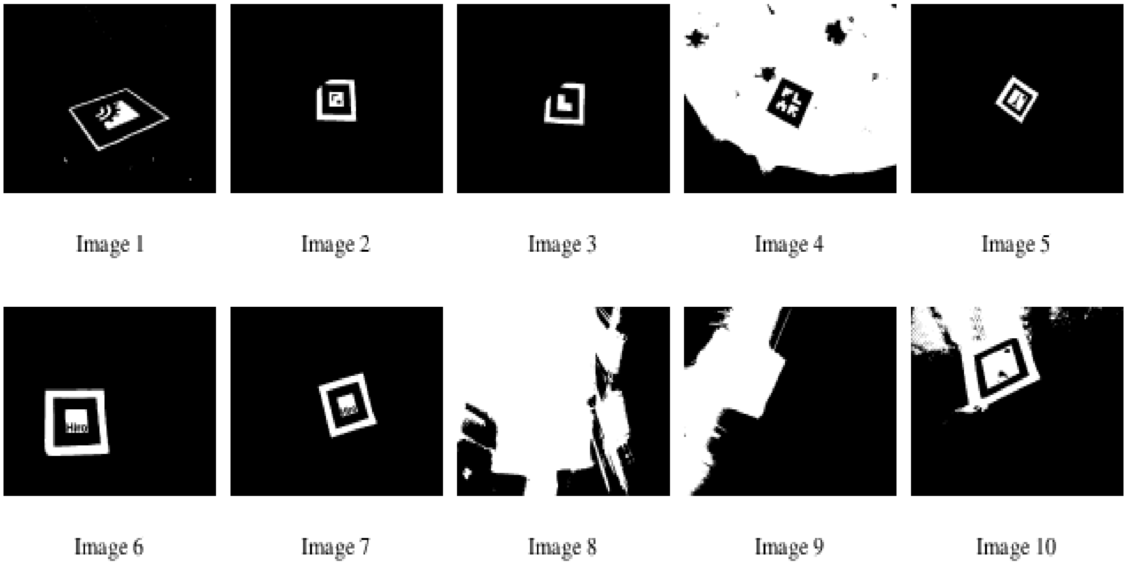


Image 1 Image 2 Image 3 Image 4 Image 5

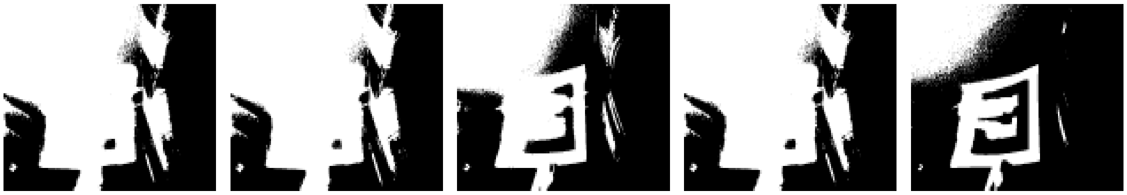
Image 6 Image 7 Image 8 Image 9 Image 10

Figure 4.6: *Minimum's method results.*

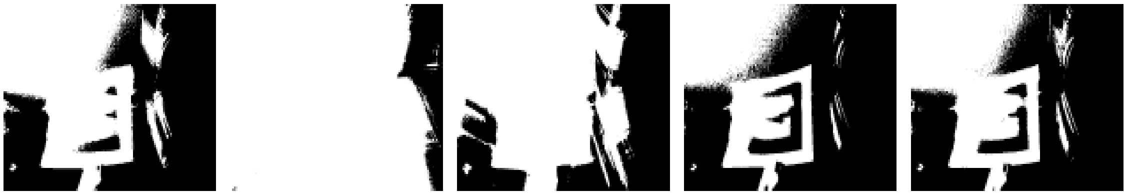
subset of images is going to be wrongly thresholded with a very high probability using a global method. In fact, those images were tested with the latest version of ARToolkit and the system was unable to successfully detect the marker, as expected. Some thresholding methods give errors around 0.2, but a closer visual inspection reveals that the thresholding is totally wrong; some high quality results according to misclassification error metric are actually wrong segmentations. Conclusions cannot be extracted from misclassification error alone, as happened with region uniformity.

A closer inspection of the histogram morphology and properties of those images reveals that spread histograms like the one of image 10 are hard for global methods. More formally, image 10 has the highest variance of the test set.

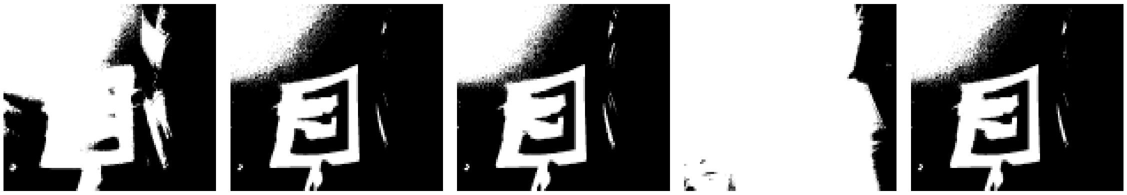
For the rest of images analyzed, it can be said that no particular global method was a clear winner, with many of them sharing the best value for many images. Otsu's method, one of the most common thresholding algorithms in augmented reality applications and available by default in the ARToolkit library, had problems when the marker was placed on a patterned environment and this supposed that method, on average, was ranked on a high



Huang Prewitt Ridler-Calvard Li Kapur



Mean Kittler-Illingworth Minimum Tsai Otsu

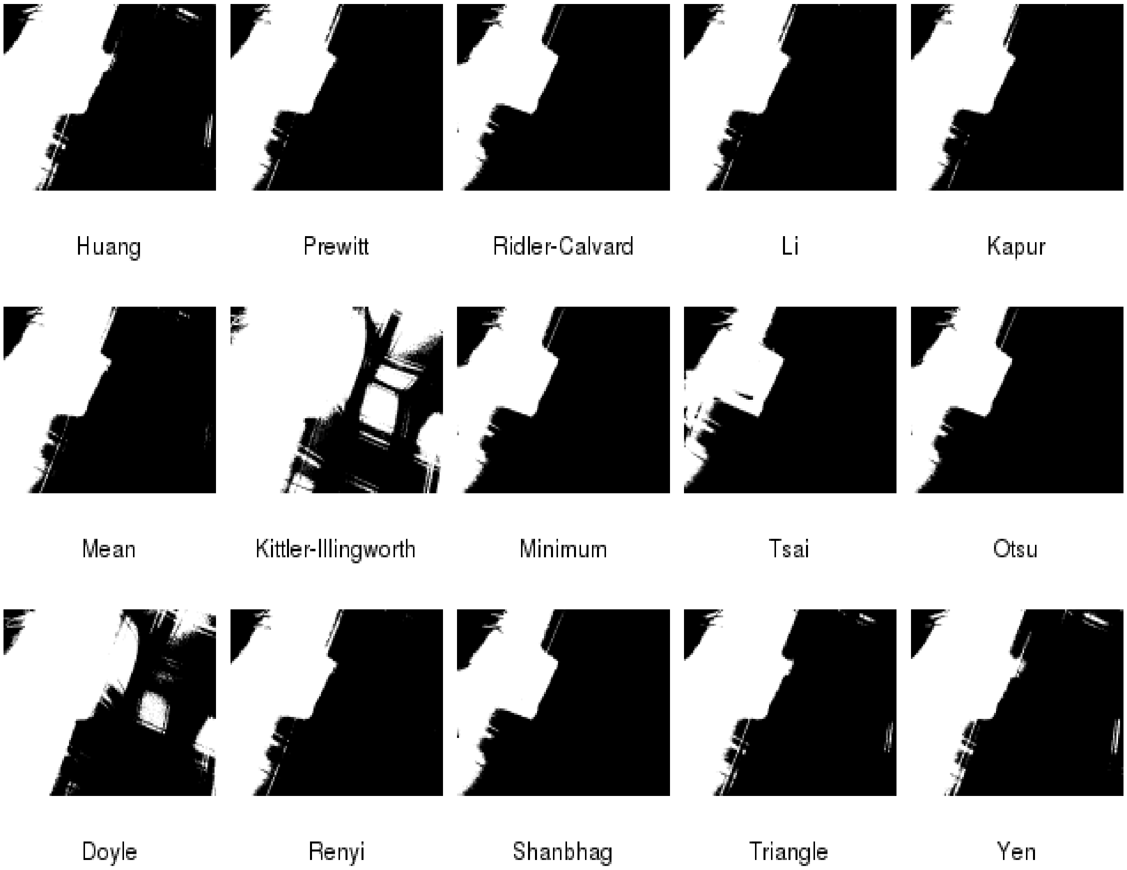


Doyle Renyi Shanbhag Triangle Yen

Figure 4.7: *Image 9 results using global thresholding methods.*

position with 0.012, but not close to the best ranked method on average, Tsai's method, with a misclassification error of 0.006.

Otsu was not the only method with problems when the marker was placed on a patterned background. A visual inspection of some of the results (for example, image 4.10, shows that Huang, Li, Mean, Kittler-Illingworth, Doyle, and Triangle achieved high misclassification error values for that image, that is, the object could not be correctly segmented from the background. In that kind of images, entropy methods and Shanbhag's method obtained a good result. Shanbhag's use of information measure did not give it a clear advantage in the results compared to entropy methods, except over Huang's fuzzy thresholding method using Shannon's entropy function. Minimum method produced a top quality result after



Huang Prewitt Ridler-Calvard Li Kapur

Mean Kittler-Illingworth Minimum Tsai Otsu

Doyle Renyi Shanbhag Triangle Yen

Figure 4.8: *Image 10 results using global thresholding methods.*

visual inspection. The use of bimodal histograms in this method helped this good results, because the image histogram did not have extremely unequal peaks or flat valleys, as figure 4.9 proves.

An important observation that can be made from the results is that the method with the highest misclassification error on average was the iterative version of Kittler-Illingworth's. In our conclusions, the use of this thresholding algorithm is discouraged in augmented reality applications. The only images where Kittler Illingworth's method highlighted were the ones similar to image 4.5, with a clear gray-level differentiation between object and background, as shown by the prominent peak in their histograms (see figure 4.11).

As it was noted before, the global method with the lowest misclassification error on

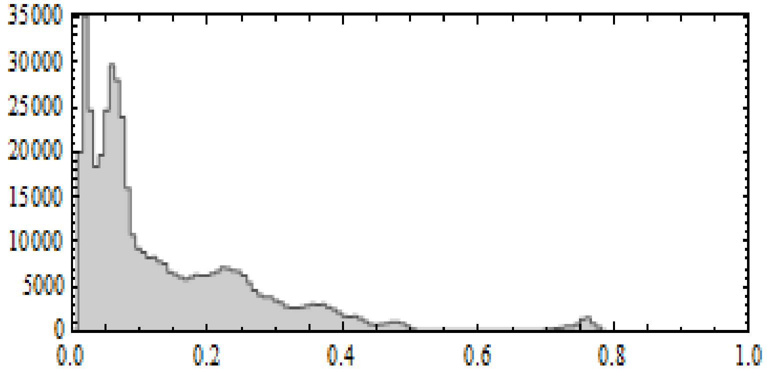


Figure 4.9: *Image 4 histogram.*

average was Tsai's method. This approach to thresholding selects a thresholding value so that the moments of the input picture are preserved in the output picture. The method performed well in two out of the three categories of pictures: dark images and markers over a patterned background. Performance under uneven lighting conditions was a problem for Tsai's method as the higher misclassification error rate indicates.

Triangle's method was able to successfully threshold image 10, where the rest of global algorithms failed to correctly segment the marker's inside drawing. Examining its histogram properties, as table 4.1 shows, image 10 has the largest variance of the evaluation set, with a great concentration of pixels around the white gray level. Triangle's method is appropriate for images where the maximum is very near the histogram extremes because of its design: the algorithm assumes a peak near one of the histogram extremes and searches the threshold value toward the other end.

Local methods demonstrated an overall improvement over global ones with respect to misclassification error. No method averaged a value greater than 0.7, and even the best global method (Tsai's) performed worse than the worst local method (Median). Among local methods, Niblack's is ranked best in our experiment, with image 2 being the best result overall. The distinctive characteristic of this image is its dark background, where black background pixels can be interspersed with black, foreground, marker pixels. The histogram shows a predominant dark gray levels. The problem with local methods is their

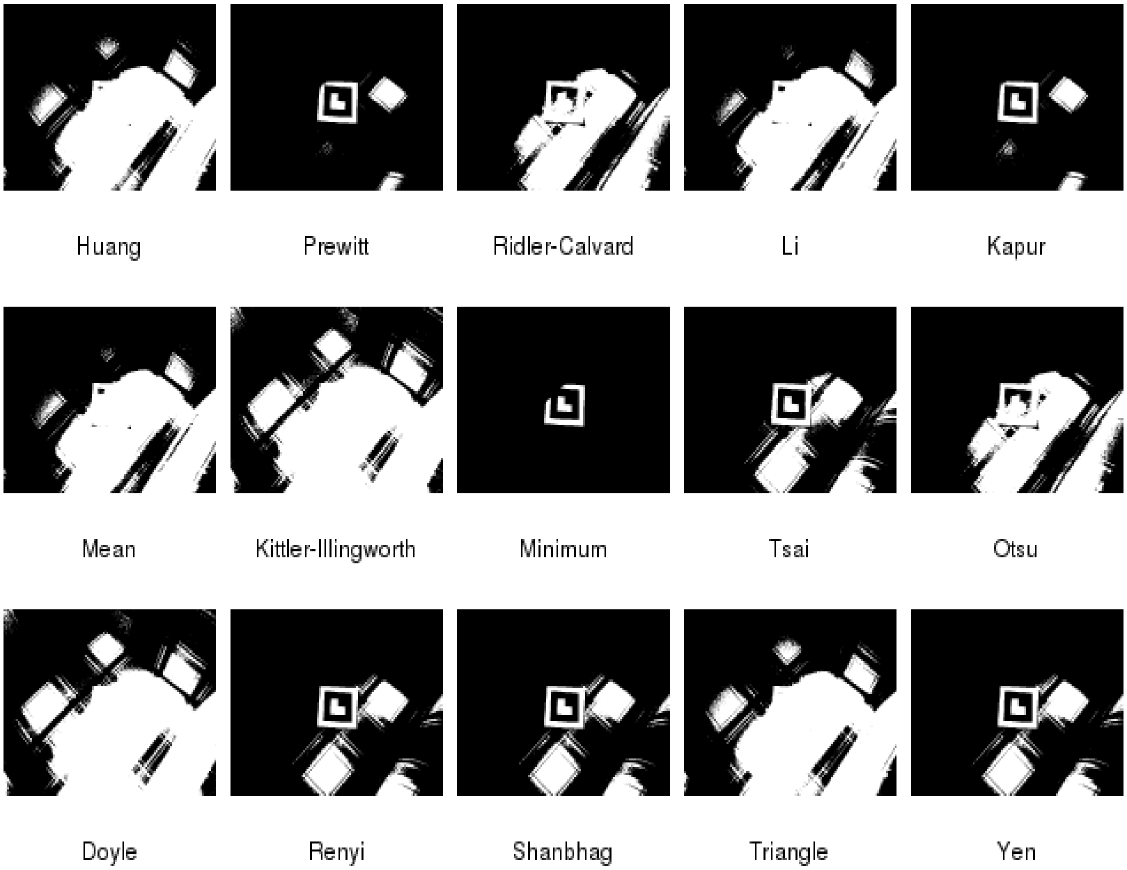


Figure 4.10: *Image 4 results using global thresholding methods.*

increased computational cost, a critical aspect in applications with strict delay requirements, like mobile augmented reality. In particular, local methods averaged a running time two orders of magnitude greater than global ones, with the Median's method being the slowest of all, with an average of 2 seconds for a 800x600 image in our implementation. Moreover, this local method performed worst according to misclassification error metric, so Median's method showed clear disadvantages if a local thresholding method is desired to solve the thresholding problem in augmented reality applications. Mean's method was the fastest local method on average in our results, with the drawback that it was also the second worst performance in the local thresholding algorithms category. Running time of Mean's method was on average nearly two times better than the next methods in running time performance:

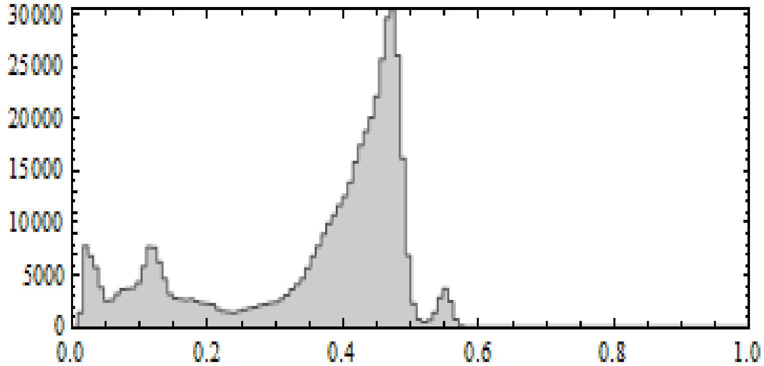
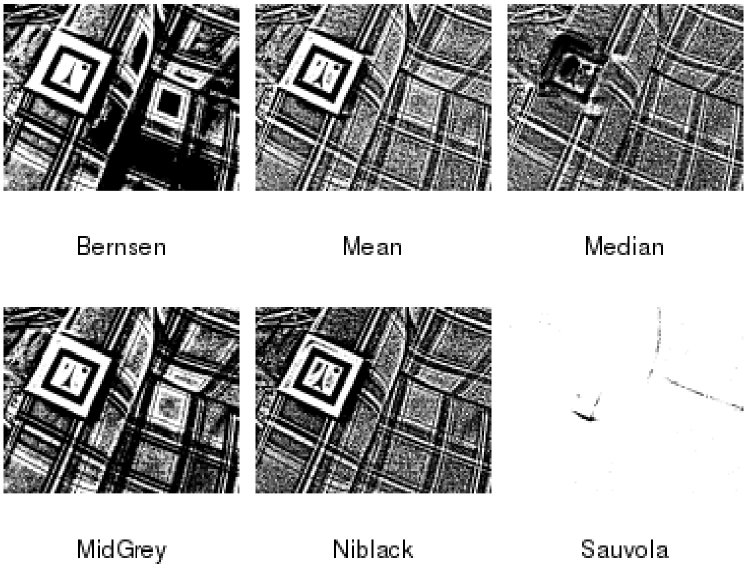


Figure 4.11: *Image 5 histogram.*


Bernsen Mean Median

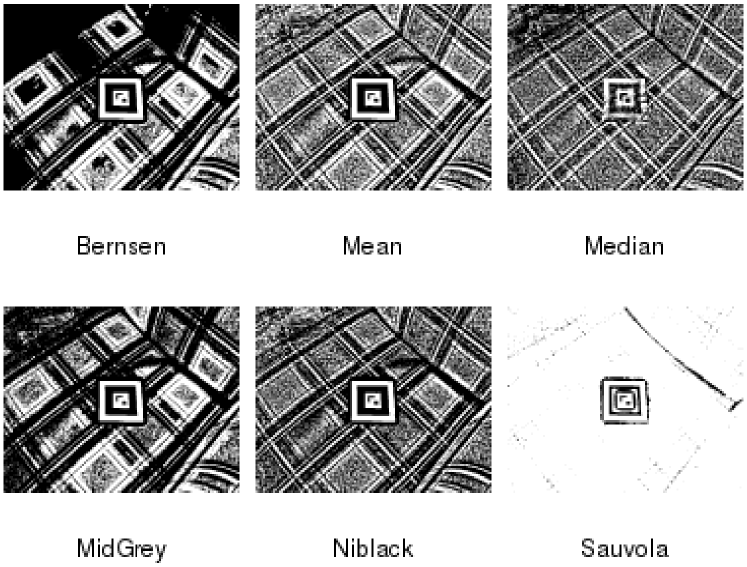
MidGrey Niblack Sauvola

Figure 4.12: *Image 10 results using local (dynamic) thresholding methods.*

Niblack and Sauvola. Niblack's performance was particularly good in running time (second best), and in misclassification error (best), so it makes it a very interesting local method when thresholding augmented reality marker images.

Once available thresholding methods had been evaluated according to different metrics (region uniformity, misclassification error and running time), the proposed method in this work was compared against available methods.

Indeed, the proposed method was evaluated against the rest of methods where table 4.10



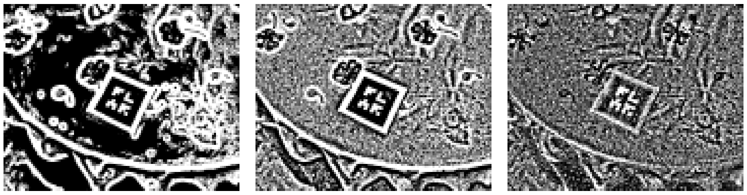
Bernsen Mean Median

MidGrey Niblack Sauvola

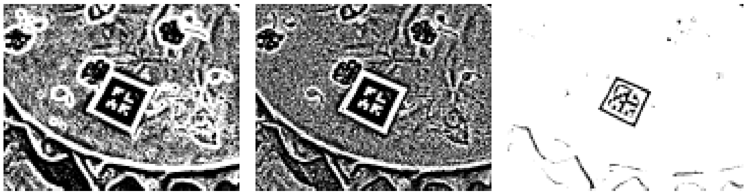
Figure 4.13: *Image 3 results using local (dynamic) thresholding methods.*

sums up the results for a standard window of size 25 and $k_g = 0.6$, $k_l = 0.4$ parameters defined in the heuristic. It averaged a misclassification error of 0.303, placing it over Mean method, Median, and Sauvola's, in the group of dynamic methods. Compared against global methods, our proposed algorithm offered lower misclassification error than Triangle's, Huang's, and Kittler-Illingworth's methods. Examining particular results, as image 4.18 depicts, the proposed method got good results with image 10, only bettered by Bernsen's method. Niblack's algorithm, which also computes the sum of gray levels and sum of squares inside every window, performed similarly to our method (slightly better), except in images 2 and 5.

The other method that resembles our proposed method in the fact that it also uses moment preserving techniques is Tsai's. Our method performed better than Tsai's in images 10 and 8, where Tsai's was unable to correctly threshold the marker, as a visual inspection reveals. The results of the application of moment-preserving thresholding to local regions shows that it improves correct classification rates with respect to moment-preserving techniques applied to the global image, with the drawback that it worsens the classification rate



Bernsen Mean Median



MidGrey Niblack Sauvola

Figure 4.14: *Image 5 results using local (dynamic) thresholding methods.*

in certain kind of images, like image 5.

Figure 4.18 shows the results of our proposed method with default parameters of window size and heuristics. In order to test the influence of window sizes and heuristic values in the final result, the algorithm was repeatedly invoked using different window sizes and weight heuristic values. The first evaluation consisted of a small window size (5x5 pixels) and a 0.8 weight for the moment-preserving threshold and 0.2 for the sum of squares threshold. This window size is not particularly appropriate considering that our algorithm evenly divides the picture in window regions and a small window results in many windows and, because of the increased processing time for each window, a higher running time. Visual results are shown in figure 4.15.

A second evaluation was performed with a window size of 15x15 pixels and the same weights as the previous experiment (0.8/0.2). The results express a lower misclassification error than with the window size of 5x5 pixels, given the higher amount of information available in each window. Figure 4.16 shows the results of this thresholding technique.

Expanding the window size to 25x25 pixels resulted in lower misclassification error rates,

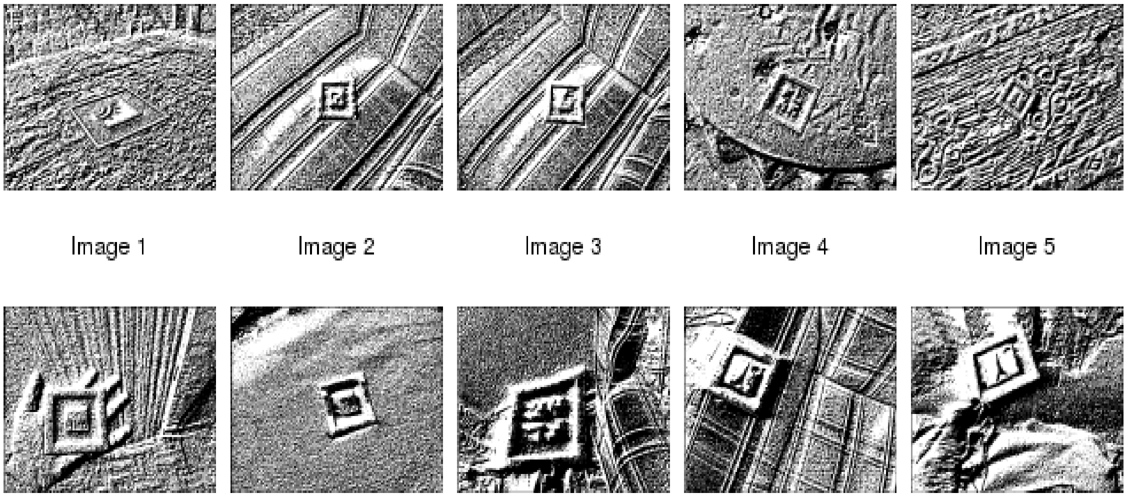


Image 1 Image 2 Image 3 Image 4 Image 5

Image 6 Image 7 Image 8 Image 9 Image 10

Figure 4.15: *Results of our proposed method, with a window size of 5 pixels.*

	5x5	**15x15**	**25x25**	**25x25(0.6/0.4)**	**55x55**
Image 1	0.673	0.654	0.567	0.556	0.432
Image 2	0.723	0.678	0.654	0.578	0.511
Image 3	0.752	0.764	0.678	0.787	0.640
Image 4	0.765	0.543	0.456	0.452	0.340
Image 5	0.566	0.654	0.678	0.567	0.610
Image 6	0.877	0.675	0.567	0.432	0.420
Image 7	0.767	0.545	0.523	0.430	0.397
Image 8	0.867	0.786	0.678	0.620	0.387
Image 9	0.687	0.673	0.567	0.653	0.564
Image 10	0.610	0.567	0.456	0.560	0.498

Table 4.11: *Misclassification error (ME) around the marker for our proposed method with different window sizes and parameters.*

but the output image presented medium-sized blocks of white or black pixels that increased the noisiness of the result, as figure 4.17 reveals.

Maintaining the 25x25 pixels window, weight values were varied in another experiment. A 0.6 weight for the moment-preserving threshold and 0.4 for the sum of squares was tested and results show that misclassification metrics remained within a similar range, with a slight improvement in images 4 and 6. Tsai's method, also based on moment-preserving

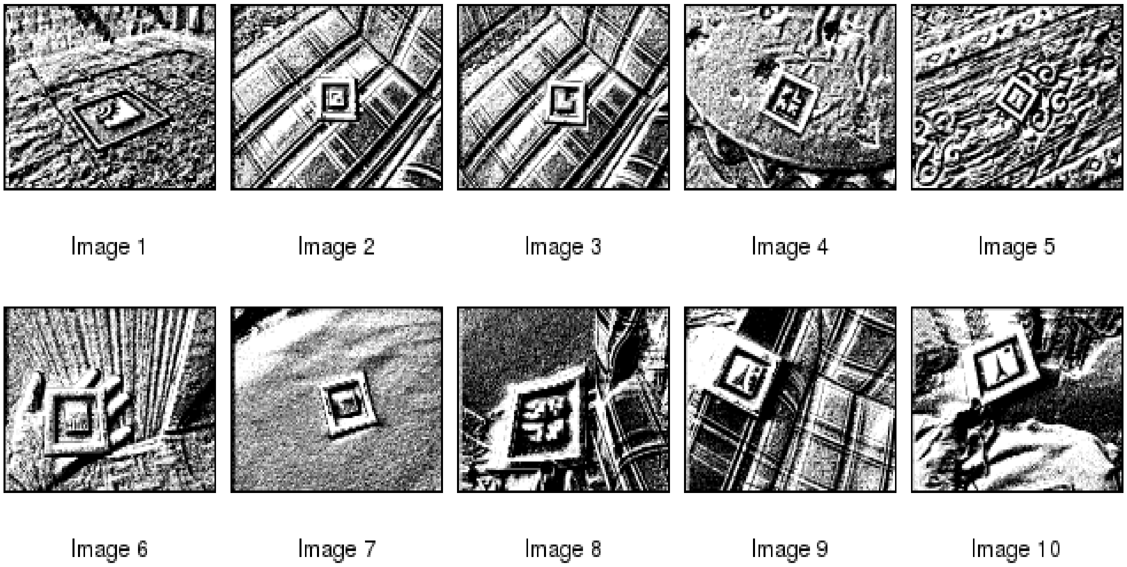


Image 1 Image 2 Image 3 Image 4 Image 5

Image 6 Image 7 Image 8 Image 9 Image 10

Figure 4.16: *Results of our proposed method, with a window size of 15 pixels.*

techniques also produced slightly worse results for those images. In figure 4.18 the results of the thresholding are shown.

Finally, a window of size 55x55 was used. It delivered fairly high quality results with respect to misclassification error, but inside forms were not accurately thresholded. Tests using the ARToolkit library showed that marker detection was irregular. Bigger windows only worsened this problem, so the use of windows bigger than 55x55 pixels does not provide any benefit; moreover, thresholded images start to lose their definition and fidelity.

4.5 Computational Cost

This metric was evaluated because it is an important measure for real-time applications like augmented reality. For the experiment, a C implementation of the evaluated algorithms and the proposed method was developed and run in an Intel® Core 2 duo™ laptop computer. Figures 4.19 and 4.20 represent graphically the computational cost of the evaluated algorithms against my proposed algorithm with different window sizes (5x5, 15x15, 25x25, and 55x55).

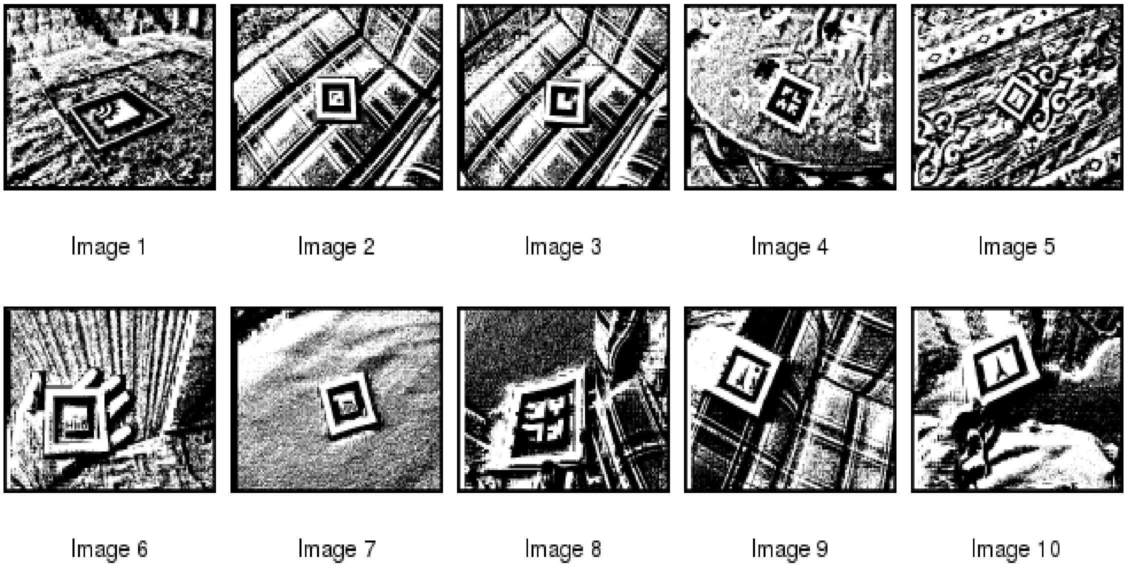


Image 1
Image 2
Image 3
Image 4
Image 5
Image 6
Image 7
Image 8
Image 9
Image 10

Figure 4.17: *Results of our proposed method, with a window size of 25 pixels.*

From the results we can conclude that global thresholding methods have a similar computational cost regardless of the image (around 4-10 ms), except the one based on Rényi entropy, whose running time performance averaged 11.7 ms. Kapur's method, based on maximizing entropy, averaged 7 ms and, as the other performance metrics showed, produced results with a similar quality as Rényi's, so Kapur's method may be preferable to Rényi at least in applications with strict delay requirements.

The results show that bigger windows effectively reduce running time from 1400 milliseconds in the 5x5 pixels window to around 700 milliseconds in the 55x55 one. Even bigger windows do not lower the running time substantially and, in fact, it is higher when windows are very big (from 61x61 pixels onwards) due to the time complexity of calculations inside each window.

Figure 4.21 represents graphically the running time values when the algorithm is invoked with a particular window size. Around 57x57 pixels, running time starts to increase linearly with the size of the window. Below this number, running times decrease almost logarithmically.

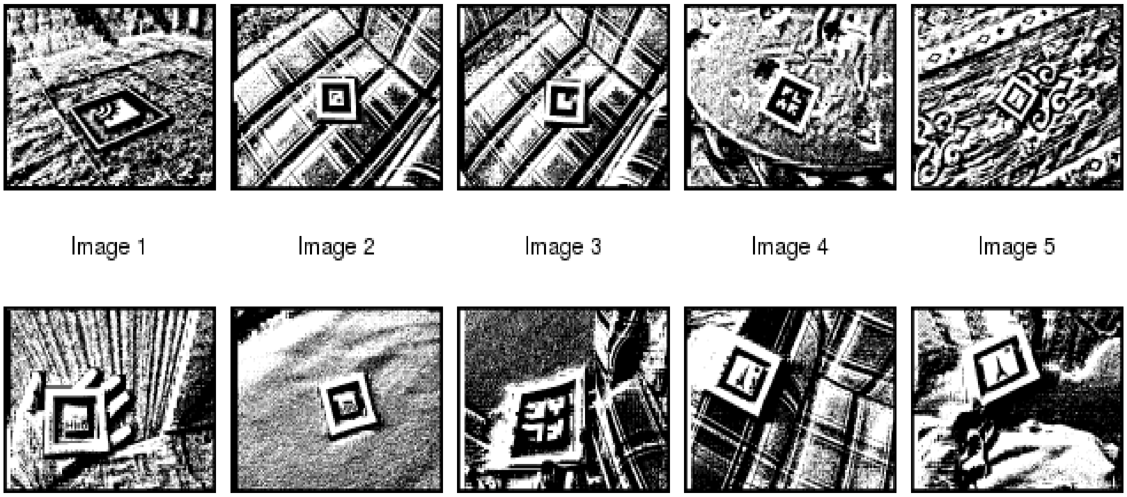


Image 1 Image 2 Image 3 Image 4 Image 5

Image 6 Image 7 Image 8 Image 9 Image 10

Figure 4.18: *Results of our proposed method, with a window size of 25 pixels and heuristics 0.6/0.4.*

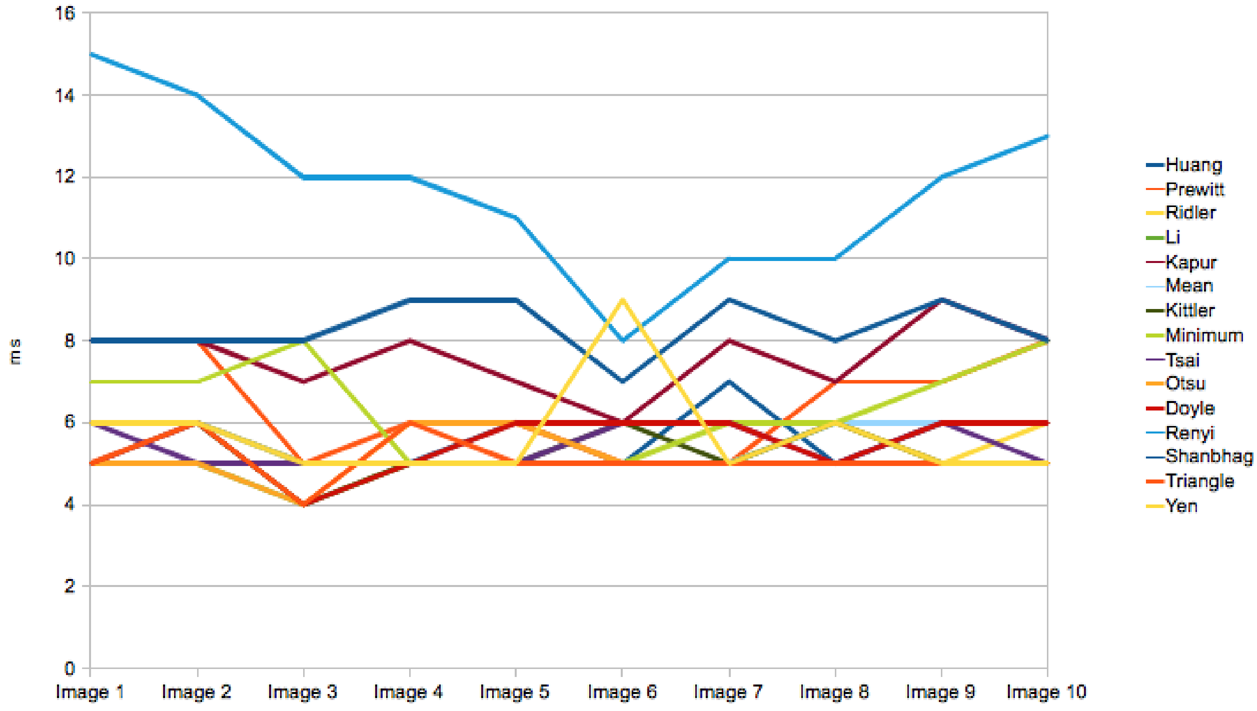


Figure 4.19: *Running time comparison between global thresholding methods tested in the experiment.*

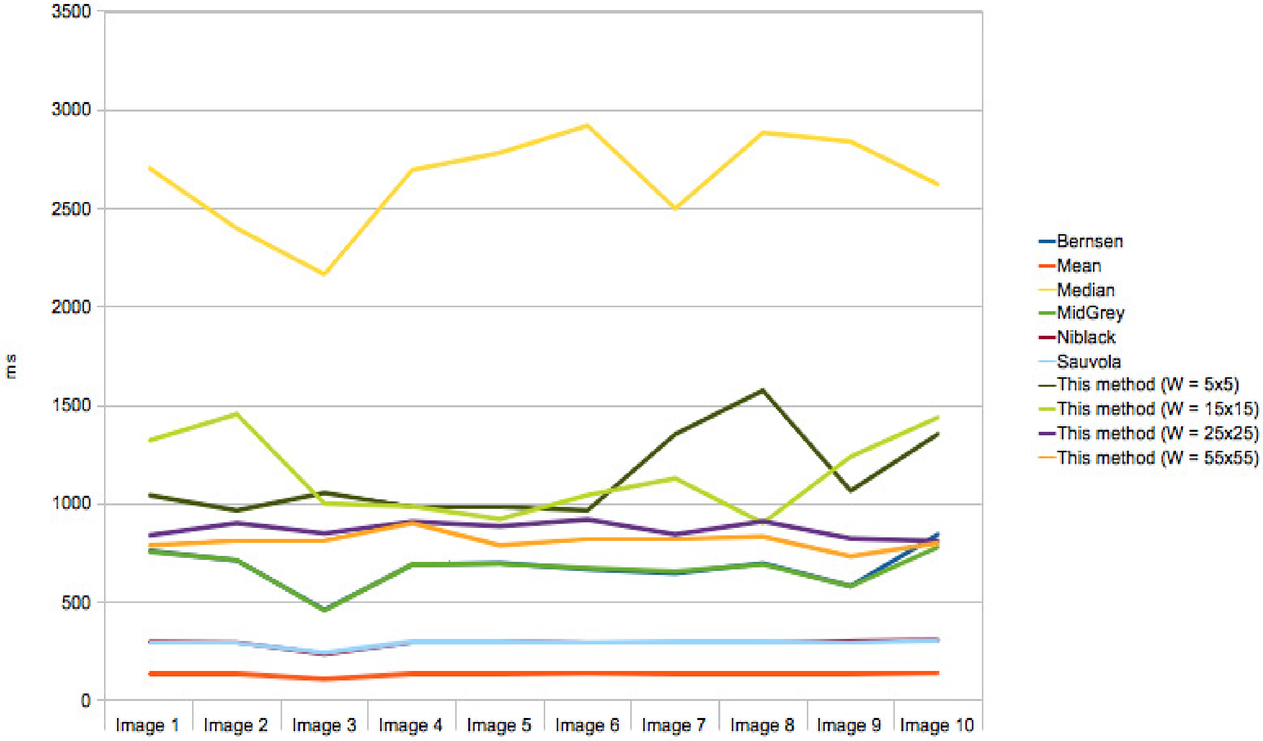


Figure 4.20: *Running time comparison between dynamic thresholding methods and the proposed algorithm with multiple window sizes (W).*

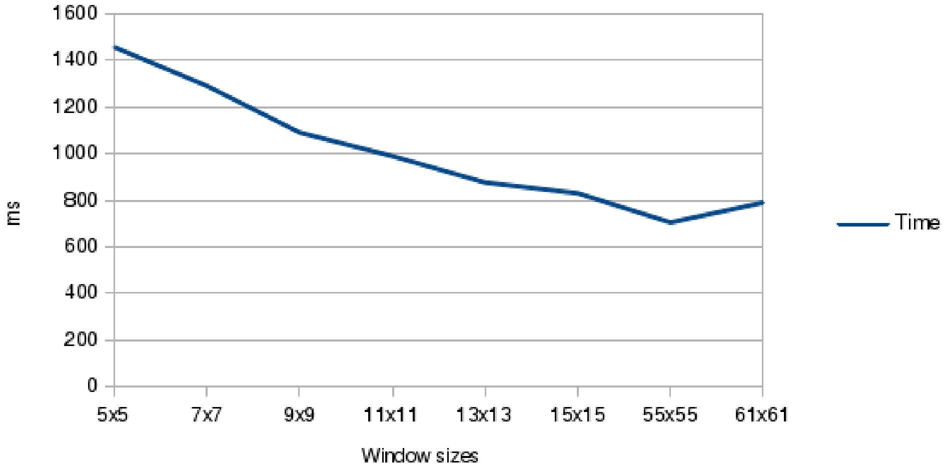


Figure 4.21: *Running time of the proposed algorithm with multiple window sizes (W).*

Chapter 5

Conclusions and Future Work

Evaluation results showed that in image thresholding for augmented reality applications there is not a single method that outperforms the rest in every case. Certain kinds of histograms, however, help decide an appropriate thresholding method that offers good results for that type of images. For example, experimental results from this work have demonstrated that Triangle's method is good at thresholding images whose histogram has a maximum near one of the extremes. There have been many studies of image thresholding in limited environments with good results, but real-world conditions, particular to augmented reality applications, seems to demand an approximation using several techniques. When combining moment-preserving techniques with sum of squares applied to subsets of the input image, results obtained with the proposed approach demonstrate better results than global methods using moment-preserving in images where the latter methods perform poorly, for example, in images with irregular brightness. Running time performance is low compared to global thresholding methods, but it is average in the group of dynamic methods, what makes it an interesting method to try, even in real-time applications.

Future work should revolve around two aspects: better running time performance and more homogeneous results. About running time performance, moment-preserving techniques are computationally expensive with histogram extraction and normalization taking most of the time, so possible optimizations in this step should be considered. Other improvements could be the development of a better locality technique to produce more homogeneous results. The use of squared windows as subsets with the same thresholding value tends to produce "patches" in the results that reduce region uniformity. The use of probability measures inside a particular window to assign a thresholding value to each pixel could improve region uniformity performance and, consequently, the accuracy of the augmented reality marker detection subsystem. Other possible improvement, supported by a paper by [1] could be the use of Gamma distributions instead of Gaussian distributions in the moment-preserving technique. The use of regions of interest and thresholds that vary over time, as proposed in previous research, would be an interesting addition to explore.

Bibliography

[1] A. Al-Hussain and A. El-Zaart. Moment-preserving thresholding using gamma distribution. In *Computer Engineering and Technology (ICCET), 2010 2nd International Conference on*, volume 6, pages V6–323. IEEE, 2010.

[2] R.T.W. Calvard et al. Picture thresholding using an iterative slection method. *IEEE Transactions on Systems Man and Cybernetics*, 8(Aug):630–632, 1978.

[3] J.H. Chang, K.C. Fan, and Y.L. Chang. Multi-modal gray-level histogram modeling and decomposition. *Image and Vision Computing*, 20(3):203–216, 2002.

[4] S.C. Cheng and W.H. Tsai. A neural network implementation of the moment-preserving technique and its application to thresholding. *Computers, IEEE Transactions on*, 42 (4):501–507, 1993.

[5] S. Cho, R. Haralick, and S. Yi. Improvement of kittler and illingworth's minimum error thresholding. *Pattern Recognition*, 22(5):609–617, 1989.

[6] CK Chow and T. Kaneko. Automatic boundary detection of the left ventricle from cineangiograms. *Computers and biomedical research*, 5(4):388–410, 1972.

[7] L.K. Huang and M.J.J. Wang. Image thresholding by minimizing the measures of fuzziness. *Pattern recognition*, 28(1):41–51, 1995.

[8] JN Kapur, P.K. Sahoo, and AKC Wong. A new method for gray-level picture thresholding using the entropy of the histogram. *Computer vision, graphics, and image processing*, 29(3):273–285, 1985.

[9] J. Kittler and J. Illingworth. Minimum error thresholding. *Pattern recognition*, 19(1): 41–47, 1986.

[10] CK Lee, FW Choy, and HC Lam. Real-time thresholding using histogram concavity [image processing]. In *Industrial Electronics, 1992., Proceedings of the IEEE International Symposium on*, pages 500–503. IEEE, 1992.

[11] M.D. Levine and A.M. Nazif. Dynamic measurement of computer generated image segmentations. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (2): 155–164, 1985.

[12] CH Li and PKS Tam. An iterative algorithm for minimum cross entropy thresholding. *Pattern Recognition Letters*, 19(8):771–776, 1998.

[13] L. Naimark and E. Foxlin. Circular data matrix fiducial system and robust image processing for a wearable vision-inertial self-tracker. In *Proceedings of the 1st International Symposium on Mixed and Augmented Reality*, page 27. IEEE Computer Society, 2002.

[14] Y. Nakagawa and A. Rosenfeld. Some experiments on variable thresholding. *Pattern Recognition*, 11(3):191–204, 1979.

[15] W. Niblack. *An introduction to digital image processing.* Prentice-Hall International, 1986. ISBN 9780134806747. URL `http://books.google.es/books?id=XOxRAAAAMAAJ`.

[16] W. Oh and B. Lindquist. Image thresholding by indicator kriging. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(7):590–602, 1999.

[17] N. Otsu. A threshold selection method from gray-level histograms. *Automatica*, 11: 285–296, 1975.

[18] JR Parker. Gray level thresholding in badly illuminated images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 13(8):813–819, 1991.

[19] T. Pintaric. An adaptive thresholding algorithm for the augmented reality toolkit. In *Augmented Reality Toolkit Workshop, 2003. IEEE International*, page 71. IEEE, 2003.

[20] J. Prewitt and M.L. Mendelsohn. The analysis of cell images*. *Annals of the New York Academy of Sciences*, 128(3):1035–1053, 1966.

[21] N. Ramesh, J.H. Yoo, and IK Sethi. Thresholding based on histogram approximation. In *Vision, Image and Signal Processing, IEE Proceedings-*, volume 142, pages 271–279. IET, 1995.

[22] A. Rosenfeld and de la Torre. Histogram concavity analysis as an aid in threshold selection(in image processing). *IEEE Transactions on Systems, Man, and Cybernetics*, 13:231–235, 1983.

[23] J. Sauvola and M. Pietikäinen. Adaptive document image binarization. *Pattern Recognition*, 33(2):225–236, 2000.

[24] M.I. Sezan. A peak detection algorithm and its application to histogram-based image data reduction. *Computer vision, graphics, and image processing*, 49(1):36–51, 1990.

[25] M. Sezgin and B. Sankur. Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic imaging*, 13:146, 2004.

[26] W.H. Tsai. Moment-preserving thresolding: A new approach. *Computer Vision, Graphics, and Image Processing*, 29(3):377–393, 1985.

[27] NB Venkateswarlu and RD Boyle. New segmentation techniques for document image analysis. *Image and Vision Computing*, 13(7):573–583, 1995.

[28] J.M. White and G.D. Rohrer. Image thresholding for optical character recognition and other applications requiring character image extraction. *IBM Journal of Research and Development*, 27(4):400–411, 1983.

[29] M.K. Yanni and Horne E. A new approach to dynamic thresholding. *EUSIPCO'94: 9th European Conf. Sig. Process*, pages 34–44, 1994.

[30] S.D. Yanowitz and A.M. Bruckstein. A new method for image segmentation*. *Computer Vision, Graphics, and Image Processing*, 46(1):82–95, 1989.