|  |
| --- |
| UPB |
| Software Requirements Specification |
| BAM - Java image binarization using Otsu’s algorithm |
|  |
| **Mihaita Tinta** |
| **10/24/2013** |

|  |
| --- |
| There are many binarization algorithms these days and almost all of them involve several steps or processing. The raw image is transformed in a black and white version. This step is critical because many errors can be propagated from here. In character recognition is very important to separate the background from the letters. |

Contents

[Document purpose 2](#_Toc370378415)

[Document overview 2](#_Toc370378416)

[General description of the product 2](#_Toc370378417)

[The current situation 2](#_Toc370378418)

[Purpose of the product 2](#_Toc370378419)

[Product context 3](#_Toc370378420)

[Benefits 3](#_Toc370378421)

[Functional requirements 3](#_Toc370378422)

[Actors 3](#_Toc370378423)

[User 3](#_Toc370378424)

[System boundary 4](#_Toc370378425)

[Use cases description 4](#_Toc370378426)

[Non-functional requirements 5](#_Toc370378427)

[User Interface Requirements 5](#_Toc370378428)

[Performance Requirements 5](#_Toc370378429)

[Availability & Reliability 5](#_Toc370378430)

[Security Requirements 5](#_Toc370378431)

[Implementation 5](#_Toc370378432)

[Mathematical background 5](#_Toc370378433)

# Document purpose

This document is intended to describe the capabilities of the software product “Voting-Based Image Binarization” provided to its end-users. It also specifies all the non-functional requirements that the application should implement, regarding: performance, availability, reliability and security.

# Document overview

The remainder of this document is three chapters, the first offering a general description of the software product about the initial situation, the purpose of the project, the context and the benefits of the project.

The second chapter lists the functional requirements that the software product should meet. It describes the actors, the system boundary and the use cases.

The final chapter exposes the non-functional requirements of the application: performance, safety and security issues.

# General description of the product

# **The current situation**

There are many binarization algorithms these days and almost all of them involve several steps or processing. The raw image is transformed in a black and white version. This step is critical because many errors can be propagated from here. In character recognition is very important to separate the background from the letters.

## Purpose of the product

The purpose of the project is to develop an ”Image Binarization System” (IBS). The IBS will consist from two parts:

* A ”Binarization Algorithm Module” (BAM):

This will be an executable which will receive an input continuous-tone image and will produce an output binary image.

* A “Voting Binarization Algorithm Module” (VBAM): using more BAMs a “smart-voting” technology will be used to blend the independent BAMs results into a binary image.

## Product context

A BAM will be an executable which will receive from command line two file names (input\_image and an output\_image). The BAM will return an error-code: zero for no error (results are valid and will be used for voting purposes) and nonzero in case of an error occurrence (the error code should specify the error type; in this case the result will not be considered).

The output of the BAM is a 1bpp image, output\_image and an 8bpp image, output\_image- confidence. The first image is the actual binarization and the second a gray-scale image containing the confidence for the binarization for every pixel. 0 means that the respective pixel was randomly assigned a color (black or white); 255 means that the algorithm is absolutely certain that the respective color of the pixel is correctly assigned.

Benefits

The Otsu binarization method created by Nobuyuki Otsu assumes that the image to be thresholded contains two classes of pixels: foreground (something with interest for the user) and background (no interest for the user). It calculates the optimum threshold separating those two classes so that their combined spread (intra-class variance) is minimal (this represents the error).

The main advantage is the speed because we only need to compute the histograms and arrays of 256 and the easiness of the implementation.

# Functional requirements

## **Actors**

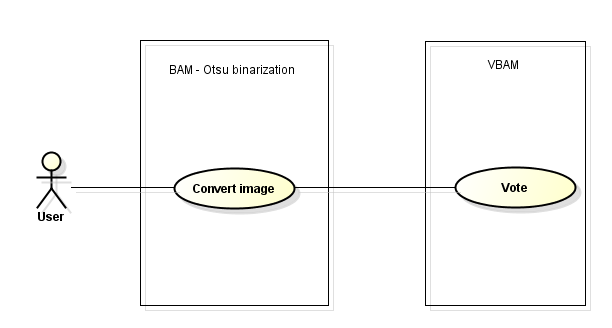
## User

A human operator or some other application is required to input an image to the BAM module. The result is then output to a destination folder. The VBAM takes the image from there and process it.

VBAM

The VBAM module gets the result and validates the images.

## System boundary



## Use cases description

|  |  |
| --- | --- |
| Use Case Name: | Convert image |
| Brief Description: | The program takes the image from an input folder. It returns a zero for no error and a nonzero when an exception occurred. |
| Priority | Essential |
| Trigger | The user operator calls the process or an actual daemon can be scheduled to check the input file periodically |
| Precondition | A valid image should be added as input |
| Basic Path | 1. Images are sent in the input folder. 2. Results are added in the output folder. |
| Post condition | The output is a 1bpp image and a 8bpp image (the confidence ) 0 means a random value and 255 is certitude |
| Exception Path | All errors are logged in the application log file and currently processing thread is stopped. |

# Non-functional requirements

# **User Interface Requirements**

The application runs as a process a displays a simple java swing interface. It also shows the current status of the processing.

# **Performance Requirements**

The system should convert the image in less than 5 seconds.

# **Availability & Reliability**

The application can be stopped manually. A service is periodically checking the input folder every 15 seconds.

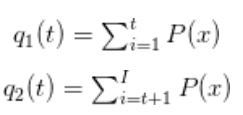
# **Security Requirements**

The original image is never deleted. It is moved in an backup archive folder.

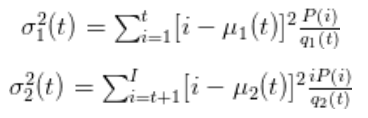
# Implementation

## Mathematical background

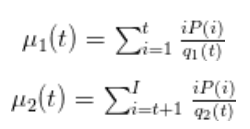
For the two classes we define the class probabilities, dependent of the P(x) histogram.



The class variance:



The classes are defined as:



The error can be minimized when maximizing the difference between the two class variance.



Therefore, the solution is ***t*** when maximizing

We have to check each potential threshold T and separate the pixels into two clusters according to the threshold, find the mean of each cluster, square the difference between the means, multiply by the number of pixels in one cluster times the number in the other.

We compute the histogram and probabilities of each intensity level, initialize the class probabilities at the beginning (Cauchy conditions), iterate over all possible threshold maximum intensity, update the class probabilities and the classes. Compute the error and remember the threshold with the minimum error ( that is the maximum mean class difference).