

A JavaScript tool to present Mathematical Morphology to beginner

CÉSAR C. NUÑEZ and AURA CONCI

Universidade Federal Fluminense (UFF), Brazil
{cnunez,aconci}@ic.uff.br

1. Introduction

This work presents an Object Oriented JavaScript program for learning basics of binary Mathematical Morphology. It employs DOM (Document Object Model) resources supported by the following browsers (or compatibles): **Internet Explorer v5.0+**, **Netscape v6.0+**, **Opera v7.0+**, **Firefox v1.0+** or **Mozilla v1.7+**. It is composed of only four Hyper Text Markup Language (.html) files, one file of Cascading Style Sheets (.css), three external JavaScript files (.js) and fourteen (.gif) figures. The code is open and can be included in html pages or adapted to other applications. Figure 1 shows its appearance.

The implementation can be used for educational purposes in several different ways. In the simplest one, on the internet, it allows experiences using the implemented operations and operators (such as expansion, contraction, intersection, union, subtraction, complement, reflection, dilation, erosion, opening, closing, etc). Images and structuring elements can be drawn on the screen using painting tools of various types. Composed operators such as top-hat, hit-or-miss, morphological gradient, and any other built from previously defined operators may be defined. Combination may be in cascade or in a parallel structure. Once tested in the tool, the corresponding code of the composed operators can be used in another program.

Students of JavaScript language can learn the basic structure of the program and improve it including new functions since the code is easy and comprehensible. For beginners in mathematical courses, the tool can be used to explain the common set theory operations (union, complement, intersection, subtraction, etc). In addition, it works as laboratory experiments for use in classrooms.

2. The tool

The tool is an open code and it has been created to be used on the main browsers/compatible in the market, as long as they support CSS and DOM.

There is no need for any additional plug-in or any other components. It allows the creation of simple binary images in a matrix of 30×30 points, and there are 11 built-in operations which can be combined in any quantity and order. However, all these limits can be increased in the implementation according to the user needs. The number of options has been kept small in order to maintain simplicity of use and one-frame interface.

Objects and methods The tool has two main classes of objects: `matrixDisplay` and `imageObject`. The first one defines an object type that simulates the pixels on the screen. It creates a square block matrix that works as an image in the user display. The dimension of this matrix can be dynamically defined, during object instantiation. Each block can assume two states (black and white), allowing the representation of images as bitmaps.

The data about each block are kept in a bi-dimensional array and can be altered according to the image being displayed (active image). An image is linked to an object and, at any time, the active image can be altered. The only method of this class, `clearDisplay`, is used to clean the display, changing all blocks into white. On the program interface, there are two image display areas: one for the image definition, with 30×30 blocks, and other for the structuring element definition, with 5×5 blocks.

The `imageObject` has two attributes: its own identifier, and a bi-dimensional coordinate array, that keeps the points of the active image. It has five methods: `addPixel` adds a new pair of coordinates to the array; `delPixel` removes a given pair of coordinates from the array; `showArray` sets up a string that serves to represent, textually, the coordinate's array; `showImage` shows the linked display's image; `clearAll` deletes array's coordinates.

In addition to these two classes, the tool has various functions that perform tasks related to user interaction and apply the operations, transforming the original image.

User interface User interaction is performed through four distinct areas: the brush, the main design, the structuring element, and operation definition areas (Figure 1). The brush area, located on the top-left, is used for definition of the brushes to be used for image construction. There is also a tool

to fill quickly an entire bounded area: if the clicked block is white, the corresponding area will be painted black, and if the clicked block is already painted, it acts as an eraser, removing the black points from the area. The brushes only work on the design area. Design area consists of a matrix with 30×30 blocks and is used to create and display transformed images. The displayed image will be modified according to the transformations. The working image is associated to character "A" for identification on the transformation operations. The identification label indicates which image is displayed on the screen. There is a box to display the coordinates of the image points. The "x" button on hides this box if selected by the user.

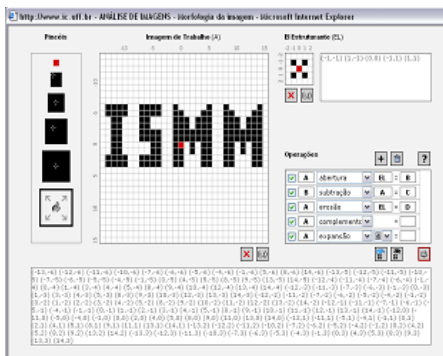


Figure 1. Tool interface.

The structuring element area, similar to the design area, simulates bit maps of 5×5 pixels for construction of structuring elements. In this area, the brush size is fixed to one point, regardless of the selected brush size. On the right hand side there is a box showing the coordinates of the structuring elements points. There are buttons for cleaning the design and for hiding or showing this box.

The operator definition area is used to define the transformations to be applied to the image. There are five control operations: add; delete; make; quick access; done and open help file. They allow, respectively, the addition of a new operation to the stack, removing of non-selected operations from the stack, making an image active in the design matrix performing the operations stored in the stack, and opening the help file.

Shortcut keys The interface has shortcut keys to the interface visual controls. TAB changes the brush size. DELETE cleans the main matrix active image. If the active image is the resulting image, the work-

ing image becomes the active image. F2 hides or exhibits the box with the active image coordinate on the main matrix. DEL (in numeric keyboard part) clears the structuring element's image. F4 hides or exhibits the box with the structuring element's coordinate. Key + (in the numeric keyboard) adds an operation to the operation stack. Key - removes the non-selected operations from the operation stack. F1 opens a pop-up window with the help file. ESCAPE makes the working image the active image of the main matrix. SPACE BAR displays the working image, without making it the active image; when the space bar is released, it displays back the resulting image. ENTER performs the operations selected from the stack.

Defining new operators To configure a sequence of transformations, the user must click the "+" button of the interface. The first selected operation must always be identified by character "A". The user then selects the operation from the list-box. Dilatation, erosion, opening, and closing operations uses the structuring element (EL). Other operations ask for definition of the neighborhood shape or for the name of another image that has already been calculated (each image may be identified by one character only). In the last field to the right of each operation one character that identifies the resulting image must be chosen. This character can be used in other operations, if one wants to apply transformations on cascade. In case of many cascade operations, the resulting image will be the answer to the last operation, or their sequence, if they have been configured on cascade form. To cascade the operations, the character of the last field has to be the same as the letter of the first field of the next operation.

3. Conclusions

The tool presented in this article only requires a browser to be executed. It is open code, and it can be used also in JavaScript classes. This tool can be downloaded from <http://www.ic.uff.br/~aconci/Morfologia> where the source-code is also available. Helps and hints are provided in Portuguese.

References

- [1] G. J. F. Banon and J. Barrera, *Bases da morfologia matemática para a análise de imagens binárias*, 2nd, INPE, São José dos Campos, 1998.
- [2] J. Serra, *Image Analysis and Mathematical Morphology*, Academic Press, London, 1982.