

# A study on discrimination of SIFT feature applied to binary images

Insaf Setitra

*Research Center on Scientific and Technical Information Cerist, Algeria  
University of Science and Technology Houari Boumediene, Algeria*

Slimane Larabi

*University of Science and Technology Houari Boumediene, Algeria*

**ABSTRACT:** Scale Invariant Feature Transform (SIFT) since its first apparition in 2004 has been (and still is) extensively used in computer vision to classify and match objects in RGB and grey level images and videos. However, since the descriptor used in SIFT approach is based on gradient magnitude and orientation, it has always been considered as texture feature and received less interest when treating binary images. In this work we investigate the power of discrimination of SIFT applied to binary images. A theoretical and experimental studies show that SIFT can still describe shapes and can be used to distinguish objects of several classes.

## 1 INTRODUCTION

Description of a shape is very important for many applications. Usually, accuracy of further processing such as matching and classification relies on the description of the shape. Morphological features are usually used to describe binary shapes. Shape Contexts (10), Graph Transduction (14), Inner Distance (11), and full and partial matching (12) are some examples of descriptors. In this work we analyze a descriptor that has been extensively used for gray level shapes but not for binary shapes. Scale Invariant Feature Transform proposed by David Lowe (2004) (7) is commonly known to be a texture feature and is used to describe gray level shapes in terms of gradient orientation and magnitude. However, previous to computing gradient orientation and magnitude, the shape is convoluted several times so that to keep strong edges. This cue is very important to describe binary shapes and makes the feature suitable for them.

The most relevant work dealing with SIFT on binary images is devoted to hand gestures classification (13) by the use of SIFT feature applied to binary masks. Authors claim to improve accuracy of classification and to decrease time processing since the descriptor is not sensitive to illumination changes and less information is needed to quantify the keypoints.

In this work, we enlarge the concept of discrimination of SIFT feature on several shapes. We first give a review of SIFT computation and study its effect on binary image. Experiments are conducted in order to verify robustness of the descriptor on both retrieval

and classification.

The remainder of the paper is organized as follows: We describe in section 2 the descriptor, in section 3 we describe our shape retrieval approach followed in section 4 by the shape classification. We present and discuss results in section 3

## 2 SHAPE DESCRIPTOR

In (7), scale invariant feature transform has been proved to be robust to affine transformations (scale and orientation) and slight changes in illumination. In (8) its robustness has been proved mathematically. Although the descriptor was successfully applied to tasks of classification, matching, and reconstruction for color and gray level images, it lacks application to binary images. Theoretical background and experiments shows that SIFT can also be applied to binary images. We describe in what follows steps of computation of the descriptor and the effect of each step on a binary image description.

### 2.1 *keypoint localization*

The most distinctive feature in a binary image is its contour, and more precisely curvatures usually contained in the edges of the image. In order to describe binary shapes we first localize these features or keypoints.

To do so, let us consider an original binary image  $I(n \times m)$ . The image is convoluted  $c$  times using a Laplacian of Gaussian filter (we describe how  $c$  is

computed below). The expression of Laplacian of Gaussian Filter is given by:

$$LoG(x, y; \sigma) = -\frac{1}{\pi\sigma^4} \left[ 1 - \frac{x^2 + y^2}{2\sigma^2} e^{-\frac{x^2 + y^2}{2\sigma^2}} \right] \quad (1)$$

Where  $x$  and  $y$  are respectively row and column coordinates in the shape to be convolved and  $\sigma$  is the standard deviation of the Gaussian distribution.

Using the Laplacian of Gaussian besides being an accurate edge detector due to its precision in detecting zeros-crossings, has many other advantages. First, the Laplacian of Gaussian is the second derivative of an image convolved with a Gaussian which prevents from noise in the image. Secondly, as the filter is pre-calculated in advance, processing time is reduced. Convoluting the image several times allows to define the zeros-crossing over different scales.

Number of convolutions  $c$  is computed as follows:

$$c = \left\lceil \frac{\log \sigma_c - \log \sigma}{\log k} \right\rceil \quad (2)$$

Where  $\lceil \cdot \rceil$  is the ceiling function which maps a real number to its smallest following integer,  $\sigma$  and  $\sigma_c$  are respectively initial and final standard deviation of the Laplacian of Gaussian Filter and are given as input parameters.  $\log$  is the natural logarithm of base 2 and  $k$  is the scale step.

At each of the  $c$  convolutions applied to  $I$ ,  $\sigma$  is updated as follows:

$$\sigma_i = \sigma \times k^{i-1} \quad (3)$$

where  $1 \leq i \leq c$  is a counter ranging from 1 to  $c$ . The initial image  $I$  is convoluted with the new  $\sigma$  using the formula 1.

Once the initial image  $I$  is convoluted  $c$  times, local maxima are defined and their coordinates (row, column indices and radius) are returned. The maxima is defined as follows:

Let  $L_i$ ,  $L_{i+1}$  and  $L_{i+2}$  be three adjacent images computed as follow:

$$L_i = LoG(x, y; \sigma_i) * I, L_{i+1} = LoG(x, y; \sigma_{i+1}) * I \\ L_{i+2} = LoG(x, y; \sigma_{i+2}) * I$$

The local maxima is first computed on the neighborhood of  $L_{i+1}$  which is the center of the current 3-tuples of convoluted images ( $L_i$ ,  $L_{i+1}$ ,  $L_{i+2}$ ). More precisely, each pixel  $p(x, y)$  in  $L_{i+1}$  is compared to its 8 neighbors. If  $p$  is either minimum or maximum, it is considered as a potential maxima of the inner scale and its coordinates are stored.

Then,  $p$  is compared to its 18 neighbors in a  $3 \times 3$  windows in  $L_i$  and  $L_{i+2}$  centered at  $(x, y)$ .

Comparisons are done the same way as in the inner scale i.e. if the central pixel in  $L_{i+1}$  is either maximum or minimum over its 18 neighbors defined above, it is

considered potential maxima and its coordinates are saved.

Once potential maxima over all scales defined, a further refinement is performed in order to eliminate weak edges. This is done by suppressing from each convoluted image  $L_{i+1}$  potential maxima which are less than an input threshold (measure of stability).

Maxima detected in this step are returned as output of this phase and are called keypoints. Each maxima is defined by its row coordinate  $r$ , column coordinate  $c$  and its radius  $rad$  which is computed as follows:

$$rad = \sigma \times k^{i-1} \times \sqrt{2} \quad (4)$$

Changes in scale allows to see the image across different resolutions. The next step is to describe keypoints detected in this step.

## 2.2 keypoint description

A binary image can be defined as a matrix which can have only two possible values typically relative to black and white. Depending on the image representation, those two values can be either 0 and 255 or 0 and 1. In a sake of simplicity we use the second representation in this paper.

Description of keypoints localized in the first step is done by first, computing the gradient magnitude and direction of the original image  $I$  around its keypoints and quantifying orientations weighted by the magnitude to form the final descriptor. We describe the whole process in what follows.

First gradient of the original image  $I$ : ( $I_x = \frac{\partial I}{\partial x}, I_y = \frac{\partial I}{\partial y}$ ) is computed. Hence, Gradient of an image can have either  $-1$ ,  $0$  or  $1$  value in both directions.

Gradient magnitude and Gradient orientation are computed. As gradient orientation is based on the gradient magnitude, 8 Gradient orientations can then take up to eight values:  $-\frac{3\pi}{4}, -\frac{\pi}{2}, -\frac{\pi}{4}, \pi, 0, \frac{3\pi}{4}, \frac{\pi}{2}, \frac{\pi}{4}$ .

Once orientations of the gradient for the whole image computed, the second step is to get for each keypoint its window computed as follows:

$$x_o = \lceil \max(1, r - rad) \rceil, y_o = \lceil \max(1, c - rad) \rceil \\ x_e = \lceil \min(r + rad, n) \rceil, y_e = \lceil \min(c + rad, m) \rceil$$

Where  $x_o, y_o, x_e, y_e$  are respectively the (row, column) of the beginning respectively end of the keypoint window.

$r, c$  and  $rad$  are row, column coordinates and radius of the keypoint.  $n$  and  $m$  are the two dimension sizes of the image  $I$ .

The next step is to divide that window into 16 blocks arranged in a  $4 \times 4$  manner (4 blocks horizontally and 4 blocks vertically). Number of pixels in each block depends on the size of the window and is rounded to its nearest integer. Histogram orientation of each block is then computed.

### 3 SHAPE RETRIEVAL

Previous steps allow us to describe binary shapes in a form of a set of keypoints. Each keypoint is a  $128 \times 1$  vector. Matching two shapes is then performed by matching each keypoint of the first image to one keypoint of the second image by computing minimal distance between them. In the following are the notation used for matching followed by the matching process.

#### 3.1 Notation used

Let  $B$  be a  $n \times 128$  matrix representing all  $n$  keypoints of an image query  $I$  and  $B_a$  be a  $n_a \times 128$  matrix representing all  $n_a$  keypoints of an image  $I_a$  of a dataset. In order to get the best match of  $I$  over all  $I_a$ 's images where  $a$  ranges from 1 to  $s$  ( $s$  is number of images in the dataset), best match of  $B$  to one  $B_a$  is computed and the process is repeated to all remaining images.

#### 3.2 Retrieval by matching keypoints independently

**Step 1:** The first match of  $B$  to  $B_a$  is done by first computing the angles between each keypoint of  $B$  to each keypoint of  $B_a$  and get the minimum value for a best match. Angles between keypoints is computed as follow:

$$Angles_a = \text{acos}(B * B_a^t) \quad (5)$$

Where  $\text{acos}$  is the inverse cosine,  $B_a^t$  is the transpose of  $B_a$  and  $*$  is the dot product of the two matrices.

**Step 2:** Angles will be a  $n \times n_a$  matrix where each line is relative to a keypoint in  $B$  and each column is relative to a keypoint in  $B_a$ . The matrix Angles is then sorted column wise in increasing order. The minimum distance is then the first column of Angles.

**Step 3:** step 1 and 2 are repeated to all  $B_a$  where  $a$  ranges from 1 to  $s$  (number of images in the dataset). For each  $B_a$ ,  $Angles_a$  are concatenated vertically so that to form the matrix  $D$  of all minimum angles between keypoints of  $B$  and  $B_a$  ( $1 \leq a \leq s$ ).  $D$  is then

$$D = Angles_1 || Angles_2 || \dots || Angles_s \quad (6)$$

Where  $||$  is a concatenation operator. Each line in  $D$  represents a keypoint and each column the minimum angle with a keypoint in the  $a^{th}$  image.

**Step 4:** Construct the vote matrix  $F$ .  $F$  takes indices of column for each line in  $D$  where  $D$  is minimum. The vector  $F$  reflects then to which image in the dataset, a keypoint in  $B$  looks more similar.

Number of lines in  $F$  is equal to  $n$  ( $n$  is number of keypoints of  $B$ ) and number of columns to 1.

**Step 5:** The last step is to compute number of occurrences of each image index in  $F$  (values in  $F$  which range from 1 to  $s$ ).  $V$  is then a  $s \times 1$  matrix which contains for each image index, number of its occurrences in  $F$ .  $V$  is then sorted in a increasing order and the first index represent the best matching image among all  $s$  images.

**Step 6:** The ancient dataset is updated where the best matching image found in step 5 is removed. If the dataset is empty or number of images( $y$ ) which should be matched is reached, then the retrieval is stopped. Otherwise steps from 1 to 6 are repeated for the query shape and the new dataset.

### 4 SHAPE CLASSIFICATION.

Classification aims to classify patterns from a dataset using a set of patterns for training. In our context, the dataset is comprised of images and is divided into two subsets: a training set and a test set. The training set is used to train the classifier and the test set is using for tests. Classification of images using SIFT is very difficult because of the high dimensionality of SIFT feature and the non-fixed feature representation. Classification using a majority vote such as done previously is very time and memory consuming since each key point of each image is stored and compared over all key points of all images of the training set. A good compromise between accuracy and computations will be to have a new representation of image features with a fixed length. The bag of features method seems to be a good solution. In what follows, we describe briefly how we use Bag of Features in classifying binary images using sift. The first step is to quantize SIFT features and build visual vocabulary using K-means and is described in subsection 4.1

Then, for a new instance, the second step is to classify it using one of supervised learning classifiers. We choose the Nearest Neighbor classification and describe it in subsection 4.2.

#### 4.1 Building visual vocabulary using K-means

SIFT features of images of training are concatenated into one  $128 \times d$  matrix  $M$  where  $d = \sum_{i=1}^t n_i$ ,  $n_i$  is number of keypoints of image  $I_i$  and  $t$  is the number of images of the training set.

Rows of the matrix  $M$  are then clustered using  $K - means$  clustering algorithm described in algorithm 1

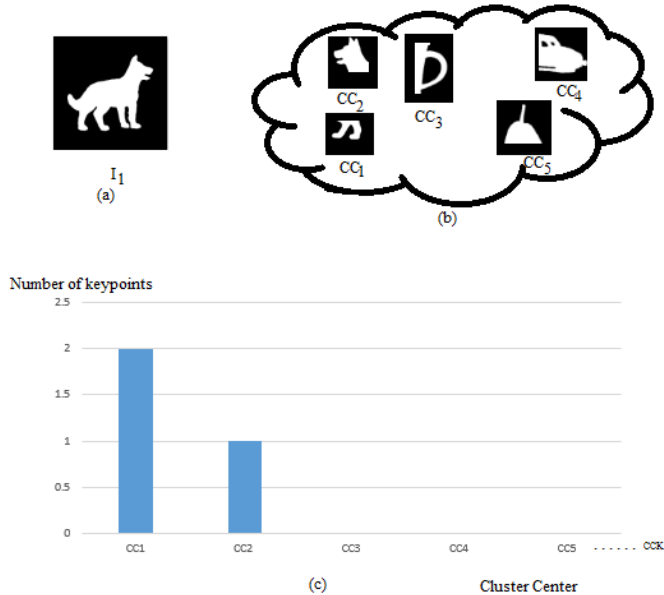


Figure 1: Example of Bag of Features representation of an image  $I$  (a) the image  $I$  to be represented by the visual vocabulary, (b) Visual Vocabulary represented by its centers, (c) Bag of Features of the image  $I$ .

---

#### Algorithm 1 $K - means$ Algorithm

---

**Begin** -Input  $l$  number of clusters

-Input number of iterations

-Choose initial center of the  $l$  clusters

**Repeat**

-Compute Euclidean Distance between the feature and each center

-Assign the feature to the cluster which minimizes the Euclidean Distance

-Recompute the center of each cluster

**UNTIL** stabilization of the centers is observed or number of iterations is reached

**End.**

---

Result of clustering is then  $l$  stabilized clusters, each cluster is represented by its center which is a one of the 128 dimensional vector of keypoints and represents the visual vocabulary. Each image of the training and test set is then represented by a histogram where number of bins is  $l$  and each bin stands for number of keypoints nearest to one of the  $l$  centers. Figure 1 represents an example of Bag of Words representation.

#### 4.2 Training and classification using bag of words

After having Bag of Features descriptor of each of training and test images, the distance between Bag of Features of test images and training images is computed, for each test image, the minimal distance is chosen, and label of the nearest training image is attributed to the test image. We use two distances, Euclidean and  $\chi^2$ .  $\chi^2$  distance is described as follows:

Let  $HI_i$  be the feature vector representing the image  $I_i$ . Note that This feature if the normalized histogram

of visual words of the image  $I_i$ . Let  $HT_i$  be one of the feature vector representing on of the training examples.  $\chi^2$  distance is described as:

$$\chi^2 = \sum_{i=1}^l \frac{HI_i - HT_i^2}{HT_i} \quad (7)$$

where  $l$  is the dimension of the feature vector equal to number of clusters of the  $K - means$  algorithm.

## 5 EXPERIMENTAL RESULTS

### 5.1 Shape Retrieval accuracy of the dataset ETH-80

Retrieval experiments are done on the dataset ETH-80 (9). The dataset comprises 8 categories each of which has 400 images. We choose randomly 10 images of each category which results in 80 images in the dataset prepared for retrieval. Then, we chose randomly an image from each category for retrieval. We use the bullseye score for retrieval accuracy measured as the percent of correct retrievals in the top 20 ranks. Table 1 shows retrieval accuracy of retrieval. The first column represents the query shape, the second column represents shapes retrieved in the order of similarity i.e. similar shapes start in the top left of the column and are ordered line wise. Names of shapes are the same as in the original database <sup>1</sup>. The third column stands for the processing time. The table shows that best rates are for shapes with few curvatures whereas weak rates are observed for shapes with high curvatures. This is because keypoints in curvatures are repetitive in many shapes. However, processing time for shapes with few curvatures is higher. Besides, shapes with few curvatures are most often matched to shapes with few curvatures, examples include: apple, pear and tomato. However, some shapes of high curvatures are confused with shapes with few curvatures. The reason is that, many keypoints in a shape can be matched to only one keypoint in another shape and distance is minimal. The opposite is also true. One solution could be to match a keypoint in a query shape to only one keypoint of a shape in the dataset. We let this improvement to a future work.

### 5.2 Classification accuracy of the dataset ETH-80

Experiments on the dataset ETH-80 are divided into two parts. The first part concern classification accuracy of RGB images, the second part concerns classification accuracy of the same binary images.

Extensive experiments of ETH-80 are done by dividing the whole dataset into training and classification. Division is done following several percentages from 10 to 70% where this percentage represents ratio of

<sup>1</sup><http://people.csail.mit.edu/jjl/libpmk/samples/eth.html>

Table 1: Retrieval accuracy on ETH-80 dataset

Query Image	Bullseye score (%)	Processing Time (ms)
apple20	100	135.58
car4	50	53.69
cow2	50	55.67
cup15	100	133.21
dog31	50	88.36
horse20	80	82.72
pear20	100	87.36
tomato36	100	107.81

training set with the whole dataset. In a first experiment, we vary size of both training and test set where sum of their two ratios is equal to 100%. By doing this, we insure not to include images of the training set into the test set. Table 2 represents accuracy of this variation. In a second experiment, we choose a small ratio of the test set (30%) and fix it for all percentages of training. Table 3 represents this experiment. Size of the training set in both cases do not exceed 70% which represent the classical percentage used in most classification experiments (2/3 training and 1/3 test). We analyze classification accuracy of RGB images and their binary masks (binary images) using the same parameters of the code book (same number of iterations of K-means clustering, same number of features in the visual vocabulary, same distance measure).

Table 2 and 3 show that classification accuracy of RGB images in most cases outperform the one of the binary images of the original RGB images. However, the difference is slight and in one case, classification of binary images outperformed classification of RGB images. In order to analyze more deeply the classification results, we moreover generate for each ratio confusion matrices. We generate more than 60 confusion matrices by varying the size of both training and test set, however, we kept only the most significant ones which correspond to ratios of table 2. Confusion matrices show that classification accuracy is enhanced in RGB images compared to binary images, however, the difference is still slight. Also, in some cases, classification of RGB images. Some interesting points can be seen in the confusion matrices: Confusion of RGB images can occur even when the shape is different, example of such a confusion can be seen in figure 2 where horses have been confused with cows. This confusion is less apparent in the binary images and accuracy of horse classification can reach 100% for binary images while it does not for RGB images. Such errors occurred for horses with same colors as cows and dogs and when some animals have same skin (texture). The same observation can be made for classes "Dog" and "Cow". For all other classes, classification accuracy of RGB images is slightly higher than classification accuracy of the same binary images.

Table 2: Overall classification accuracy on ETH-80 dataset by varying size of training set and test set

Training %	Test %	Accuracy % Binary	Accuracy % RGB
20	80	83.01	90.22
40	60	93.91	96.79
60	40	99.03	98.71

Table 3: Overall classification accuracy on ETH-80 dataset by varying size of training set and fixing size of test set

Training %	Test %	Accuracy % Binary	Accuracy % RGB
10	30	77.77	84.61
20	30	84.82	88.35
30	30	90.49	93.05
40	30	92.20	94.87
50	30	94.23	95.51
60	30	96.68	97.75
70	30	99.14	99.25

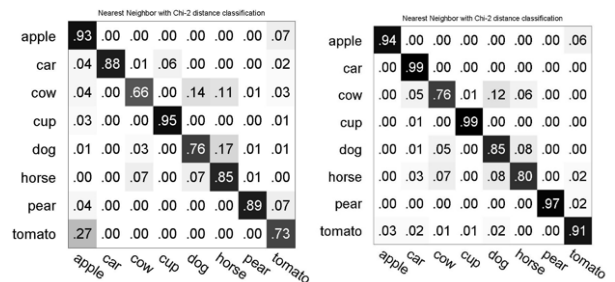


Figure 2: Confusion matrices for ETH-80 with training size 20%, test size 80% (right matrix binary images, left matrix RGB images).

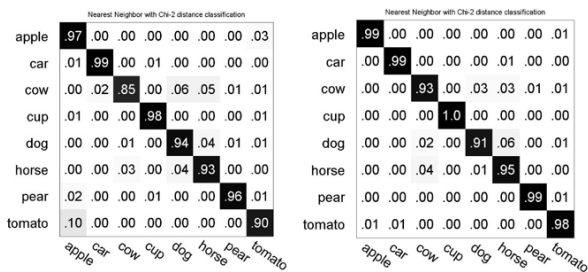


Figure 3: Confusion matrices for ETH-80 with training size 40%, test size 60% (right matrix binary images, left matrix RGB images).

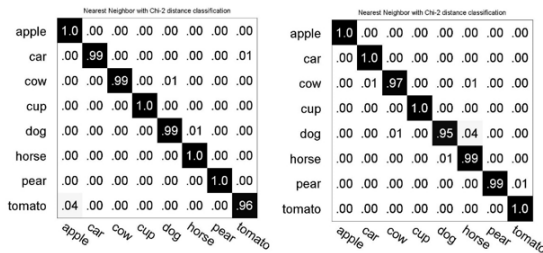


Figure 4: Confusion matrices for ETH-80 with training size 60%, test size 40% (right matrix binary images, left matrix RGB images).

## 6 CONCLUSION

We have presented in this work the Scale Invariant Feature Transform SIFT feature and its application to binary images. We showed theoretically how the feature can be applied to binary image and enhance the study with experiments both on retrieval and classification. The study concludes that SIFT can be applied to binary shapes. We are in further works to improve the matching and apply the approach on binary masks derived from background subtraction (15).

## REFERENCES

- [1] Veltkamp, R.C., Shape matching: similarity measures and algorithms, Shape Modeling and Applications, SMI 2001 International Conference on. 2001.
- [2] Learning Context-Sensitive Shape Similarity by Graph Transduction, Xiang Bai and Yang, Xingwei and Latecki, L.J. and Wenyu Liu and Zhuowen Tu, Pattern Analysis and Machine Intelligence, IEEE Transactions on, 2010.
- [3] Xiang Bai and Cong Rao and Xinggang Wang, Shape Vocabulary: A Robust and Efficient Shape Representation for Shape Matching, Image Processing, IEEE Transactions on, 2014.
- [4] Haibin Ling and Jacobs, D.W., Using the inner-distance for classification of articulated shapes, Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on,
- [5] J.F. Nunes, P.M. Moreira, J.M.R.S. Tavares., Simple and fast shape based image retrieval, Computational Vision and Medical Image Processing: VipIMAGE 2011.

- [6] Bouagar, Saliha and Larabi, Slimane, Efficient descriptor for full and partial shape matching Multimedia Tools and Applications, 2014.
- [7] Lowe, David G., Distinctive Image Features from Scale-Invariant Keypoints, Int. J. Comput. Vision, 2004.
- [8] Jean-Michel Morel and Guoshen Yu, Is SIFT scale invariant?, Inverse Problems and Imaging, 2011.
- [9] Leibe, B. and Schiele, B., Analyzing appearance and contour based methods for object categorization Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on.
- [10] Belongie, S. and Malik, J. and Puzicha, J., Shape matching and object recognition using shape contexts, Pattern Analysis and Machine Intelligence, IEEE Transactions on, 2002.
- [11] Haibin Ling and Jacobs, D.W., Shape Classification Using the Inner-Distance, Pattern Analysis and Machine Intelligence, IEEE Transactions on. 2007,
- [12] Bouagar, Saliha and Larabi, Slimane, Efficient descriptor for full and partial shape matching, Multimedia Tools and Applications, 2014.
- [13] Lin, Wei-Syun and Wu, Yi-Leh and Hung, Wei-Chih and Tang, Cheng-Yuan, A Study of Real-Time Hand Gesture Recognition Using SIFT on Binary Images, Advances in Intelligent Systems and Applications - Volume 2, 2013;
- [14] Xiang Bai and Xingwei Yang and Longin Jan Latecki and Wenyu Liu and et al., Learning Context-Sensitive Shape Similarity by Graph Transduction, 2010.
- [15] Setitra, I. and Larabi, S., Pattern Recognition (ICPR), 2014 22nd International Conference on, Background Subtraction Algorithms with Post-processing: A Review, 2014.