Accepted version of paper published in International Conference on Multimedia Retrieval (ICMR), 2017.
DOI: https://doi.org/10.1145/3078971.3079017

An Unsupervised Distance Learning Framework for Multimedia Retrieval

Lucas Pascotti Valem and Daniel Carlos Guimarães Pedronette
Department of Statistics, Applied Math. and Computing, São Paulo State University (UNESP)
Rio Claro, Brazil
lucasvalem@rc.unesp.br,daniel@rc.unesp.br

ABSTRACT

Due to the increasing availability of image and multimedia collections, unsupervised post-processing methods, which are capable of improving the effectiveness of retrieval results without the need of user intervention, have become indispensable. This paper presents the *Unsupervised Distance Learning Framework* (UDLF), a software which enables an easy use and evaluation of unsupervised learning methods. The framework defines a broad model, allowing the implementation of different unsupervised methods and supporting diverse file formats for input and output. Seven different unsupervised methods are initially available in the framework. Executions and experiments can be easily defined by setting a configuration file. The framework also includes the evaluation of the retrieval results exporting visual output results, computing effectiveness and efficiency measures. The source-code is public available, such that anyone can freely access, use, change, and share the software under the terms of the GPLv2 license.

CCS CONCEPTS

• **Information systems** → **Multimedia and multimodal retrieval**;

KEYWORDS

content-based image retrieval; unsupervised learning; reranking; rank-aggregation

1 INTRODUCTION

The facilities in image acquisition and sharing available nowadays have been resulted in profound changes in human lifestyle. Mainly supported by the development of mobile devices, social networks, and cloud environments, visual content have become the mean of communication for a growing number of people. Considering the huge and increasing amount of image collections available, the development of automatic methods for analysing, indexing and searching the visual content became indispensable.

In this scenario, Content-Based Image Retrieval (CBIR) systems have been established as a central solution [7]. The main objective of such systems consists in retrieving relevant images ranked according to their similarity to a query input. Initially, the ranking tasks relied only in the comparison between images based on their low-level features. Therefore, for many years, the progress of CBIR systems have been mainly supported by the development of diverse visual features [2]. In order to provide a common and public available implementation of such features, open-source tools as LIRE [9] and Eva [20] have been proposed.

However, the retrieval model grounded on pairwise comparisons computed between feature vectors faces serious challenges. Such model analyzes similarity only in terms of pairs of images, ignoring the global relationships encoded in the dataset. More recently, various methods [1, 3–5, 14, 15, 17, 18, 21, 25–28] have focused on other stages of the retrieval process, post-processing the initial results in order to improve the retrieval effectiveness, without the need of user intervention. Generally, such methods compute unsupervised global affinity measures capable of considering the intrinsic manifold structure of datasets. Although such methods represent an indispensable tool for improving the retrieval results, few of them are public available and, when available, require specific software environments and input/output formats. In this scenario, a common tool capable of executing different methods under a unified environment is missing.

In this work, we introduce the *Unsupervised Distance Learning Framework* (UDLF), which provides a software environment to easily implement, use, and evaluate unsupervised post-processing methods. The framework defines a general model, allowing the implementation of different methods, based on distance measures or rank information. The user can easily select the method to be executed and set the respective parameters. Different file formats (for both input and output) are supported, and effectiveness and efficiency evaluation are also available. The framework includes the implementation of seven different unsupervised methods and is licensed under the terms of the GPLv2, such that it can be easily extended.

The paper is organized as follows: Section 2 presents the main aspects of the framework and describes the methods currently implemented; Section 3 describes the use of the framework and discusses some examples. Finally, Section 4 discusses the conclusions and presents possible future works.

ICMR '17, June 06-09, 2017, Bucharest, Romania

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM. . . $15.00
https://doi.org/10.1145/3078971.3079017

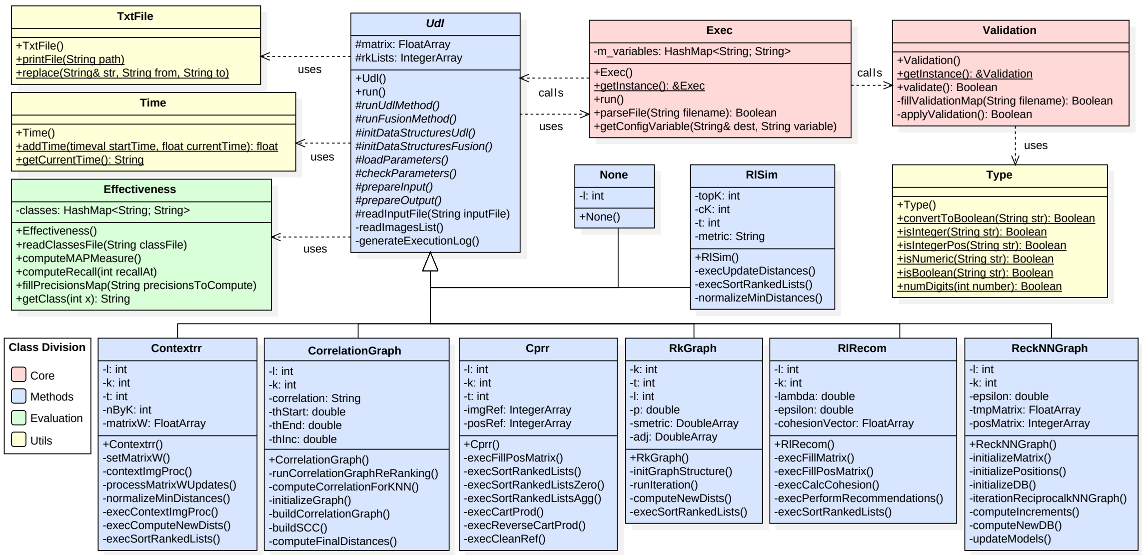

Figure 1: General organization of the UDLF proposed framework in a UML class diagram.

2 UDL FRAMEWORK

As discussed, several unsupervised post-processing methods have been proposed recently aiming at improving the effectiveness of multimedia retrieval tasks. However, the implementation of such methods are not always public available and, when provided, they do not follow any standard. In this way, any person interested in using such methods is required to understand the implementation, parameters configuration, and file formats for input and output. This process is necessary for each method, therefore limiting the widespread use of such relevant tools. This paper aims to fill this gap by introducing the *Unsupervised Distance Learning Framework* (UDLF). The main objective consists in providing a software environment which includes the follow requirements:

- **General and extensible model:** the framework defines a broadly unsupervised distance learning model, which can be used for implementing different methods. The framework was initially validated considering seven distinct methods (discussed in Section 2.2) and is ready to extensions.
- **Flexible input/output:** the retrieval results can be read and exported in different file formats, defined in terms of distance measures or ranking information.
- **Easy use and configuration:** the code is compiled once and different executions can be done just by changing file paths and parameter values in a configuration file, since no installation is required. This allows the user to easily perform experiments using different methods and datasets.
- **Evaluation:** in addition to the processed output files, the framework includes evaluation information considering both effectiveness and efficiency aspects. The framework reports measures as Precision, Recall and MAP (Mean Average Precision).

The framework is an open-source software licensed under the terms of GPLv2. The framework is public available[1], allowing anyone to access, use, and contribute with the code.

[1] https://github.com/UDLF/UDLF

2.1 Language and Organization

UDLF is designed trough a object-oriented paradigm and implemented using C++ 2011. The framework is independent of external libraries and portable among different operation systems. Figure 1 presents a UML class diagram, which illustrates the general organization of the framework, including the most important functions and attributes for each class. The colors represent directories in which the classes are divided, according to their role.

The execution flow starts from the `Core` classes, where `Exec` parses the configuration file reading the parameters values. The `Validation` class verifies if all the attributed values are lexically acceptable. Otherwise, they are set to the default values. All the classes of `Methods` are a generalization of `Udl`, an abstract class which establishes a protocol to implement new methods. Regarding the remaining classes, while `Utils` encompasses all static classes that have auxiliary purposes, `Evaluation` implements the necessary measures to evaluate the results. As can be seen, different effectiveness measures are currently implemented in the `Effectiness` class. To compute the efficiency, the `Time` class is used as an utility by the `Udl` class.

2.2 Distance Learning Methods

This section presents a briefly description of the unsupervised methods currently implemented in the framework.

- **Correlation Graph Manifold Learning** [**16**]: the algorithm builds a correlation graph for encoding the dataset similarity information. The dataset manifold is analysed through the graph and its strongly connected components, in order to compute a more effective distance measure.
- **Ranked List Graph Distance** [**12**]: the method represents each ranked list as a weighted sub-graph based on its top-k positions. Subsequently, the sub-graphs are combined in a single graph and the weight of the edges is used to increment the similarity scores between images.

• **RL-Sim* [11, 15]**: the algorithm is defined in terms of an iterative re-ranking method that computes a similarity score between images based on the similarity of their ranked lists. The algorithm is motivated by the conjecture that if two images are similar, their ranked lists should also be similar.

• **CPRR [22]**: the Cartesian Product of Ranking References (CPRR) algorithm applies the Cartesian product operations to kNN and reverse kNN sets aiming at considering the contextual information encoded in the ranked lists.

• **Manifold Reciprocal kNN Graph [19]**: the method builds a graph considering the reciprocal references in the top-k positions of the ranked lists. The method exploits the graph structure through scores in order to analyze the dataset manifold and compute new ranked lists.

• **RL-Recom [23]**: the algorithm exploits the concept of recommendations among ranked lists. The motivation is based on the conjecture that similarity information available in the ranked lists can be used to recommend images among themselves. In this scenario, a recommendation indicates a reduction in the distance measures.

• **ContextRR [13]**: the method uses context images, which are abstractions obtained from the ranked lists and distance measures to make an analysis of the dataset as a whole. Based on information extracted from context images, a more effective distance measure is computed.

3 FRAMEWORK USAGE

The framework usage is mainly based on the configuration file, which specifies all information about the execution: the desired task, method being used, dataset information, input and output files, evaluation settings, and other details. In this way, no recompilation is necessary, such that the user is able to perform a totally different execution just by changing the configuration file. The software considers only a single configuration file per execution, allowing the user to have distinct configuration files for different executions.

A command line interface, which varies according to the operating system, must be used to execute the software. When the binary is executed, a `config.ini` file is searched in the current directory. The user can also specify a different configuration file as a parameter: `./udlf my_conf.ini`. The next sub-sections briefly describe the structure of the configuration file and usage examples. More detailed information can be found at https://github.com/UDLF/UDLF/wiki.

3.1 Configuration File

The configuration file syntax is at the same time very simple and powerful. Basically, values are attributed to parameters using the expression: `PARAMETER = VALUE`. All the parameters are upper case and have words separated by _. If a nonexistent parameter is used, the attribution is ignored. If the attributed value is invalid, the parameter is set to its default value, defined in internal configuration files. Comments can be included using the character `#`, such that all the text after it is ignored.

Listing 1 shows an example of configuration file. The comments contain explanations about parameter meaning and its acceptable values (provided by the regular expressions). For a better organization, the parameters were separated into five different categories, which are described in the following.

Listing 1: Configuration file example.

```
0  #The comments follow the structure:
1  #PARAMETER = VALUE #(regular expression): Explanation about the parameter
2  #If a regular expression is not specified, any input string can be used
3  #To simplify the expressions, we adopt:
4  #TBool  = (TRUE|FALSE)
5  #TUInt  = (0-9)*
6  #TFloat = ["+"|"-"] [0-9]* ["."] [0-9]+
7
8  #CATEGORY 1. GENERAL CONFIGURATION
9  UDL_TASK = UDL #(UDL|FUSION): Selection of task to be executed
10 UDL_METHOD = CPRR #(NONE|CPRR|RLRECOM|RLSIM|CONTEXTRR|RECKNNGRAPH|RKGRAPH|
   CORGRAPH): Selection of method to be executed
11 #CATEGORY 2. INPUT FILE SETTINGS
12 SIZE_DATASET = 1400 #(TUint): Number of images in the dataset
13 INPUT_FILE_FORMAT = MATRIX #(MATRIX|RK): Format of input file
14 INPUT_MATRIX_TYPE = DIST #(DIST|SIM): Type of matrix file
15 INPUT_RK_FORMAT = NUM #(NUM|STR): Format of ranked list file
16 MATRIX_TO_RK_SORTING = HEAP #(HEAP|INSERTION): Convert matrix to rks
17 NUM_INPUT_FUSION_FILES = 2 #(TUint): Number of files for FUSION tasks
18 INPUT_FILES_FUSION_1 = input1.txt #Path of the first input file
19 INPUT_FILES_FUSION_2 = input2.txt #Path of the second input file
20 #INPUT_FILES_FUSION_* = input*.txt #Path of the *th input file
21 INPUT_FILE = input.txt #Path of the main input file (matrix/rks)
22 INPUT_FILE_LIST = list.txt #Path of the list file
23 INPUT_FILE_CLASSES = classes.txt #Path of the classes file
24 INPUT_IMAGES_PATH = images/ #Dataset images path
25 #CATEGORY 3. OUTPUT FILE SETTINGS
26 OUTPUT_FILE = TRUE #(TBool): Generate output file(s)
27 OUTPUT_FILE_FORMAT = MATRIX #(RK|MATRIX): Format of output file
28 OUTPUT_MATRIX_TYPE = DIST #(DIST|SIM): Type of matrix file to output
29 OUTPUT_RK_FORMAT = ALL #(NUM|STR|HTML|ALL): Output format for rks
30 OUTPUT_FILE_PATH = output #Path of the output file(s)
31 OUTPUT_HTML_RK_PER_FILE = 1 #(TUint): Number of rks for each html file
32 OUTPUT_HTML_RK_SIZE = 20 #(TUint): Number of images per ranked list
33 OUTPUT_HTML_RK_COLORS = TRUE #(TBool): Color borders around images
34 OUTPUT_HTML_RK_BEFORE_AFTER = TRUE #(TBool): Comparison of rks
35 #CATEGORY 4. EVALUATION SETTINGS
36 EFFICIENCY_EVAL = TRUE #(TBool): Enable efficiency evaluation
37 EFFECTIVENESS_EVAL = TRUE #(TBool): Enable effectiveness evaluation
38 EFFECTIVENESS_COMPUTE_PRECISIONS = TRUE #(TBool): Compute precisions
39 EFFECTIVENESS_COMPUTE_MAP = TRUE #(TBool): Compute MAP
40 EFFECTIVENESS_COMPUTE_RECALL = TRUE #(TBool): Compute recall
41 EFFECTIVENESS_RECALL_AT = 40 #(TUint): Position to compute recall
42 EFFECTIVENESS_PRECISIONS_TO_COMPUTE = 5, 20 #(TUint ["," TUint]*):
   Precisions to be computed (unsigned integers separated by commas)
43 #CATEGORY 5. METHOD PARAMETERS
44 PARAM_CPRR_L = 400 #(TUint): Size of ranked lists to consider
45 PARAM_CPRR_K = 20  #(TUint): Number of nearest neighbors
46 PARAM_CPRR_T = 2   #(TUint): Number of iterations
```

1. General configuration: the execution type and the method to be executed are defined in lines 9 and 10, respectively. For the execution types, the available options are re-ranking (traditional distance learning) and rank-aggregation (fusion distance learning). Regarding the methods, there is an option called `NONE`, which can be used to easily perform an execution to change file formats or evaluate files.

2. Input file settings: the main input files encode similarity information, generally provided by visual descriptors. The files are supported both in the ranked lists (numeric or string) or matrices (similarity or distance) format. Lines 13, 14, and 15 define the format to be used. While only one main input file is considered for `UDL` tasks (line 21), `FUSION` tasks consider two or more main input files (lines 17-20). Lines 22, 23 and 24 specify the list file, the classes file, and the images path (used to export visual results to `html`), respectively. The list file is always required because it offers the name of each image. However, the classes file is only necessary when effectiveness evaluation is enabled. Internally, the framework can convert matrices to ranked lists using the sorting method specified in line 16. Complete examples of input files for distinct datasets are available at https://github.com/UDLF/Datasets.

3. Output file settings: analogous to the input files, the output format can also be configured. Additionally, there is a feature for exporting ranked lists to `HTML` files, which provides visual results (customized in lines 31-34).

4. Evaluation settings: an evaluation report can be provided in the end of execution, containing information as the time elapsed (line 36) and effectiveness measures as Precision, Recall, and MAP (lines 38, 39 and 40, respectively). Recall and Precision measures can be computed considering different depths of the ranked lists (lines 41 and 42).

5. Method parameters: definition of parameters settings, according to the method being executed (CPRR [22] in this example, lines 44-46).

3.2 Input/Output File Formats

The framework input is composed by three files: list, classes and ranked lists (or matrix). The list file considers each line as an *identifier* for a dataset element, as shown in Listing 2. For the effectiveness evaluation, each image is associated to a class, information that is defined in the classes file using the expression: `image:class`. An example of a classes file is presented in Listing 3. Ranked lists can be represented in both numeric or string format. A string example is shown in Listing 4. Each line corresponds to a ranked list and the name of the images are separated by spaces. The output file is generated according to the same format.

Listing 2: List file.

```
apple1.png
apple2.png
bird1.png
bird2.png
bat1.png
bat2.png
```

Listing 3: Classes file.

```
apple1.png:apple
apple2.png:apple
bird1.png:bird
bird2.png:bird
bat1.png:bat
bat2.png:bat
```

Listing 4: Ranked list file example - string format.

```
apple1.png apple2.png bird1.png bat1.png bat2.png bird2.png
apple2.png apple1.png bird2.png bird1.png bat1.png bat2.png
bird1.png bird2.png bat2.png apple2.png apple1.png bat1.png
bird2.png bird1.png bat2.png apple1.png apple2.png bat1.png
bat1.png bat2.png apple1.png apple2.png bird2.png bird1.png
bat2.png apple1.png apple2.png bat1.png bird2.png bird1.png
```

3.3 Execution Reports and Visual Results

The framework also generates execution reports and visual retrieval results. Listing 5 shows an example of an execution report (`log.txt`) generated by the framework.

Listing 5: Example of `log.txt` file.

```
- GENERAL INFORMATION -
Task:               UDL
Method:             CPRR
Dataset Size:       1400
Image List File:    desc/lists/mpeg7.txt
Image Class File:   desc/classes/mpeg7.txt
Input File:         desc/matrices/mpeg7/cfd.txt
Input Format:       MATRIX DIST
Output File:        output/output
Output Format:      RK ALL
--------------------------------------
- METHOD PARAMETERS -
PARAM_CPRR_K = 20
PARAM_CPRR_L = 400
PARAM_CPRR_T = 2
--------------------------------------
- EVALUATION RESULTS -
* Efficiency: Total Time of the Algorithm Execution: 0.0438 s
* Effectiveness:
Before:
    P@20        0.7559
    Recall@40   0.8444
    MAP         0.8064
After:
    P@20        0.8979
    Recall@40   0.9477
    MAP         0.9215
Relative Gains:
    P@20        +18.7866%
    Recall@40   +12.2404%
    MAP         +14.2707%
--------------------------------------
Log generated at 2017/1/26 16:37:24
```

The report is organized in sections, providing various information about the framework execution. Firstly, general information about the input/output files are shown. Next, the parameters of the executed method are presented. Finally, effectiveness and efficiency measures are reported, including the effectiveness gains obtained.

Figure 2 presents visual examples of ranked lists exported from executions of the framework considering different image datasets (Corel5k [8], MPEG-7 [6], OxfordFlowers-17 [10] and Soccer [24], respectively). The query images are presented in green borders and wrong results in red borders. The first line represents the original retrieval results and the second line, the results after the algorithm execution.

Figure 2: Visual examples showing the impact of distance learning on retrieval results.

4 CONCLUSIONS

In this work, we have presented UDLF, an open-source software that currently contains seven different unsupervised distance learning methods for multimedia retrieval. As future work, we intend to keep improving the software incorporating new features and other methods from different authors. We also intend to provide scripts, facilitating conversions between file formats and providing a visual interface to generate the configuration file. This facilitates the execution of experiments for users that do not want to use the command line. Aiming at maximizing the performance of the algorithms, parallel computing versions of the methods can also be implemented. We intend to provide more information about how the framework can be expanded, allowing the scientific community to contribute to the software.

5 ACKNOWLEDGMENTS

The authors are grateful to São Paulo Research Foundation - FAPESP (grants 2013/08645-0 and 2014/04220-8).

REFERENCES

[1] S. Bai and X. Bai. Sparse contextual activation for efficient visual re-ranking. *IEEE Transactions on Image Processing*, 25(3):1056–1069, March 2016.

[2] R. Datta, D. Joshi, J. Li, and J. Z. Wang. Image retrieval: Ideas, influences, and trends of the new age. *ACM Computing Surveys*, 40(2):5:1–5:60, 2008.

[3] M. Donoser and H. Bischof. Diffusion Processes for Retrieval Revisited. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1320–1327, 2013.

[4] H. Jegou, C. Schmid, H. Harzallah, and J. Verbeek. Accurate image search using the contextual dissimilarity measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(1):2–11, 2010.

[5] J. Jiang, B. Wang, and Z. Tu. Unsupervised metric learning by self-smoothing operator. In *Internation Conference on Computer Vision (ICCV)*, pages 794–801, 2011.

[6] L. J. Latecki, R. Lakmper, and U. Eckhardt. Shape descriptors for non-rigid shapes with a single closed contour. In *CVPR*, pages 424–429, 2000.

[7] M. S. Lew, N. Sebe, C. Djeraba, and R. Jain. Content-based multimedia information retrieval: State of the art and challenges. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 2(1):1–19, 2006.

[8] G.-H. Liu, L. Zhang, Y.-K. Hou, Z.-Y. Li, and J.-Y. Yang. Image retrieval based on multi-texton histogram. *Pattern Recogn.*, 43(7):2380–2389, July 2010.

[9] M. Lux. Content based image retrieval with LIRE. In *ACM Multimedia (MM'11)*, pages 735–738, 2011.

[10] M.-E. Nilsback and A. Zisserman. A visual vocabulary for flower classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 1447–1454, 2006.

[11] C. Y. Okada, D. C. G. a. Pedronette, and R. da S. Torres. Unsupervised distance learning by rank correlation measures for image retrieval. In *ACM International Conference on Multimedia Retrieval (ICMR'15)*, pages 331–338, 2015.

[12] D. C. G. Pedronette, J. Almeida, and R. da S. Torres. A graph-based ranked-list model for unsupervised distance learning on shape retrieval. *Pattern Recognition Letters*, 83, Part 3:357 – 367, 2016.

[13] D. C. G. Pedronette and R. da S. Torres. Exploiting contextual information for image re-ranking and rank aggregation. *International Journal of Multimedia Information Retrieval*, 1(2):115–128, 2012.

[14] D. C. G. Pedronette and R. da S. Torres. Exploiting pairwise recommendation and clustering strategies for image re-ranking. *Information Sciences*, 207:19–34, 2012.

[15] D. C. G. Pedronette and R. da S. Torres. Image re-ranking and rank aggregation based on similarity of ranked lists. *Pattern Recognition*, 46(8):2350–2360, 2013.

[16] D. C. G. Pedronette and R. da S. Torres. A correlation graph approach for unsupervised manifold learning in image retrieval tasks. *Neurocomputing*, 208:66 – 79, 2016.

[17] D. C. G. Pedronette and R. da S. Torres. Rank diffusion for context-based image retrieval. In *ACM International Conference on Multimedia Retrieval (ICMR'2016)*, 2016.

[18] D. C. G. Pedronette and R. da Silva Torres. A correlation graph approach for unsupervised manifold learning in image retrieval tasks. *Neurocomputing*, 208:66–79, 2016.

[19] D. C. G. Pedronette, O. A. Penatti, and R. da S. Torres. Unsupervised manifold learning using reciprocal knn graphs in image re-ranking and rank aggregation tasks. *Image and Vision Computing*, 32(2):120 – 130, 2014.

[20] O. A. Penatti and R. d. S. Torres. Eva: An evaluation tool for comparing descriptors in content-based image retrieval tasks. In *International Conference on Multimedia Information Retrieval*, MIR '10, pages 413–416, 2010.

[21] D. Qin, S. Gammeter, L. Bossard, T. Quack, and L. van Gool. Hello neighbor: Accurate object retrieval with k-reciprocal nearest neighbors. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 777 –784, 2011.

[22] L. P. Valem and D. C. G. Pedronette. Unsupervised similarity learning through cartesian product of ranking references for image retrieval tasks. *Conference on Graphics, Image and Patterns (SIBGRAPI)*, 2016.

[23] L. P. Valem, D. C. G. Pedronette, R. d. S. Torres, E. Borin, and J. Almeida. Effective, efficient, and scalable unsupervised distance learning in image retrieval tasks. *ACM International Conference on Multimedia Retrieval (ICMR)*, 2015.

[24] J. van de Weijer and C. Schmid. Coloring local feature extraction. In *European Conference on Computer Vision (ECCV)*, volume Part II, pages 334–348, 2006.

[25] J. Wang, Y. Li, X. Bai, Y. Zhang, C. Wang, and N. Tang. Learning context-sensitive similarity by shortest path propagation. *Pattern Recognition*, 44(10-11):2367–2374, 2011.

[26] X. Yang, X. Bai, L. J. Latecki, and Z. Tu. Improving shape retrieval by learning graph transduction. In *European Conference on Computer Vision (ECCV)*, volume 4, pages 788–801, 2008.

[27] X. Yang, S. Koknar-Tezel, and L. J. Latecki. Locally constrained diffusion process on locally densified distance spaces with applications to shape retrieval. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 357–364, 2009.

[28] X. Yang, L. Prasad, and L. Latecki. Affinity learning with diffusion on tensor product graph. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 35(1):28–38, 2013.