Technical Report on A Generic Multi-Dimensional Data Generator for Earth Mover’s Distance Similarity Analysis

Rui Zhang, Jin Huang
Department of Computing and Information Systems
University of Melbourne
{rui.zhang, jin.huang}@unimelb.edu.au

Abstract
Earth Mover’s Distance based Similarity Analysis (EMDSA) is an important and effective tool in many multimedia retrieval and pattern recognition applications. Currently there is no benchmark or publicly available datasets for evaluating EMDSA techniques. We would like to share a large-scale image feature generator we have designed and implemented for evaluating EMDSA techniques. The generator supports a wide range of image features (16 commonly used features are embedded in). It provides flexible execution options: (i) it may run on a single machine or on a Hadoop cluster; (ii) the source image data may reside locally in a single machine, in a Hadoop file system, or on the Internet as URLs. We have made the source code of our generator and dozens of pre-generated datasets available online.

1 Introduction
Image sharing social networks such as Instagram and Flickr receive hundreds of millions of images uploaded by users per day. There is an emerging need for processing and analyzing such “big” image data to innovate the critical image-related applications in various fields. Earth Mover’s Distance based Similarity Analysis (EMDSA) is an important and effective tool in many multimedia retrieval and pattern recognition applications. Earth Mover’s Distance (EMD) is a transformation based distance metric and is generally perceived as a better similarity measure than $\ell_p$ distance because it considers the spatial relationships between dimensionality and allows small shifts in arbitrary features, simulating the human perception when comparing images [15]. For example, for the two pictures in Figure 1, the $\ell_p$ metric may result in a significant distance due to the viewpoint shifts while EMD can capture these shifts easily and therefore produces only a small distance value. A more concrete example is given in Section 2.1.

Figure 1: Small shifts of colors or shapes may lead a significant $\ell_p$ distance while EMD can capture the similarity by shifting pixels

EMD has been extensively studied in the content based image retrieval community and has been applied in many applications [8, 17]. Recently, efficiently processing and managing data for EMDSA has attracted enormous attention [1, 21, 22, 16], all of which focus on algorithms optimized on a single machine and handle a limited amount of data. Huang et al. [9] made the first attempt to perform EMD based similarity joins on large scale data, and proposed a framework named MELODY-JOIN.
However, we are lack of benchmark datasets or a publicly available data generator for large-scale EMDSA. Meanwhile, collecting and generating data may be tedious because different applications may require different types of image features and generating features on big image datasets may take days to weeks. In this paper, we describe a data generator for evaluating EMDSA techniques. Our data generator is *generic* in the following ways.

1. Since different applications require different features, the generator supports a wide range of content-based image features ranging from the standard MPEG-7 descriptors to some of the recent ones published in the prestigious venues.

2. Since scaling to large image datasets is vital, the generator can run on either a local machine or a Hadoop cluster. When running on a Hadoop cluster, it employs the MapReduce paradigm to benefit from its horizontal scalability via parallel processing.

3. Since image files may reside on multiple locations in practical applications, the generator can process images stored on a local file system, the Hadoop file system (HDFS), or the Internet (where a URL list is provided).

Different from the existing large-scale image feature generators, our feature generator is specifically designed for EMDSA. It considers the spatial correlations between contents in an image by partitioning the image into sub-images and encoding the locations of the sub-images in the “ground distance” of EMD computation while the existing generators process each image as a whole. The generator will help researchers focus on the research problems in EMDSA rather than the tedious and time-consuming data preparation work. We have made the source code of our feature generator and a number of pre-generated feature datasets with typical parameter settings publicly available online\(^1\).

The remainder of this paper is as follows. Section 2 and Section 3 present the design and implementation of the generator, respectively. Section 4 describes the provided feature datasets and the usage of the generator. Section 5 discusses the literature of image feature generators and Section 6 concludes the paper.

## 2 Feature Generator Design

### 2.1 Earth Mover’s Distance

Earth Mover’s Distance (a.k.a. Wasserstein metric) between two data objects is defined as the minimum transformation cost from one to the other. A data object is represented as a histogram with \(n\) bins, each of which associates a multi-dimensional vector \(l\), representing the spatial location of the bin, with a nonnegative value \(w\), representing the feature value of the bin. To transform one histogram to the other histogram is to move the values between bins so that the two histograms have the same values in the corresponding bins (assuming both histograms are normalized such that \(\sum w = 1\)). The cost of the transformation is the sum of the distance between the vector \(l\) of bins times the amount of value moved between bins. Here, the distance between two vectors \(l\) is called the *ground distance* (or cost matrix). Formally,

\[
EMD(h_\alpha, h_\beta) = \min \sum_i \sum_j f_{i,j} d_{i,j}
\]

s.t. \(\forall i: \sum_j f_{i,j} = w_i; \forall j: \sum_i f_{i,j} = w_j; \forall i, j: f_{i,j} \geq 0,\)

where \(d_{i,j}\) and \(f_{i,j}\) are the ground distance and the amount of values transformed between the \(i^{th}\) and the \(j^{th}\) bins, respectively.

The superiority of EMD to the \(\ell_p\) distances in pattern similarity is demonstrated in the example of Figure 2, where three data objects are represented as histograms and compared. Visually, A is more similar to B than to C since B has a more uniform distribution. If the \(\ell_2\) distance is used, both B and C are equally similar to A as they have the same distance to A. If EMD is used, the distance between A and B is smaller than the distance between A and C, congruent with the visual perception.

\(^1\)http://spatialanalytics.cis.unimelb.edu.au#generator
Figure 2: Example on comparing three 5-bin one-dimensional feature histogram; visually, A is more similar to B than to C: $\ell^2(A, B) = \ell^2(A, C) = 2.83$, $\text{EMD}_\ell(A, B) = 1 \times 1 + 1 \times 1 + 1 \times 1 = 3$, and $\text{EMD}_\ell(A, C) = 2 \times 4 = 8$.

Figure 3: Extracting the features from sub-images and generating a 16-bin histograms using the 1st feature value

2.2 Feature Extraction for EMD

A straightforward way to capture the spatial correlations between pixels is to treat each pixel as a single bin where the vector $l$ represents the spatial location of the pixel and the value $w$ represents the color value of that pixel. However, extracting features in such a way has two major drawbacks for EMDSA. First, most previously proposed content-based image features such as textures and shapes are defined on an image rather than on a pixel. There is only one piece of information about a pixel, i.e., its color value, and it has no other content-based features. Second, the one-bin-per-pixel approach will produce an enormous number of bins. For example, an 800 by 600 resolution image produces 480,000 bins. Since the cost of a single EMD computation is super cubic to the number of bins, the one-bin-per-pixel approach is prohibitive in terms of the computation cost.

Instead, following the previous practice [22, 16], we partition an image into multiple disjoint sub-images and perceive each sub-image as a bin and the whole image as a histogram. The content-based feature values extracted for each sub-image then serve as the bin value $w$ while the sub-image location serves as the bin vector $l$.

However, each bin of a histogram has only a one-dimensional value $w$, which cannot capture the multi-dimensional feature values commonly seen in content-based image features. Therefore, usually multiple histograms are generated for each image where each histogram corresponds to one dimension of the multi-dimensional image features chosen. The generator provides a parameter for users to specify which dimension to use in generating the feature histograms.

The partitioning of images is based on a two-dimensional grid. The granularity of the grid is configurable by a parameter. Such partitioning results to a two-dimensional histogram structure and naturally suggests $\ell_p$ distances to be used as the ground distance. As the generator partitions all the images using the same
grid, the bin vectors are shared by all the histograms. The generator will output a file listing the shared bin vectors so that EMD can be computed accordingly. Figure 3 shows an example of the feature extraction and the composite histogram generation using a $4 \times 4$ grid and the first dimension of the feature value. The generator will additionally perform a normalization step on the generated histograms to make sure all histograms have the same sum of values in all the bins.

3 Implementation

The generator is implemented in Java and built upon the Hadoop library. This section describes the features supported, generation modes, and the image source supported.

3.1 Supported Features

The following content-based image features are supported by the generator.

- **Auto Color Correlogram (ACC)** [10]: a color based histogram that incorporates the spatial correlations to tolerate large changes in both appearance and shape.
- **Color and Edge Directivity Descriptor (CEDD)** [3]: a low level feature that incorporates both color and texture information in one histogram.
- **Simple Color Histogram (CH)**: a RGB space histogram using Jensen-Shannon divergence as the distance function.
- **MPEG-7 Color Layout Descriptor (CL)**: a 33D histogram descriptor that captures the spatial distribution of image color using DTC representation.
- **MPEG-7 Edge Histogram (EH)**: a histogram descriptor capturing four directional and one non-directional texture edge features.
- **Fuzzy Color and Texture Histogram (FCTH)** [4]: a histogram that combines three fuzzy systems to capture the low-level color and texture information.
- **Gabor Texture (GABOR)**: a homogeneous texture histogram descriptor, also used as the MPEG-7 Homogeneous Texture descriptor.
- **Hashing Color and Edge Directivity Descriptor (HCEDD)** [3]: a histogram similar to the CEDD with BitSampling based hashes in an additional index field for sub linear search.
- **Joint CEDD and FCTH Histogram (JCD)**: a histogram that combines the features from the CEDD and the FCTH histograms.
- **Joint Histogram (JH)**: a histogram that extracts local pixel feature combinations to differ images with the same color histograms.
- **JPEG Coefficient Histogram (JCH)** [5]: a histograms that stores the DCT coefficients used in compressing the images.
- **Luminance Layout (LL)**: a histogram designed for grey-scale images which scales the original images to smaller ones and uses the smaller versions as the descriptors.
- **Opponent Histogram (OH)** [20]: a histogram that combines RGB histograms based on the channels of the opponent color spaces.
- **Spatial Pyramid Kernel (PHOG)** [2]: a histogram which incorporates the local image shape and its spatial layout information.
- **MPEG-7 Scalable Color Descriptor (SC)**: a HSV color space histogram encoded by Haar transform.
- **Tamura Texture (TAMURA)** [19]: a histogram that captures the texture features in the image.

The generator employs the LIRE library [14] to extract the above content-based image features.

3.2 Generation Mode

The generator can run either on a local machine or on a Hadoop cluster.

- **Local mode.** This mode is for testing and tuning the parameters such as grid granularity and the feature value to be used.

2http://mpeg.chiariglione.org/standards/mpeg-7/visual
• **MapReduce mode.** This mode runs the feature extraction and histogram generation in a parallel manner on a Hadoop cluster using the MapReduce paradigm.

### 3.2.1 Local Mode

The generator iterates all images, partitions each image using a grid, extracts the selected image features for sub-images, and composes the feature histograms according to the parameters chosen.

### 3.2.2 MapReduce Mode

We run it on MapReduce to reduce running time and also because similarity analysis on big image data is likely to be run on a distributed file system such as the Hadoop file system. The MapReduce paradigm is designed to simplify parallel programming on a cluster of commodity machines yet its original targeted application is text-oriented processing. Moreover, the persistent layer in Hadoop, i.e., HDFS, is designed for large files and its performance degrades drastically when dealing with a large amount of small files. To overcome this issue, the generator employs *Hadoop Image Processing Interface (HIPI)*\(^3\) to concatenate image files into large bundles, i.e., HIPI image bundles. Each HIPI image bundle will span multiple HDFS blocks, for each of which one map task will be created. In the map task, the image EXIF meta data will serve as the *map-key* while the image data will serve as the *map-value*. The generator then extracts the feature values and composite the feature histograms in a similar way to that of the local mode.

The *reduce-key* is the feature name, i.e., ACC, SC, etc., so that all histograms containing the same features are shuffled to the same reducer. The reducers simply concatenate all the histograms for the same features into a file and output the file to HDFS.

However, as HIPI does not provide an interface to incorporate meta-data other than EXIF into the bundle, an additional step is carried out to retain the association between the image ID and the image data objects. Specifically, the generator uses MD5 hash values of the image data as their fingerprints and output the relationships between the IDs and fingerprints for all the images using an extra MapReduce job before generating the features.

### 3.3 Image Source

The generator can process images from three types of sources when running in the MapReduce mode.

- **Local file system.** If all the images are stored in a single machine that can access the Hadoop cluster, we run the generator on this machine. The generator concatenates all the images into HIPI image bundles, and uploads the bundles to HDFS. Then these bundles can be accessed from the Map Reduce jobs in the generator.
- **HDFS.** There are two scenarios as the images can be either in the form of individual images or in the form HIPI image bundles. If the individual file form is the case, the generator will again concatenate the images to HIPI bundles. If the images are already in HIPI image bundles, the generator will use them directly. Either way, the images are accessed from the HIPI bundles in the MapReduce jobs.
- **HTTP URLs.** The input is a list of HTTP URLs to the images files and the generator will fetch the image, generate the histograms, and only output the generated data. No HIPI image bundles are created as the images are fetched on the fly from the Internet.

It is worth noting that if the images are obtained from HTTP URLs, the input list is likely to be much smaller than the image dataset. Hence only a small number of map tasks will be scheduled if we use the map phase to perform crawling and generating. To fully employ the distributed computation power, we instead use the map phase to shuffle the input URLs in a random fashion, and utilize the reduce phase to perform the crawling and generating.
### Table 1: Pre-generated Datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Features</th>
<th>Bin #</th>
<th>Image #</th>
</tr>
</thead>
<tbody>
<tr>
<td>M16</td>
<td>ACC, CEDD, CH, CL, EH, FCTH, HCEDD, JCD, JH,</td>
<td>16</td>
<td>1,000,000</td>
</tr>
<tr>
<td></td>
<td>GABOR, JCH, LL, PHOG, TAMURA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M64</td>
<td>ACC, CEDD, CH, CL, EH, FCTH, HCEDD, JCD, JH</td>
<td>64</td>
<td>1,000,000</td>
</tr>
<tr>
<td>M256</td>
<td>ACC, CEDD, CH, CL, EH, FCTH, HCEDD, JCD, JH</td>
<td>256</td>
<td>1,000,000</td>
</tr>
<tr>
<td>F16</td>
<td>ACC, CEDD, CH, CL, EH, FCTH, HCEDD, JCD, JH</td>
<td>16</td>
<td>2,994,379</td>
</tr>
<tr>
<td>I16</td>
<td>CEDD, CH, CL, EH, FCTH, HCEDD, JCD, JH</td>
<td>16</td>
<td>8,765,995</td>
</tr>
</tbody>
</table>

### Table 2: Generator Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>generate.mode</td>
<td>string</td>
<td>'local', 'mr'</td>
</tr>
<tr>
<td>generate.mr.source</td>
<td>string</td>
<td>local, 'hdfs', 'HTTP'</td>
</tr>
<tr>
<td>generate.hdfs.name</td>
<td>string</td>
<td>HDFS namenode</td>
</tr>
<tr>
<td>generate.input.image.dir</td>
<td>string</td>
<td>local or HDFS path</td>
</tr>
<tr>
<td>generate.output.hist.dir</td>
<td>string</td>
<td>local or HDFS path</td>
</tr>
<tr>
<td>generate.enabled.features</td>
<td>string</td>
<td>'acc', 'cedd', 'ch', 'cl', 'eh', 'fcth', 'gabor', 'hcedd', 'jcd', 'jh', 'jch', 'll', 'oh', 'phog', 'sc', 'tamura'</td>
</tr>
<tr>
<td>generate.grid.granularity</td>
<td>integer</td>
<td>not smaller than 2</td>
</tr>
<tr>
<td>generate.feature.value</td>
<td>integer</td>
<td>smaller than chosen feature dimension</td>
</tr>
</tbody>
</table>

## 4 Usage

### 4.1 Pre-generated Datasets

We provide dozens of pre-generated datasets from the MIRFLICKR [13], the Flickr3.5 [12], and the ImageNet [7] image collections with typical parameter settings (the first dimension feature value is used). These datasets should suit the needs of common evaluation studies of EMDSA techniques. Table 1 lists the summary of these datasets.

### 4.2 Generator Options

To use the generator for extracting feature histograms from image datasets, the configuration file should be edited accordingly and passed as an argument of the generator. The configurable parameters are listed in Table 2.

To use the generator on arbitrary image datasets, first choose a mode for the `generate.mode` parameter according to the workloads and the location of the image datasets. If the ‘mr’ mode is chosen, the image source should be specified in the parameter `generate.mr.source` and the HDFS namenode address should be filled to the parameter `generate.hdfs.name`. Next, both the input and the output paths are required. They can be either local paths or HDFS paths, depending on the value of `generate.mode`. The supported features can be enabled by adding them to the string list separated by ‘/’. The grid granularity defines the bin vectors for the generated histograms, and the feature value defines which dimension of the content-based feature is used to generate the histogram. Finally, the generator can be run with the commands `java -jar` or `hadoop jar`.

---

3http://hipi.cs.virginia.edu/
<table>
<thead>
<tr>
<th>Feature</th>
<th>Dimension</th>
<th>Elapsed Time (ms) for Generating n-bin Feature Histogram from One Image</th>
<th>1M Images (hours)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>n=10</td>
<td>n=64</td>
<td>n=256</td>
</tr>
<tr>
<td>ACC</td>
<td>1024</td>
<td>87</td>
<td>91</td>
</tr>
<tr>
<td>CEDD</td>
<td>144</td>
<td>40</td>
<td>39</td>
</tr>
<tr>
<td>CH</td>
<td>64</td>
<td>19</td>
<td>19</td>
</tr>
<tr>
<td>CL</td>
<td>33</td>
<td>41</td>
<td>40</td>
</tr>
<tr>
<td>EH</td>
<td>80</td>
<td>29</td>
<td>29</td>
</tr>
<tr>
<td>FCTH</td>
<td>192</td>
<td>42</td>
<td>41</td>
</tr>
<tr>
<td>GABOR</td>
<td>60</td>
<td>307</td>
<td>4,788</td>
</tr>
<tr>
<td>HCEDD</td>
<td>144</td>
<td>54</td>
<td>68</td>
</tr>
<tr>
<td>JCD</td>
<td>168</td>
<td>78</td>
<td>104</td>
</tr>
<tr>
<td>JH</td>
<td>576</td>
<td>18</td>
<td>18</td>
</tr>
<tr>
<td>JCH</td>
<td>192</td>
<td>239</td>
<td>237</td>
</tr>
<tr>
<td>LL</td>
<td>64</td>
<td>166</td>
<td>633</td>
</tr>
<tr>
<td>OH</td>
<td>64</td>
<td>25</td>
<td>24</td>
</tr>
<tr>
<td>PHOG</td>
<td>630</td>
<td>158</td>
<td>169</td>
</tr>
<tr>
<td>SC</td>
<td>64</td>
<td>17</td>
<td>18</td>
</tr>
<tr>
<td>TAMURA</td>
<td>18</td>
<td>1,795</td>
<td>7,149</td>
</tr>
</tbody>
</table>

### 4.2.1 Generation Time of the Supported Features

The experiment is conducted on a single computer with a CPU 3.2GHz and 6GB main memory. The computer runs CentOS 6.5 with Linux kernel 2.6.32 and JDK 1.7.0 update 17. The reported results are averaged on processing 10,000 random images. Table 3 summarizes the average elapsed time for the generator to extract histograms from one image, where the last column estimates the time for generating 1 million histograms on 500x500 images.

### 5 Related Work

#### 5.1 EMD Based Data Analysis

Computing EMD can be transformed to solving a transship problem and the simplex solution for the problem results to an $O(n^3 \log n)$ computation cost. Much efforts therefore have been devoted in designing efficient index structures and algorithms to speed up approximating [18, 15, 6], retrieving [1, 21, 22, 16], and computing EMD. As large scale data emerge, computing EMD on large scale data is necessary for exploiting its unique power on distinguishing complex data objects. The task is challenging in terms of efficiency as it involves potentially a large number of EMD computations. The framework MELODY-JOIN is proposed to employ the MapReduce paradigm to efficiently process large-scale EMD based similarity join on distributed computers [9]. Recently, it is reported that a better algorithm [11] can largely improve the efficiency of solving the max-flow problem. Since the optimization in computing EMD can be solved using max-flow algorithms, this new technique may have the potential to speedup large-scale EMD computations as well.

#### 5.2 Large Scale Feature Datasets

There are several image and feature datasets made for the training phase of machine learning techniques applied in computer vision contexts. ImageNet [7] is a collaborating image datasets with the mapping information between the images and the WordNet tags. ImageNet provides the URLs of images to the public. CloudCV\(^4\) provides 16 types of popular scene descriptors extracted from 1.2 million images in the ImageNet dataset which are primarily designed for object detection. Pattern Analysis, Statistical Modeling, and Computational Learning Visual Object Classes Challenge (PASCAL VOC)\(^5\) also provides several image datasets along its 8 years’ challenges. The original images of PASCAL VOC are available. Flickr 3.5M [12] is another dataset made for tag relevance learning task and is publicly available as URL lists. Our generator

\(^4\)http://cloudcv.org/objdetect/
\(^5\)http://pascal.in.ecs.soton.ac.uk/challenges/VOC/
is the first publicly available one that can generate EMD analysis ready data from regular image datasets including the above resources.

6 Conclusion

We designed and implemented a data generator for large scale EMDSA on image datasets and made it publicly available. The generator supports a wide range of content-based image features and can run in either local or Hadoop distributed cluster modes. It is expected that the generator will be highly useful to the future work in the fields of the large-scale EMDSA.

Acknowledgment

The work is supported by the Australian Research Council (ARC) Discovery Project DP130104587. Dr. Rui Zhang is supported by the ARC Future Fellowships Project FT120100832.

References