

Image Retrieval: Feature Primitives, Feature Representation, and Relevance Feedback

Xiang Sean Zhou, Thomas S. Huang

Beckman Institute for Advanced Science and Technology

University of Illinois at Urbana Champaign, Urbana, IL 61801, USA

{xzhou2, huang}@ifp.uiuc.edu

Abstract

In this paper we review the feature selection and representation techniques in CBIR systems, and propose a unified feature representation paradigm. We revise our previously proposed water-filling edge features with newly proposed primitives and present them using this unified feature formation paradigm. Multi-scale feature formation is proposed to support cross-resolution image matching. Sub-image feature extraction is applied for regional matching. Relevance feedback as an on-line learning mechanism is adopted for feature and tile selection and weighting during the retrieval. We discuss in detail the revised water-filling edge features, cross-scale feature extraction and image matching, and relevance feedback on regional/tile-based matching.

1. Introduction

The performance of a CBIR system is inherently constrained by the features adopted to represent the images in the database. The most frequently referred “visual contents” are color, texture, and shape[2]. If we regard information embedded in a digital image as *chrominance* information combined with *illuminance* information, then color feature captures the *chrominance* information; and both texture and shape represent the *illuminance* information. Texture features (e.g. co-occurrence features[4] and Wavelet based features[8]) and shape features (e.g. Fourier descriptors[15] and moment invariants[5]) have been applied extensively in image retrieval systems. Even though lack a formal definition, “texture features” can be described as the features that captures spatial distribution of illuminance variations in terms of “repeating patterns”. Whereas shape represents a specific edge feature that is related to object contour. However, in most real-world applications, shape features are not applicable since a meaningful segmentation is unachievable. People also tried various other features such as edge densities, edge directions, turning angles, colinearity, salient points, etc., in general or domain-specific applications. Even though there have been efforts to classify these features into the “shape” category, it is obvious that this is inappropriate—e.g., it is hard to relate the concept of “edge density” to the concept of “shape”. So we would rather regard “shape” as a special feature component of a much broader category of “structural

features”, which captures information represented in non-repeating illuminance patterns, or more specifically, edge patterns in general[16]. Under this categorization scheme, one can think of “structural features” as the features capturing the edge information in the image, while “shape” is just a special case only capturing the (outer?) edge of some object(s) in the image. As we try to answer the question of “how to represent information embedded in the edges” instead of “how to represent information of the object shape”, we are left with much more freedom and flexibility in terms of feature formation. In this paper we try to use a unified feature representation paradigm to illustrate the process of feature formation for images and apply it to guide the feature formation of our proposed water-filling features. The detailed discussion is presented in Section 2 and 3. As we include cross-scale feature extraction into the general feature formation paradigm, cross-scale image matching becomes possible. Section 4 presents the results on cross-scale matching. If we allow either rough region segmentation or straightforward tiling on the image, regional/tile-based features can be formed together with global image features, and an on-line learning scheme based on the user relevance feedback can be applied to automatically determine the relative importance of the tile-based features versus the global features. An experiment is described in Section 5.

2. Unified Feature Representation Paradigm

Most of the existing image features can be regarded as constructed in the following two steps: 1. Feature primitives are selected based on the original image, which will retain useful information embedded in the image. Certain transformation can be applied prior to the primitive extraction; 2. A compact representation is chosen to capture information carried by the feature primitives. Most of these representations are of statistical nature. Table I shows some examples of feature formations, including color moments, color histogram, wavelet moments for texture, co-occurrence matrix for texture, and water-filling features for structure. Initially, information is carried in the pixel values—both color and intensity. These can serve as feature primitives themselves. Or feature primitive can be the output of some lossless or lossy transformation, such as DCT, wavelet Transform, or edge detection followed by water-filling operation.

Table I. Feature formation: from feature primitives to feature representation
(*: based on transformed feature primitives)

Primitives/ Transformations		Pixel color {H,S}		Pixel intensity {V}	
		<i>DCT*</i>		<i>Wavelet transform*</i> => <i>Wavelet coef.</i>	<i>Edge detection + Water-filling*</i> => <i>FillingTime, ForkCount, etc.</i>
Statistics	<i>Independent</i>	Color-moments,		Wavelet moments*[8]	Max*, sum*, histogram(1-D)*[16]
	<i>Dependent</i>	Color histogram(2-D), Mixture models*[14]		Salient points*	"ForkCount associated w/ the max FillingTime", etc.*[16]
Spatial relation		Color-correlogram[6]		Spatial co-occurrence[4]	Edge densities
Scale/Resolution		(Resolution invariant?)		(Intrinsic)	Multiscale edge detection[13]
Regions/tiles		Feature representation within image regions/tiles			

Based on the feature primitives such as pixel {hue, saturation} values, pixel intensity values, DCT coefficients[14], wavelet coefficients, or water-filling primitives (see Section 3.), the most common representation is to use the statistics. In case the feature primitives are multidimensional vectors, e.g., DCT coefficients for 8X8 blocks, we can either assume that the components are independent thus use only moments, maximum/minimum, or 1-D histogram of the individual component to represent the information; or assume that the components are dependent, thus use multidimensional histogram. Vasconcelos and Lippman argue that mixture modeling is a better alternative due to intractable computation induced by high dimensional histograms[14]. Wavelet based salient points use the dependency (correlation) among coefficients from different bands to establish saliency for a given point. The water-filling features such as "the ForkCount associated with the maximal FillingTime" captures certain dependency between the feature primitive FillingTime and ForkCount (see Section 3). Neither histogram nor mixture modeling captures spatial relations among feature primitives. The early work that emphasized spatial relations is co-occurrence matrices and the derived texture features[4]. Later work by Huang et al. extended the idea into color histogram, by adding spatial relations among pixel/blocks to form color correlogram[6]. Edge density gives information in spatial distribution of edges (See Table I).

To support cross-scale image matching, resizing all the images to the minimal size for feature extraction can be one solution, but definitely not an ideal one. For dynamic databases or integrated databases in which resizing the images are formidable in terms of computational efficiency, direct matching of images with different sizes are desired. Alternative solutions could be either using resolution-invariant features, or feature extraction at different scale for resolution-variant features. Some color feature such as color histogram can be regarded as resolution invariant to certain degree. But with large variation in scale, or for features like color correlogram, invariance will not sustain and feature extraction under different scales may be necessary for cross-scale image matching. Wavelet transformation intrinsically captures the information cross resolutions, therefore can directly support cross-scale image matching. However, edge

detectors are usually not scale invariant, e.g., at a small scale with low resolution, the edge detector can fail to extract the details in the structure where it could at a higher resolution. A multiscale edge/region detector[13] can be applied to form edge maps at multiple scales for a given image. One feature vector is extracted from each scale. The matching is then based on certain "compound distance" from cross-scale matching (See Table I).

To support region/tile based local matching versus global matching, features should be constructed from image regions/tiles separately. Regional/tile-based feature vectors can be concatenated together with the global feature to form the final feature vector. Relevance feedback technique can be applied to dynamically adjust the relative importance of these features for a given retrieval task.

3. Water-Filling Feature Formation

In this section we discuss in detail the water-filling algorithm and the water-filling feature formation. This is an extension of the work in [16] with additional features such as LoopCount proposed and analyzed. The water-filling algorithm operates on edge maps to extract edge features. The algorithm also assumes that thinning operation has been performed on the edge map so that all the edges are one pixel wide. In this paper, we use 4-connectivity for simplicity. The feature representation largely follows the above proposed paradigm.

To illustrate the algorithm, let's first consider the simple case of an 8 by 8 edge map with all the edge pixels connected (Fig. 1: shaded pixels are edge pixels). The algorithm will do a first raster scan on the edge map and start a traverse (think it as filling in water) at the first edge pixel encountered that has less than 2 neighbors, i.e., start at an end point. In Fig.1 the pixel with label "1" is the first end point encountered. (In case all edge pixels have at least two neighbors, i.e., no end points but all loops, e.g., Fig. 3, then start at the first unfilled edge pixel during the second scan. So it is necessary to have two raster scans to avoid possible miss). The waterfront then flows along the edges in the order indicated by the numbers. Note that when there are more than one possible paths to go at one point, the waterfront will fork (e.g., at pixel "6" and "9" in Fig. 1).

One can see that this algorithm can be regarded as a simulation of “flooding of connected canal systems (i.e., connected edges)”, hence the name “water-filling algorithm”. The assumptions implied by the algorithm include: i) we have unlimited water supply at the starting pixel; ii) water flows at a constant speed in all directions; iii) water front will stop only at a dead-end, or at a point where all possible directions have been filled.

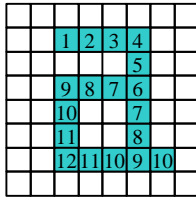


Figure 1

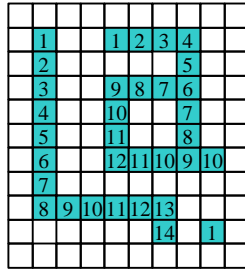


Figure 2

When there are more than one set of connected edges in the edge map (e.g., the case in Fig.2), the algorithm will fill all the sets independently in sequential or in parallel.

As water fills the canals (edges), various information are extracted, which are stored as the feature primitives. Feature vectors can then be constructed based on these feature primitives. The time complexity of this algorithm is linear, proportional to the number of edge points in the image.

3.1 Feature primitives

1). Filling time

Filling time is the time for water to fill a set of connected edges. For Fig. 1 through 3, the filling times are {12}, {14, 12, 1}, and {12}, respectively. Using different starting pixels, the filling time can vary in a range of $[t, 2t]$, where t is the minimum filling time among all possible selection of the starting pixels. This is easily proved as follows: denote the starting pixel that gives the minimum filling time t as S . Since we assume water runs in both directions on an edge, water can reach S from any other starting pixel, say P , in time t , and then the waterfront can go from S to reach any pixels left unfilled within time t . So the filling time from P is $\leq 2t$.

To minimize the variation in filling time due to selection of starting pixels, we can impose additional constraints on the selection of starting pixels (e.g., choose only the end points), or choose different starting pixels and average the results.

To achieve scaling invariance, normalize the filling time according to the image size. For example, divide filling time by (width + height).

2). Fork count

Fork count is the total number of branches the waterfront has forked during the filling of a set of edges. If we consider the initial waterfront as one branch to start with, then the fork count for Fig. 1 through 3 are {3}, {1, 3, 1}, and {6}, respectively. If we choose an end pixel as

starting pixel whenever possible, fork count is “almost invariant” to starting pixel selection. The variation is within ± 1 , depend upon whether the water starts from the middle of an edge or the end of the edge. Also if multiple waterfronts collide at one intersection, even though the water does not actually fork, the potential forks should be counted to achieve the “almost invariance”. E.g., an extra fork is counted both at the upper “9” and the lower “10” in Fig. 3; but none at “12”, since it is not a potential fork point in any way).

3). Loop count

Loop count is the number of simple loops (or, “simple cycles” as defined in Corman et al., 1997, p. 88) in a set of connected edges. For example, in Fig.1 through 3, the loop counts are {1}, {0, 1, 0}, and {3}, respectively. Loop count is invariant to rotation.

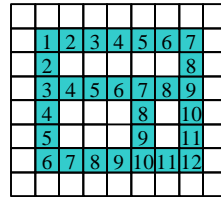


Figure 3

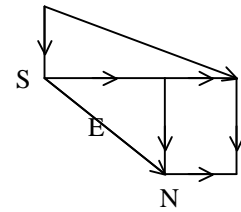


Figure 4

To get the loop count during the water-filling process, we make use of the following “Theorem of Splashes”:

If we assume that when two waterfronts collide, we see one “splash”; or more generally, when n waterfronts collide at one intersection, we see $n-1$ “splashes” (think it as $n-1$ waterfronts collide with the first waterfront sequentially).

Then the number of splashes = the number of simple loops.

For example, in Fig. 3, three splashes are recorded at time “9”, “10”, and “12”. Hence the loop count is 3.

The above theorem provides a way of recording loop counts within the water-filling process with very little overhead computation.

4). Water amount

Water amount is the total amount of water used to fill up the set of edges in terms of number of pixels. So it is the edge pixel count. In Fig. 1 through 3, the water amounts are {18}, {14, 18, 1}, and {29}, respectively.

Note that there exist many other possibilities on selecting the feature primitives, such as horizontal (vertical) cover, etc. However, the final selection should depend upon the specific application, i.e., what information is important and most discriminative toward the classification.

3.2 Edge/structural feature representation

Based on the feature primitives in 4.1, we can then construct edge/structural features from their statistics. For example: moments (e.g., average filling time); order statistics (e.g., maximum loop count); distributions (e.g.,

water amount histogram); etc. In the following we discuss some examples with the emphasis on their meanings from a human perception point of view.

1). (MFT&FC) MaxFillingTime and the associated ForkCount

MaxFillingTime(MFC) is defined as $\max\{\text{filling times}\}$. For Fig. 1 through 3, the **MFC** is 12, 14, and 12, respectively. And the associated **ForkCount(FC)** is 3, 1, and 6 respectively. So the **MFT&FC** vectors for Fig. 1 through 3 are (12, 3), (14, 1), and (12, 6), respectively.

MFT&FC are features most probably associated with a salient object (with the “longest” edge) in the image. The **MFT** conveys a rough measure of the size (edge length) of this object, while the associated **FC** gives measure of complexity of the structure of the object (complexity of the edges).

2). (MFC&FT) MaxForkCount and the associated FillingTime

Similarly defined as **MFT&FC**, these are also features most probably associated with a salient object in the image. The **MFT** conveys a rough measure of the complexity of the object. This object may or may NOT be the same object as the previous one. For Fig. 1 and 3, the **MFC&FT** is the same as the **MFT&FC**. But for Fig. 2, the **MFC&FT** vector is (3, 12).

3). (GLC&MLC) GlobalLoopCount and MaxLoopCount

GlobalLoopCount is defined as $\text{sum}\{\text{loop counts}\}$. **MaxLoopCount** is $\max\{\text{loop counts}\}$. This feature vector can capture structural information such as the windows in the build images. Or can be used toward character detect and recognition applications.

4). (FTH&FC) FillingTime Histogram and the associated averaged ForkCount within each bin

This is a global feature on all sets of *connected* edges in the edge map. It represents the edge map by the distribution of edge “length”. Noise or changing background with short edges may only affect part of the histogram, leaving the portion depicting the salient objects unchanged. Thus by proper weighting of the components (e.g. by relevance feedback[11]), we could achieve robust retrieval.

5). (WAH) WaterAmount Histogram

This is also a global feature with multiple components. It is another measure of the distribution in edge length or density.

Note again that there can be many other possible ways to construct feature vectors according to Table I, such as the *moments of filling times, fork counts, loop counts, or water amounts, etc.* Water-filling features have been shown to capture salient structural information and can improve the retrieval performance significantly [16].

4. Cross-Scale Matching

To match images of different sizes, we tested the multiscale edge/region detector of [13], which yields

homogeneous regions surrounded by closed edge contours. It can represent image structures at different scales without scaling of the image itself. The advantage of this algorithm is that no prior knowledge of the image sizes is required, whereas if we adopt a straightforward approach of extracting in advance multiscale features by actually scaling the images, we need to decide on the sizes to use, which is tricky.

From the multiscale edge maps, one set of water-filling features is extracted from each scale. The distance measure on water-filling features between two images, *i* and *j*, is then defined as:

$$D_{ij} = \min_{s,t} (|\bar{f}_s^i - \bar{f}_t^j|^2)$$

where *s*, *t* = 1, 2, ..., *S* are the scale index and *S* is the number of scales used. The actual size of the image, if available in the metadata, can be incorporated into the distance measure or the scale selection criteria.

Table II: Average rank for the same image of smaller sizes

	½ size	¼ size
Sobel edge	2.7	5.6
Single scale	2.6	5.1
Cross-scale	2.2	3.7

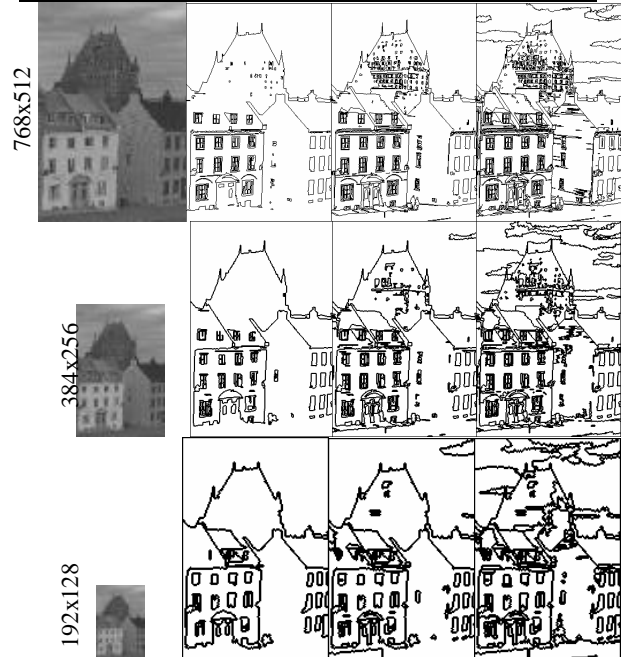


Figure 6. Multiscale edge/region detection for an image at three sizes.

An experiment is designed for cross-scale image matching using water-filling features. The Mpeg7 dataset is used, we build ground truth by actually scaling the 92 original images down by half (i.e., half in length and half in width) and ¼ and add these images into the database. For each image, the edge/region detector of [13] is applied to yield three edge maps at different structural scales (not image scales! See Fig. 6). Retrievals using 20 original images as queries are performed and the results

are compared. Table II shows the average ranks of the same image of smaller sizes (exclude the original image which always appears as the rank 1 return). The first row uses the Sobel edge detector; the second row uses one scale from the multiscale edge detector (the middle scale); the last row uses all three scales and performs cross-scale matching (e.g., in Fig. 6 the distance between the original and the $\frac{1}{4}$ sized image is the distance between the first scale edge map of the original and the second scale edge map of the $\frac{1}{4}$ sized). The improvement in performance is evident.

5. Relevance Feedback on Tile-Based Image Matching

In an interactive image retrieval system, a user can give the system feedback on which of the images returned by the system are relevant to his/her interest. Then a learning algorithm can be applied to adjust the query and the multilevel weights of the features to significantly improve the retrieval results[11]. With features extracted from image as a whole as well as from sub-images/tiles, relevance feedback can be applied to evaluate the relative importance of: 1. the tiled information versus global information; 2. the color information (in tiles or globally) versus texture or structural features; or 3. each component of the feature vectors for color, texture, or structure (See Figure 7). This scheme will improve the retrieval process where the user is actually looking at a specific region of the image instead of the whole image.

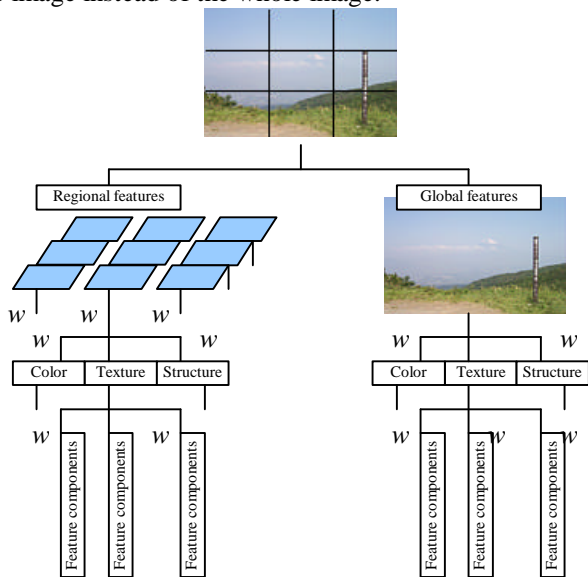


Figure 7. On-line learning for tile-based image matching, w 's are the weights adjusted by relevance feedback.

An example is shown in Figure 8, where the user is looking for the images of Mt. Fuji from a distance (thus only the tip of the mountain in the middle of the image is of interest). The left part of Figure 8 shows the result of using global features and relevance feedback, yielding only 5 images of interest. While using the tiled features

together with the global features, the number of correct returns after relevance feedback increases to 9, with improved ranking. By looking at the weights on different tiles and features, we can see that after the learning phase, the system “figured” that it is the color (“blue”) and texture (“smooth”) of the top central tile (“the sky”), and the edges (the mountain tip) in the middle of the image that are important, not the other tiles or features.



Figure 8. Example on regional/tile-based matching

Acknowledgement: This work was supported in part by National Science Foundation Grant CDA 96-24386.

Reference

- [1] Corman, T. H., C. E. Leiserson, R. L. Rivest. *Introduction to algorithms*, McGraw-Hill, New York, 1997
- [2] Flickner, M. et al., Query by image and video content: The qbic system, *IEEE Computers*, 1995
- [3] Gonzalez, R. C. and Woods, *Digital Image Processing*, Addison-Wesley, 1992
- [4] Haralick, R. M., K. Shanmugam, and I. Dinstein, Texture feature for image classification, *IEEE Trans. SMC*, Nov., 1973
- [5] Hu, M. K., Visual pattern recognition by moment invariants”, *IRE Trans. Information Theory*, 8, 1962
- [6] Huang, J., S. Kumar, M. Mitra, W. Zhu, and R. Zabih, Image indexing using color correlogram, *Proc. IEEE CVPR*, 1997
- [7] Jain, A. K., *Fundamentals of Digital Image Processing*, Prentice Hall, 1989
- [8] Laine, A., J. Fan. Texture classification by wavelet packet signatures. *IEEE Trans. Pattern Anal. Machine Intell.* 15, 1993
- [9] Persoon, E. and K. S. Fu, Shape discrimination using Fourier descriptors, *IEEE Trans. SMC*, Mar., 1977
- [10] Ratan, A. L., et. al., 1999, A framework for learning query concepts in image classification, *Proc. IEEE CVPR'99*, 423-429
- [11] Rui, Y., T. S. Huang, M. Ortega, and S. Mehrotra, Relevance Feedback: A Power Tool in Interactive Content-Based Image Retrieval”, *IEEE Tran on Circuits and Systems for Video Technology*, Vol 8, No. 5, Sept., 644-655, 1998
- [12] Smith, J. R. and Chang, Transform features for texture classification and discrimination in large image databases, *Proc. IEEE ICIP*, 1995
- [13] Tabb, M., N. Ahuja, “Multiscale Image Segmentation by Integrated Edge and Region Detection”, *IEEE trans. Image Processing*, Vol. 6, No. 5, May 1997
- [14] Vasconcelos, N. and A. Lippman, Embedded mixture modeling for efficient probabilistic content-based indexing and retrieval, in *SPIE Multimedia Storage and Archiving Systems III*, Boston, 1998
- [15] Zahn, C. T. and Roskies, Fourier descriptors for plane closed curves, *IEEE Trans. Computers*, 1972
- [16] Zhou, X. S., Y. Rui, and T. S. Huang, Water-filling: a novel way for image structural feature extraction, *Proc. ICIP*, 1999

