IWSSIP 2010 - 17th International Conference on Systems, Signals and Image Processing

Image Retrieval by Multi-Scale Interval Distance Estimation

Carlos Elias Arminio Zampieri
UNICAMP - Campinas, Brazil
carlos.zampieri@students.ic.unicamp.br

Jorge Stolfi
UNICAMP - Campinas, Brazil
stolfi@ic.unicamp.br

Abstract—We describe *multi-scale interval search* (MuSIS), a general method or for query-by-example retrieval in image collections, using interval arithmetic and special image pyramids to perform multi-scale distance estimation. The interval estimates are used to quickly eliminate candidate images by looking only at the upper levels of the pyramids, as in the branch-and-bound optimization paradigm. Experiments indicate that MuSIS can provide significant speedup relative to exhaustive search. The metdod is exact (it always returns the exact best match) and can be adapted to work with many image similarity functions.

Keywords: Content-based image retrieval; interval arithmetic; multiscale distance.

I. Introduction

Content-based image retrieval systems [3] often assume that the database is pre-processed by computing a *descriptor* for each image, which is a numerical summary of the image features that are considered relevant for searching. Since the search algorithms operate on the descriptors, the types of queries that users may pose are necessarily limited by the nature of the precomputed descriptors.

Here we describe a different approach that does not use any specialized descriptors, and relies instead on *multi-scale* or (*multi-resolution*) techniques to speed up the queries. In this method, the preprocessing phase merely creates several reduced copies of each image, at various scales. Each pixel of the reduced images consists of the average μ and deviation σ of the corresponding pixels in the original image. Using only the upper levels of the pyramids, we can quickly compute guaranteed upper and lower bounds for the distance between two images; which allow us to efficiently discard most database candidates without ever computing the corresponding exact distances.

II. Notation and definitions

In this paper we consider the following problem: given a large *image database* $B_1, B_2, \ldots B_N$, and a *query image* A, find the image B^* in the database that is closest to A, in some arbitrary metric $\mathrm{dist}(*,*)$.

We assume that each image I is a function defined on some finite rectangular *domain* $\mathcal{D} \subset \mathbb{Z} \times \mathbb{Z}$. For simplicity, we will assume that all images in the database (as well as the query image to be searched) have the same size and shape and are monochromatic; so that the value of an image I at a point p of the domain, called a *pixel* and denoted by $I[p]$, is a single real number in the interval $[0, 1]$. However, the algorithms we describe can be trivially extended to work with color images, and/or with images of different sizes.

For each image I we define an *image pyramid* $\mathbb{I} = (\mathbb{I}^{(0)}, \mathbb{I}^{(1)}, \ldots, \mathbb{I}^{(m)})$, where each $\mathbb{I}^{(k)}$ is a version of I reduced by a factor of $1/2^k$ in each direction, and therefore a factor $1/4^k$ in area. The parameter m is usually chosen so that $\mathbb{I}^{(m)}$ has a single pixel.

We will denote by $\mathcal{D}^{(k)}$ the common domain of all images reduced to scale k. Each pixel $\mathbb{I}^{(k)}[p]$ of a reduced image $\mathbb{I}^{(k)}$ is a pair of real numbers $\mu\mathbb{I}^{(k)}[p], \sigma\mathbb{I}^{(k)}[p]$ where

$$\mu\mathbb{I}^{(k)}[p] = \frac{1}{4^k} \sum_{q \in \mathcal{P}^{(k)}[p]} I[q] \tag{1}$$

$$\sigma\mathbb{I}^{(k)}[p] = \sqrt{\frac{1}{4^k} \sum_{q \in \mathcal{P}^{(k)}[p]} (I[q] - \mu\mathbb{I}^{(k)}[p])^2} \tag{2}$$

and $\mathcal{P}^{(k)}[p]$ is the set of pixels in I that correspond to pixel p of $\mathbb{I}^{(k)}$. We refer to these formulas as the *$\mu\sigma$-reduction* process.

Figure 1. A monochromatic image, and the reduced versions of the same obtained by $\mu\sigma$-reduction.

Note that the level zero version $\mathbb{I}^{(0)}$ does not have to be computed or stored, since it can be trivially recreated from the original image; namely $\mu\mathbb{I}^{(0)}[p] = I[p]$ and $\sigma\mathbb{I}^{(0)}[p] = 0$, for all p. Therefore, the pyramid of an image occupies about

$1 + 2(1/4 + 1/4^2 + \cdots + 1/4^m) < 5/3 \approx 1.67$ times as much space as the original image.

III. Multi-scale interval search

A. Description of the algorithm

Our algorithm, which we call *multi-scale interval search* (MuSIS), is shown in figure 2. It assumes that $\mu\sigma$-pyramids have been precomputed for all images B_i in the database and for the query image A. The algorithm keeps a set $\mathcal{C}$ of *candidate images* that is guaranteed to contain the correct answer — namely, the image from the database that is closest to the query image A, in the metric $\mathrm{dist}(*,*)$.

The set $\mathcal{C}$ is kept as a collection of quadruples $c = (c.B, c.\mathbb{B}, c.k, c.d)$, where $c.B$ is a handle to an image of the database, $c.\mathbb{B}$ is the precomputed pyramid of $c.B$, $c.k$ is a scale of resolution, and $c.d$ is an interval estimate for $\mathrm{dist}(A, c.B)$, computed from the reduced versions $\mathbb{A}^{(k)}$ and $c.\mathbb{B}^{(k)}$. Here and in the following, an *interval* is a pair of numbers $v = [v\!\downarrow, v\!\uparrow]$ that represents the set of all real values x such that $v\!\downarrow \le x \le v\!\uparrow$. The algorithm also maintains a global interval d^* such that $\mathrm{dist}(A, B^*) \in d^*$. This interval is the minimum of all the intervals $c.d$ in $\mathcal{C}$, that is $d^*\!\downarrow = \min\{\, e.d\!\downarrow : e \in \mathcal{C}\,\}$ and $d^*\!\uparrow = \min\{\, e.d\!\uparrow : e \in \mathcal{C}\,\}$.

1. Set $\mathcal{C}$ to contain all quadruples $(B, \mathbb{B}, m{+}1, [0\_1])$ such that B is in the database. Set $d^* \leftarrow [0\_1]$.

2. Let c be a candidate with minimum $c.d\!\downarrow$. If $\#\mathcal{C} = 1$, return $c.B$ as the answer to the query and stop. Otherwise let e be some after candidate with the second-smalest $e.d\!\downarrow$. If $c.d\!\uparrow \le e.d\!\downarrow$, then return $c.B$ as the answer to the query and stop.

3. Select some candidate c from $\mathcal{C}$. Compute a new interval estimate d' for $\mathrm{dist}(A, c.B)$, using the reduced images $\mathbb{A}^{(c.k-1)}$ and $c.\mathbb{B}^{(c.k-1)}$. Set $d' \leftarrow d' \cap c.d$. Replace the candidate c in the queue by $c' = (c.B, c.\mathbb{B}, c.k{-}1, d')$, and update d^* accordingly.

4. Remove from $\mathcal{C}$ every candidate c which has $c.d\!\downarrow > d^*\!\uparrow$. Repeat from step **2**.

Figure 2. The MuSIS algorithm.

The general situation during the MuSIS search is illustrated in figure 3 (top). Each error bar indicates the interval estimate $c.d$ for some candidate c in $\mathcal{C}$. The dashed horizontal lines indicate the interval d^*, which here coincides with the interval $c.d$ of candidate 0 At each iteration in step **3**, we refine the estimate of $\mathrm{dist}(A, c.B)$ of a candidate c from the list using the versions $\mathbb{A}^{(k-1)}, c.\mathbb{B}^{(k-1)}$ at the next finer scale. We then update d^*, and that may allow us to eliminate some candidates from the list.

Figure 3 (bottom) shows the outcome of such an event: after candidate 0 was re-evaluated, the interval d^* came to be defined in step **3** by the low end of the new candidate 0 and the high end of the new candidate c. With the lower value of $d^*\!\downarrow$, it was possible to discard more than half of the candidates from the queue—without ever computing their exact distances from A.

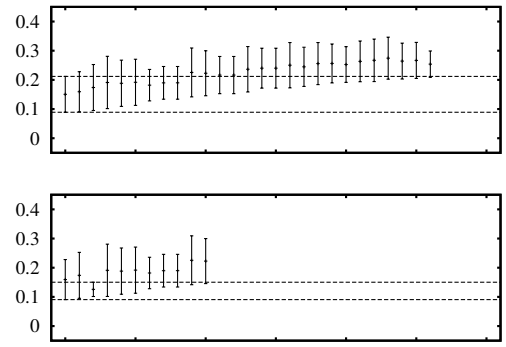

Figure 3. Two successive iterations of the algorithm.

The new interval d' computed in step **3** should ideally be a sub-interval of $c.d$. However, depending on how dist is computed, this may not be always true in practice. In any case, if both $c.d$ and d' are correct interval enclosures for the exact distance $\mathrm{dist}(A, B)$, the same is true of their intersection. The command $d' \leftarrow d' \cap d$ is therefore harmless, and it has the merit of ensuring that $d'\!\uparrow \le c.d\!\uparrow$ and $d'\!\downarrow \ge c.d\!\downarrow$. Therefore, to update the high end $d^*\!\uparrow$ it suffices to do $d^*\!\uparrow \leftarrow \min\{d^*\!\uparrow, d'\!\uparrow\}$. To update the low end $d^*\!\downarrow$ efficiently, on the ofter hand, we need to keep the candidate tuples in a heap data structure, sorted by $d\!\downarrow$, with the minimum at the root.

B. Correctness

The main loop of the algorithm preserves the following invariants (a): there is a candidate c in $\mathcal{C}$ such that $c.B$ is the correct answer B^* (the image from the database for which $\mathrm{dist}(A, c.B)$ is minimum); and (b) the exact distance $\mathrm{dist}(A, c.B)$ lies in the interval $c.d$, for every candidate c in $\mathcal{C}$. These invariants are obviously true at the beginning, and step **4** only eliminates a candidate c if $\mathrm{dist}(A, c.B)$ is guaranteed to be larger than $\mathrm{dist}(A, e.B)$ where e is the candidate in $\mathcal{C}$ that defined the value of $d^*\!\uparrow$.

Moreover, at every step the algorithm either eliminates one or more candidates, or decrements the $c.k$ field of some candidate. In a candidate c with $c.k = 0$, the interval $c.d$ must be a singleton (that is, we must have $c.d\!\downarrow = c.d\!\uparrow$). Therefore, in step **2**, if all the candidates of $\mathcal{C}$ have $c.k = 0$, then we must have $c.d\!\downarrow = c.d\!\uparrow = e.d\!\downarrow = e.d\!\uparrow$, so the algorithm will stop. The termination and correctness of the algorithm then follows.

C. Efficiency

The worst case for this algorithm is when no candidates are eliminated in step **4**, and the iteration continues until

all candidates in the queue have $c.k = 0$. Only at that point will the algorithm stop, because of the second test of step **2**.

Assuming that the interval-valued version of dist is at most δ times more expensive than the single scale version, and that the cost of computing $\mathrm{dist}(A, B)$ is approximately αn for images with n pixels, then the worst-case cost — computing $\mathrm{dist}_2^{(k)}$ at all scales, for all N database images — will be $N\alpha n(1 + \delta/4 + \delta/4^2 + \cdots + \delta/4^m)$, which is less than $N\alpha n(1 + \delta/3)$. In comparison, the brute-force algorithm has cost $N\alpha n$. Therefore, the worst-case cost of MuSIS is only $1 + \delta/3$ times the cost of the brute-force algorithm.

However, each tuple c with $k > 0$ that can be eliminated in step **4** will cost only $\delta\alpha n(1/4^k+1/4^{k+1}+\cdots+1/4^m) \leq (\delta/3/4^{k-1})\alpha n$ operations. This is $\delta/3/4^{k-1}$ times the brute-force cost, and this factor is usually less than 1 when $k \geq 2$. Therefore, if enough quadruples get eliminated when they have large k values, the savings will offset the overhead.

D. Which candidate to recompute

In step **3**, we use the following heuristic to choose the candidate c that is to be refined: let c and e be the first two candidates in order of increasing $d{\downarrow}$, breaking ties by decreasing k. If $c.k \neq e.k$, select the one with *largest* k. If $c.k = e.k$, select the one with smallest $d{\downarrow}$.

Note that one cannot have $c.k = e.k = 0$ at this time, since the two intervals would be singletons, in which case the smallest of the two should have been excluded in step **4**.

IV. Image distance

The correctness of the MuSIS algorithm is independent of the image distance function dist. In fact, dist does not have to be a metric in the mathematical sense of the term; and any metric $dist'$ that is a monotonic function of dist will result in the same output.

In some of our tests, we used the most basic metric for evaluating the discrepancy between two images, namely the *normalized Euclidean distance*, defined by

$$\mathrm{dist}_2(A, B) = \frac{1}{\#\,\mathcal{D}}\sqrt{\sum_{p\in\mathcal{D}} |A[p] - B[p]|^2} \qquad (3)$$

For efficiency, it is more convenient to work with the square of this distance, $\mathrm{dist}_2^2(A, B)$. Note that, $\mathrm{dist}_2(A, B) = \mathrm{rms}(A - B)$, and $\mathrm{dist}_2^2(A, B) = \mathrm{msq}(A - B)$, where

$$\mathrm{msq}(I) = \frac{1}{\#\,\mathcal{D}}\sum_{p\in\mathcal{D}} (I[p])^2, \quad \mathrm{rms}(I) = \sqrt{\mathrm{msq}(I)} \qquad (4)$$

If the pixel values are real numbers in $[0, 1]$, the value of $\mathrm{dist}_2(A, B)$ (or $\mathrm{dist}_2^2(A, B)$) is also in $[0, 1]$; and it is zero if and only if the images are identical at every pixel p.

As can be seen in figure 4, the Euclidean distance may be too simple for pratical applications. However, it can be used also after a suitable preprocessing of the images (e.g. replacing each image by its absolute gradient, in order to reduce the effect of lighting variations).

query 0.238870 0.258277 0.263883

Figure 4. A query image and the three best matches in our test database, according to the plain Euclidean metric dist_2.

A. Estimating distance from mean and deviation

For many kinds of images, such as outdoor photographs, the Euclidean distance $\mathrm{dist}_2(A, B)$ can be effectively estimated from the reduced versions $\mathbb{A}^{(k)}, \mathbb{B}^{(k)}$ at any scale k. Namely, let $\mathrm{dist}_2^{(k)}(\mathbb{A}, \mathbb{B})$, be the interval

$$\mathrm{dist}_2^{(k)}(\mathbb{A}, \mathbb{B}) = \sqrt{\frac{1}{4^k}\sum_{p\in\mathcal{D}^{(k)}} \Delta^2(\mathbb{A}^{(k)}[p], \mathbb{B}^{(k)}[p])} \qquad (5)$$

where $\Delta^2(a, b)$ is the interval with bounds

$$\Delta^2\downarrow(a, b) = (\mu a - \mu b)^2 + (\sigma a - \sigma b)^2 \qquad (6)$$

$$\Delta^2\uparrow(a, b) = (\mu a - \mu b)^2 + (\sigma a + \sigma b)^2 \qquad (7)$$

Formulas (5)– (7) are based on the observation that the difference between a signal and its mean value is orthogonal to the constant signal with that mean value. They yield an interval that contains the distance $\mathrm{dist}_2(A, B)$ in the original scale. Note that $\mathrm{dist}_2^{(0)}(\mathbb{A}, \mathbb{B})$ is a singleton interval containing only $\mathrm{dist}_2(A, B)$.

The $\mu\sigma$-reduction formulas (1)– (2) as well as the estimator formulas (5)– (7) are affected by roundoff errors, and by noise due to the quantization of $\mu\mathbb{I}^{(k)}[p]$ and $\sigma\mathbb{I}^{(k)}[p]$ when these are stored as digital images. Therefore, all those formulas must be computed using Moore's *interval arithmetic* [2], so that all arithmetic operations are properly rounded; and the quantization errors must be included intervals. These precautions are necessary to ensure that the interval estimate does indeed contain the exact distance.

B. Cumulative image distance

The Euclidean distance (3) is a basic tool for other image metrics, such as the *cumulative multiscale Euclidean distance* dist_2^*. This metric is a weighted mean of the Euclidean distances at all scales, that is,

$$\mathrm{dist}_2^*(A, B) = \sum_{r=0}^{m} \lambda_r\, \mathrm{dist}_2(\mu\mathbb{A}^{(r)}, \mu\mathbb{B}^{(r)}) \qquad (8)$$

The weights λ_r are numbers between 0 and 1, whose sum is 1. Typically they are chosen in geometric progression with some ratio β, that is $\lambda_r = \beta^r(\beta - 1)/(\beta^m - 1)$. If $\beta > 1$, the metric $\mathrm{dist}_2^*(A, B)$ gives more weight to the diferences

at the finest scales; if $\beta < 1$, it gives more weight to the diferences in the coarsest scales. If $\beta = 1$ all weights are equal ($1/m$). See figure 5 e 6.

query 0.222514 0.243298 0.243907

Figure 5. A query image and the three best matches with cumulative Euclidean distance dist_2^* with $\beta = 1/2$.

query 0.044553 0.047958 0.051262

Figure 6. A query image and the three best matches with cumulative Euclidean distance dist_2^* with $\beta = 2$.

Like the plain Euclidean distance, the cumulative image distance $\mathrm{dist}_2^*(A, B)$ can be efficiently estimated by looking only at the highest levels of the pyramids $\mathbb{A}, \mathbb{B}$. Namely, suppose we want to compute an estimate $\mathrm{dist}_2^{*\,(k)}(\mathbb{A}, \mathbb{B})$ for $\mathrm{dist}_2^*(A, B)$ using only levels $k, k+1, ..., m$ of the pyramids $\mathbb{A}, \mathbb{B}$. For this, we split formula (8) into four parts:

$$\mathrm{dist}_2^{*\,(k)}(\mathbb{A}, \mathbb{B}) = \lambda_0 d_0 + \sum_{r=1}^{k-1} \lambda_r d_r + \lambda_k d_k + \sum_{r=k+1}^{m} \lambda_r d_r$$

where $d_r = \mathrm{dist}_2(\mu\mathbb{A}^{(r)}, \mu\mathbb{B}^{(r)})$, for all r. The first term $\lambda_0 d_0$ is simply $\lambda_0 \mathrm{dist}_2(A, B)$ so can use the interval estimate $\mathrm{dist}_2^{(k)}(\mathbb{A}, \mathbb{B})$ of formulas (5)– (7), scaled by λ_0. The third part $\lambda_k d_k$ can be computed directly. For the fourth part $\sum_{r=k+1}^{m} \lambda_r d_r$, we just accumulate the values of $\lambda_r d_r$ previously computed for all higher levels. These two parts are exactly computedand therefore provide singleton intervals. Finally, for the part $\sum_{r=1}^{k-1} \lambda_r d_r$, we observe that the the distance d_r, for all $r < k$, lies in the interval $u = [d_k - \varepsilon, \mathrm{dist}_2^{(k)}(\mathbb{A}, \mathbb{B})\uparrow + \varepsilon]$, where ε is the expected magnitude of quantization errors in the images $\mu\mathbb{A}^{(r)}$ and $\mu\mathbb{B}^{(r)}$. Therefore, an estimate for the second part $\sum_{r=1}^{k-1} \lambda_r d_r$ is the interval u scaled by $\sum_{r=k+1}^{m} \lambda_r$. The interval-arithmetic sum of these four intervals will then contain $\mathrm{dist}_2^*(A, B)$.

V. Results

To evaluate the efficiency of MuSIS, we performed a number of closest-image searches on a database with 3840 monochromatic images derived from Corel photos [1]. All images had 128×128 pixels, and the search started at level $m = 7$, where the images have only 1×1 pixels.

Table I shows the computation costs of the MuSIS algorithm, compared to the brute-force algorithm. The numbers are averages of 100 searches with different query images. Each search was repeated twice, once using the plain Euclidean distance dist_2 and once using the cumulative distance dist_2^* with $\beta = 2$.

Table I
Costs of the MuSIS algorithm.

k	M_k	N_k	C_k	M'_k	N'_k	C'_k
7	0.0	3840.0	0.469	3840.0	3840.0	0.703
6	0.0	2278.8	1.113	651.0	651.0	0.477
5	0.0	1620.5	3.165	104.2	104.2	0.305
4	0.0	890.5	6.957	10.3	10.3	0.121
3	0.0	385.0	12.032	2.8	2.8	0.129
2	0.0	120.8	15.105	1.5	1.5	0.285
1	0.0	31.2	15.585	1.2	1.2	0.870
0	6.7	0.0	6.700	0.7	0.0	0.670
MUS			61.125			3.561
BRU			3840.000			5119.922
ρ			0.01592			0.00070

The columns M_k, N_k, C_k refer to queries using dist_2: M_k is the number of plain distances dist_2 computed between images at each level k; N_k is the number of interval estimates $\mathrm{dist}_2^{(k)}$ for the Euclidean distance at level 0, computed from the images at level k; and C_k is the cost of those operations. Columns M'_k, N'_k, and C'_k are the analogous counts and costs for queries using dist_2^*.

As in section III-C, the table assumes that the cost of dist_2 for two images at level k is 4^{-k} (in particular, 1 for level $k = 0$), and that of $\mathrm{dist}_2^{(k)}$ is $\delta 4^{-k}$. The factor 4^{-k} accounts for the relative number of pixels, and the factor of δ accounts for the overhead of fetching the two reduced versions of the image ($\mu\mathbb{B}^{(k)}$ and $\sigma\mathbb{B}^{(k)}$), and computing the interval estimate (5), instead of fetching a single image and computing a plain distance. For table I, we assumed $\delta = 2$.

Row MUS shows the total cost of a query by the MuSIS algorithm. Row BRU is the corresponding cost for the brute-force algorithm. The last row shows the relative efficiency of MuSIS, that is the ratio $\rho =$MUS$/$BRU.

VI. Conclusions

These preliminary tests show that the MuSIS algorithm, even in its simplest implementation, can substantively reduce the cost of searching for the closest image. The basic algorithm can be improved and extended in many ways, an can be combined with other traditional techniques such as clustering and application-specific descriptor extraction.

References

[1] Corel Corp. site, http://www.corel.com/. Accessed on 2009-12-13.

[2] R. E. Moore, *Methods and Applications of Interval Analysis*. SIAM, Philadelphia, USA (1979).

[3] R. Datta and D. Joshi and J. Li and J. Z. Wang, *Image Retrieval: Ideas, Influences, and Trends of the New Age*. ACM Computing Surveys, **40** (2), 1–60 (2008).