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. Experi-
ments indicate that MuSIS can provide significant speedup
relative to exhaustive search. The method 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 arith-
metic; 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 \(\text{dist}(\ast, \ast)\).

We assume that each image \(I\) is a function defined on
some finite rectangular domain \(D \subset \mathbb{Z} \times \mathbb{Z}\). 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 \(\mathcal{I} = (\mathcal{I}^{(0)}, \mathcal{I}^{(1)}, \ldots, \mathcal{I}^{(m)})\), where each \(\mathcal{I}^{(k)}\) 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 \(\mathcal{I}^{(m)}\) has a single pixel.

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

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

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

and \(\mathcal{P}^{(k)}[p]\) is the set of pixels in \(I\) that correspond to pixel
\(p\) of \(\mathcal{I}^{(k)}\). 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 \(\mathcal{I}^{(0)}\) does not have to be
computed or stored, since it can be trivially recreated from
the original image; namely \(\mu^{\mathcal{I}^{(0)}}[p] = I[p]\) and \(\sigma^{\mathcal{I}^{(0)}}[p] = 0,\)
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 \text{ 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 μσ-pyramids have been precomputed for all images \( B_i \) in the database and for the query image \( A \). The algorithm keeps a set \( C \) 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 dist(\( A, * \)).

The set \( C \) is kept as a collection of quadruples \( c = (c.B, c.B, c.k, c.d) \), where \( c.B \) is a handle to an image of the database, \( c.B \) is the precomputed pyramid of \( c.B \), \( c.k \) is a scale of resolution, and \( c.d \) is an interval estimate for dist\((A, c.B)\), computed from the reduced versions \( A^{(k)} \) and \( c.B^{(k)} \). 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 \( v\downarrow \leq x \leq v\uparrow \). The algorithm also maintains a global interval \( d^* \) such that dist\((A, B^*) \in d^* \). This interval is the minimum of all the intervals \( c.d \) in \( C \), that is \( d^*\downarrow = \min\{e.d\downarrow : e \in C\} \) and \( d^*\uparrow = \min\{e.d\uparrow : e \in C\} \).

1. Set \( C \) to contain all quadruples \((B, \emptyset, m+1, [0\downarrow, 1\uparrow])\) such that \( B \) is in the database. Set \( d^*\downarrow \leftarrow [0\downarrow, 1\uparrow] \).
2. Let \( c \) be a candidate with minimum \( c.d\downarrow \). If \(#C = 1\), return \( c.B \) as the answer to the query and stop. Otherwise let \( e \) be some other candidate with the second-smallest \( e.d\downarrow \). If \( e.d\downarrow \leq c.d\downarrow \), then return \( c.B \) as the answer to the query and stop.
3. Select some candidate \( c \) from \( C \). Compute a new interval estimate \( d' \) for dist\((A, c.B)\), using the reduced images \( A^{(c.k-1)} \) and \( c.B^{(c.k-1)} \). Set \( d' \leftarrow d' \cap c.d \). Replace the candidate \( c \) in the queue by \( c' = (c.B, c.B, c.k-1, d') \), and update \( d' \) accordingly.
4. Remove from \( C \) every candidate \( c \) which has \( c.d\downarrow > d^*\uparrow \). Repeat from step 2.

B. Correctness

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

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 \( C \) 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 $\delta$ times more expensive than the single scale version, and that the cost of computing $dist(A, B)$ is approximately $\alpha n$ for images with $n$ pixels, then the worst-case cost — computing $dist_2^{(k)}$ at all scales, for all $N$ database images — will be $N\alpha n(1 + \delta) + \delta/4^2 + \cdots + \delta/4^m$, 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 $\delta \alpha n (1/k^3 + 1/k^4 + \cdots + 1/n^m) \leq (\delta/3)/k$ 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 $d_1$, 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 $d_1$.

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

$$dist_2(A, B) = \frac{1}{\#D} \left| \sum_{p \in D} |A[p] - B[p]|^2 \right|$$

(3)

For efficiency, it is more convenient to work with the square of this distance, $dist_2^2(A, B)$. Note that, $dist_2(A, B) = \text{rms}(A - B)$, and $dist_2^2(A, B) = \text{msq}(A - B)$, where

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

(4)

If the pixel values are real numbers in $[0, 1]$, the value of $dist_2^2(A, B)$ (or $dist_2^2(A, B)$) 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 practical 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).

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 $dist_2(A, B)$ can be effectively estimated from the reduced versions $k^{(k)}$, $\overline{k}^{(k)}$ at any scale $k$. Namely, let $dist_2^{(k)}(A, B)$, be the interval

$$dist_2^{(k)}(A, B) = \left\{ \frac{1}{4^k} \sum_{p \in B^{(k)}} \Delta^2(k)[p], \overline{k}(k)[p] \right\}$$

(5)

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

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

(6)

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

(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 $dist_2(A, B)$ in the original scale. Note that $dist_2^{(0)}(A, B)$ is a singleton interval containing only $dist_2(A, B)$.

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 $\mu_2^{(k)}[p]$ and $\sigma_2^{(k)}[p]$ 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,

$$dist_2'(A, B) = \sum_{r=0}^{m} \lambda_r \text{dist}_2(\mu_2^{(r)}, \mu_2^{(r)})$$

(8)

The weights $\lambda_r$ are numbers between 0 and 1, whose sum is 1. Typically they are chosen in geometric progression with some ratio $\beta$, that is $\lambda_r = \beta^r (\beta - 1)/(\beta^m - 1)$. If $\beta > 1$, the metric $dist_2'(A, B)$ gives more weight to the differences
at the finest scales; if $\beta < 1$, it gives more weight to the differences in the coarsest scales. If $\beta = 1$ all weights are equal ($1/m$). See figure 5 e 6.

![Figure 5. A query image and the three best matches with cumulative Euclidean distance $\text{dist}^*_2$ with $\beta = 1/2$.](image1)

![Figure 6. A query image and the three best matches with cumulative Euclidean distance $\text{dist}^*_2$ with $\beta = 2$.](image2)

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

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

where $d_r = \text{dist}_2(\mu_{A(r)}, \mu_{B(r)})$, for all $r$. The first term $\lambda_0 d_0$ is simply $\lambda_0 \text{dist}_2(A, B)$ so can use the interval estimate $\text{dist}^*_2(k)(A, B)$ of formulas (5)–(7), scaled by $\lambda_0$. The third part of those operations. Columns $M'_k, N'_k,$ and $C'_k$ are the analogous counts and costs for queries using $\text{dist}^*_2$.

As in section III-C, the table assumes that the cost of $\text{dist}^*_2$ for two images at level $k$ is $4^{-k}$ (in particular, 1 for level $k = 0$), and that of $\text{dist}^*_2(k)$ is $4^{-k}$. The factor $4^{-k}$ reflects the relative number of pixels, and the factor of $\delta$ accounts for the overhead of fetching the two reduced versions of the image ($\mu_{B(k)}$ and $\sigma_{B(k)}$), 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 Moses shows the total cost of a query by the Moses algorithm. Row Brute is the corresponding cost for the brute-force algorithm. The last row shows the relative efficiency of Moses, that is the ratio $\rho = \text{Moses}/\text{Brute}$.

VI. CONCLUSIONS

These preliminary tests show that the Moses 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, as can be combined with other traditional techniques such as clustering and application-specific descriptor extraction.

REFERENCES

