Image Retrieval by Multi-Scale Interval Distance Estimation

C. E. A. Zampieri *J. Stolfi*

Technical Report - IC-09-39 - Relatório Técnico

October - 2009 - Outubro

Image Retrieval by Multi-Scale Interval Distance Estimation

Carlos Elias Arminio Zampieri[*] Jorge Stolfi[†]

Abstract

We describe a general method for query-by-example retrieval in image collections, using interval arithmetic to perform multi-scale distance estimation. The interval estimates are used to quickly eliminate candidate images at small scales, in a fashion similar to the branch-and-bound optimization paradigm. Experiments indicate that the method can provide significant speedup relative to exhaustive search; nevertheless, the method always returns the exact best match (and not merely an approximation thereof). The technique allows queries with a wide variety of image similarity functions, without the need to precompute or store specific descriptors for each function.

1 Introduction

Content-based image retrieval systems [1, 4] often assume that the database is pre-processed to compute 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 search. In this method, the preprocessing phase merely creates reduced several copies of each image I, at various scales. The reduction is done in the style of Moore's *interval arithmetic* [3, 2], so that each pixel of a reduced image is not the average of several pixels of I, as in ordinary image scaling, but rather their range — that is, a tight interval or box that contains all those pixel values.

2 Notation an definitions

2.1 Images and pixels

In this work we assume that an image I from the database is a function from some finite *domain* $\mathcal{D} \subset \mathbb{Z} \times \mathbb{Z}$ to some finite-dimensional space $\mathbb{V}$ of color values. Thus the value of I at a point p of the domain, called a *pixel* and denoted by $I[p]$, is a list of one or more real numbers, the *samples*, one for each spectral channel.

[*]Computing Institute, University of Campinas, 13081-970 Campinas, SP

[†]Computing Institute, University of Campinas, 13081-970 Campinas, SP

The samples are usually approximated by limited-precision floating-point numbers in computer memory, and quantized to a small set of values in image files; but we can ignore these details for the purposes of this paper. For simplicity, we will also assume that all images in the database have the same size and shape as the query image, and limit the discussion to monochromatic images, where $\mathbb{V}$ is the set of real numbers. The algorithms we describe can be trivially extended to work with color images and images with different sizes (see section 4.1).

2.2 Image pyramids

For each image I we define an *image pyramid* $I^{(0)}, I^{(1)}, \ldots, I^{(m)}$, where $^{(0)}$ is the original image I and each $I^{(k)}$ 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 last scale m is a parameter of the algorithm, usually set to the smallest integer such that $I^{(m)}$ has a single pixel.

Each pixel p of the reduced version $\mathbb{I}^{(k)}$ is computed from a certain subset $P^{(0:k)}[p]$ of the pixels of I. For any $i < j$, we also define the set $P^{(i:k)}[p]$ a being the smallest set of pixels q of $\mathbb{I}^{(i)}$ such that

$$P^{(0:k)}[p] = \bigcup_{q \in P^{(i:k)}} P^{(0:i)} \tag{1}$$

For example, if the image I has a square domain $\{0..\,2^m - 1\} \times \{0..\,2^m - 1\}$, we may divide $\mathcal{D}$ into square blocks $2^k \times 2^k$ pixels, and compute each pixel of $\mathbb{I}^{(k)}$ from the corresponding block of pixels of I. In that case, we would have

$$P^{(0:k)}[p] = \left\{ 2^k p + r : r \in \left\{0..\,2^k - 1\right\} \times \left\{0..\,2^k - 1\right\} \right\} \tag{2}$$

and, in general,

$$P^{(i:k)}[p] = \left\{ 2^{k-i} p + r : r \in \left\{0..\,2^{k-i} - 1\right\} \times \left\{0..\,2^{k-i} - 1\right\} \right\} \tag{3}$$

The pyramids occupy about $2(1 + 1/4 + 1/4^2 + \cdots + 1/4^m) < 8/3 \approx 2.67$ times as much space as the original database. Actually, the level zero version $I^{(0)}$ does not have to be stored since it can be trivially recreated from the (smaller) original image.

2.3 Image operations

The *maximum*, *minimum*, the *mean square value*, and the *root mean square value* of an image I over a set $P \subseteq \mathcal{D}$ of pixels are defined by the formulas

$$\max\nolimits_P I = \max\{\, I[p] : p \in P \,\} \tag{4}$$

$$\min\nolimits_P I = \min\{\, I[p] : p \in P \,\} \tag{5}$$

$$\mathrm{msq}_P\, I = \frac{1}{\#P} \sum_{p \in P} (I[p])^2 \tag{6}$$

$$\mathrm{rms}_P\, I = \sqrt{\mathrm{msq}_P\, I} = \sqrt{\frac{1}{\#P}\sum_{p\in P}(I[p])^2} \tag{7}$$

where $\#P$ is the cardinality of P. In all these operations, we omit the subscript P when $P = \mathcal{D}$.

2.4 Interval arithmetic

The technique of *interval arithmetic* (IA) was developed by R. Moore in 1960 to obtain guaranteed results in computations with uncertain data and/or approximate operations. In IA, each quantity $\bar{v}$ that would appear in an ordinary computation is represented by an *interval*, a pair of numbers $\bar{v} = [\bar{v}{\downarrow}, \bar{v}{\uparrow}]$ that represents the set of all real values x such that $\bar{v}{\downarrow} \leq x \leq \bar{v}{\uparrow}$. If $\bar{v}{\downarrow} = \bar{v}{\uparrow}$, the interval is said to be *trivial* or *exact*, and represents that single real number. If $\bar{v}{\downarrow} > \bar{v}{\uparrow}$, the interval is said to be *empty*, and represents the empty set; we will use $[]$ to denote any such interval. The endpoints my also be infinite ($+\infty$ or $-\infty$); in particular, the interval $[-\infty, +\infty]$ represents the set $\mathbb{R}$ of all real numbers.

Moore observed that any operation $\bar{z} \leftarrow f(\bar{x}, \bar{y})$ on ordinary quantities can be replaced by an interval operation $\bar{z} \leftarrow f(\bar{x}, \bar{y})$, in such a way that the computed interval $\bar{z}$ is guaranteed to contain the ordinary result $\bar{z}$ as long as the ordinary operands $\bar{x}, \bar{y}$ are contained in the intervals $\bar{x}, \bar{y}$ [3]. For example, the minimum, maximum, sum, and difference of two intervals $\bar{x} = [\bar{x}{\downarrow}, \bar{x}{\uparrow}]$ and $\bar{y} = [\bar{y}{\downarrow}, \bar{y}{\uparrow}]$ are the intervals

$$\begin{aligned}
\min(\bar{x}, \bar{y}) &= [\min(\bar{x}{\downarrow}, \bar{y}{\downarrow}), \min(\bar{x}{\uparrow}, \bar{y}{\uparrow})] \\
\max(\bar{x}, \bar{y}) &= [\max(\bar{x}{\downarrow}, \bar{y}{\downarrow}), \max(\bar{x}{\uparrow}, \bar{y}{\uparrow})] \\
\bar{x} + \bar{y} &= [\bar{x}{\downarrow} + \bar{y}{\downarrow}, \bar{x}{\uparrow} + \bar{y}{\uparrow}] \\
\bar{x} - \bar{y} &= [\bar{x}{\downarrow} - \bar{y}{\uparrow}, \bar{x}{\uparrow} - \bar{y}{\downarrow}]
\end{aligned} \tag{8}$$

The square of an interval $\bar{x}$ is only a bit more complicated

$$\bar{x}^2 = \begin{cases} [(\bar{x}{\downarrow})^2, (\bar{x}{\uparrow})^2] & \text{if } \bar{x}{\downarrow} \geq 0, \\ [(\bar{x}{\uparrow})^2, (\bar{x}{\downarrow})^2] & \text{if } \bar{x}{\uparrow} \leq 0, \\ [0, \max((\bar{x}{\downarrow})^2, (\bar{x}{\uparrow})^2)] & \text{otherwise.} \end{cases} \tag{9}$$

In this way, any formula can be replaced by an interval version that gives a guaranteed range for the result.

An important operation in IA is the *join* $\bar{u} \vee \bar{v}$ of two intervals $\bar{u}, \bar{v}$, defined as the smallest interval w that contains them. Note that $\bar{u} \vee \bar{v}$ is the same as the union $\bar{u} \cup \bar{v}$ only if one the intervals is empty, or if they have at least one value in common. This operation is associative and has the empty interval $[]$ as a neutral element.

2.5 Interval images and interval reduction

In our approach, each pixel of the reduced-scale images $\mathbb{I}^{(k)}$ is an *interval image*, where each pixel $\mathbb{I}^{(k)}[p]$ is a tight interval that contains all the pixel values $\mathbb{I}^{(0)}[q]$ for q in the set $P^{(0:k)}[p]$. Alternatively, we can view each image $\mathbb{I}^{(k)}$ as two real-valued images, $\mathbb{I}^{(k)}{\downarrow}[p] = \mathbb{I}^{(k)}[p]{\downarrow}$ and $\mathbb{I}^{(k)}{\uparrow}[p] = \mathbb{I}^{(k)}[p]{\uparrow}$.

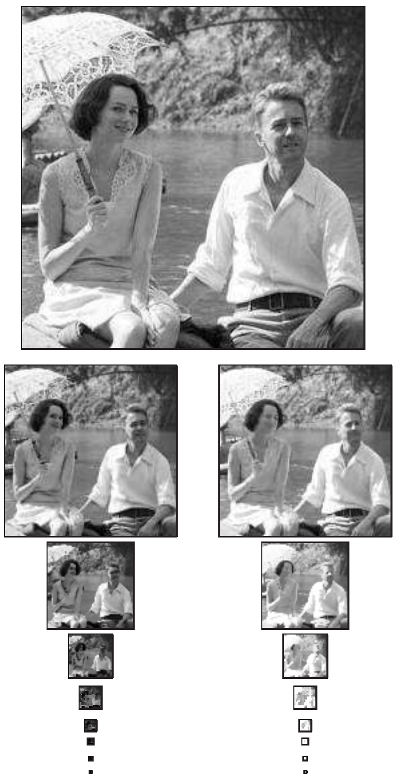


Figure 1: A monochromatic image, and interval versions of the same produced by IA-enclosure reduction.

Namely, each pixel $\mathbb{I}^{(k)}[p]$ is the join of all pixels $\mathbb{I}^{(k-1)}[q]$ such that $q \in P^{(k-1:k)}[p]$. We call this process *IA reduction*.

3 Multi-scale interval search

The fundamental idea of our algorithm, which we call *multi-scale interval search* (MuSIS) is that we can obtain guaranteed estimates of various numeric properties of images from their interval-reduced versions. Those estimates can be used to discard many database images without ever computing their precise distance from their query image.

In the MuSIS algorithm, we keep 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 some arbitrary metric dist. At the beginning, the set $\mathcal{C}$ is the entire database. The set is progressively pruned until only one image remains, which must be the correct answer B^*.

The set $\mathcal{C}$ is kept as a collection of quadruplets $(B, \mathbb{B}, k, d)$, where B is a handle to an image, $\mathbb{B}$ is the image precomputed of the pyramid in the database, k is a scale of resolution, and d is an interval estimate for $\operatorname{dist}(A, B)$, computed from the reduced versions $\mathbb{A}^{(k)}$ and $\mathbb{B}^{(k)}$.

We also maintain a global interval d^* such that $\text{dist}(A, B^*) \in d^*$. This interval is the minimum of all the d intervals in $\mathcal{C}$ as computed by formulas (8), that is

$$\begin{aligned} d^*\!\downarrow = &\quad \min\{\, t.d\!\downarrow : t \in \mathcal{C}\,\} \\ d^*\!\uparrow = &\quad \min\{\, t.d\!\uparrow : t \in \mathcal{C}\,\} \end{aligned} \tag{10}$$

The general situation during the search is illustrated in figure 2(top). Each error bar indicates the interval estimate d for some quadruplet $(B, \mathbb{B}, k, d)$ in $\mathcal{C}$. The dashed horizontal lines indicate the interval d^*, which here is determined by the high end of the estimate for candidate number 3, and the low end of candidate number 0. At each iteration, we remove a candidate from the list, recompute its distance interval $\text{dist}(A, B)$ at the next finer scale, and return it to the list. From this new interval d, we may be able to eliminate candidates from the list. Figure 2(bottom) shows the outcome of such an event: after candidate 0 was re-evaluated, it was possible to discard half the candidates from the queue without ever computing their exact distances from A.

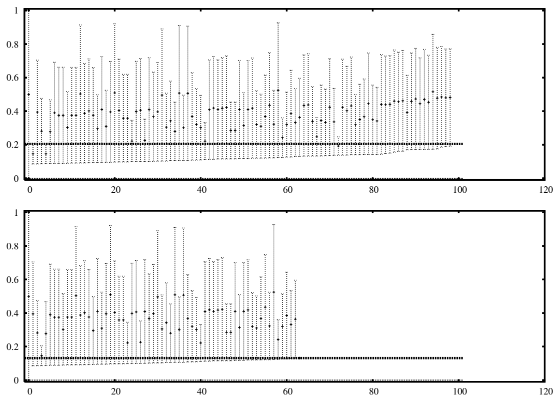


Figure 2: Two successive iterations of the algorithm.

The algorithm is described below.

1. Let m be the the maximum level in the pyramid of A. Set $\mathcal{C}$ to the set of all quadruplets $(B, \mathbb{B}, m+1, [0,1]$ such that B is in the database. Set the interval d^* to $[0,1]$.

2. Let $(B, \mathbb{B}, k, d)$ be a candidate with minimum $d\!\downarrow$. If that is only one candidate in the queue, return B as the answer to the query and stop. Otherwise let $(B', \mathbb{B}', k', d')$ be a candidate with the second-smalest $d\!\downarrow$. If $d\!\uparrow \le d'\!\downarrow$, then return B as the answer to the query and stop .

3. Select some quadruplet $(B, \mathbb{B}, k, d)$ from $\mathcal{C}$, by criteria that we will discuss later. Fetch the version $\mathbb{B}^{(k-1)}$ of B, and compute a new interval estimate d' for $\text{dist}(\mathbb{A}, \mathbb{B})$ from $\mathbb{A}^{(k-1)}$ and $\mathbb{B}^{(k-1)}$. Set $d' \leftarrow d' \cap d$. Replace the quadruplet $(B, \mathbb{B}, k, d)$ by $(B, \mathbb{B}, k-1, d')$, and update the interval d^* according to the definition (10).

4. Remove from $\mathcal{C}$ every quadruplet $(B, \mathbb{B}, k, d)$ for which d lies entirely above d^*, that is, which has $d\downarrow > d^*\uparrow$. Repeat from step 2

The new interval d' computed in step 3 should be a sub-interval of d, but, depending on how dist is computed, this may not be always true. In any case, if both intervals are correct interval enclosures for the exact distance $\operatorname{dist}(A, B)$, their intersection is correct too. The command $d' \leftarrow d' \cap d$ ensures that $d'\uparrow \leq d\uparrow$ and $d'\downarrow \geq d\downarrow$. 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, we need to keep the tuples in a heap data structure, sorted by $d\downarrow$, with the minimum at the root.

3.1 Correctness

The main loop of the algorithm preserves the following invariant: there is a quadruplet $(B, \mathbb{B}, k, d)$ in $\mathcal{C}$ such that B is the correct answer B^*, for which $\operatorname{dist}(A, B)$ is minimum. This is obviously true at the beginning, and, assuming that the intervals d are valid (that is, the distance $\operatorname{dist}(A, B) \in d$ for every quadruplet $(B, \mathbb{B}, k, d)$), step 4 only eliminates a quadruplet $(B, \mathbb{B}, k, d)$ if $\operatorname{dist}(A, B)$ is guaranteed to be larger than $\operatorname{dist}(A, B')$ where B' is the candidate image in $\mathcal{C}$ that defined $d^*\uparrow$ (candidate 3).

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

3.2 Efficiency

In step 3, if $d'\uparrow < d\uparrow$ then $d^*\uparrow$ will decrease, and some other quadruplets may be removed from $\mathcal{C}$ in the next iteration of step 4. On the other hand, if $d'\downarrow > d\downarrow$, that quadruplet may be deleted instead. Of course neither of these conditions may occur, in which case no candidate is eliminated.

The worst case for this algorithm is when none of the quadruplets are eliminated, and the iteration continues until all quadruplets have $k = 0$. This means that $\operatorname{dist}(\mathbb{A}^{(k)}, \mathbb{B}^{(k)})$ will be computed for all N images B in the database and all scales k from 0 to m. Assuming that the interval-valued version of dist is at most C times more expensive than the single-valued version, and that the cost of fetching the image B and computing $\operatorname{dist}(A, B)$ is approximately Dn for images with n pixels, then the worst-case cost of computing all those distances will be $NDn(1 + C/4 + C/4^2 + \cdots + C/4^m)$, which is less than $NDn(1 + C/3)$. In comparison, the brute-force algorithm has cost NDn. Therefore, the worst-case of the algorithm is only $1 + C/3$ times more expensive than the brute-force algorithm.

If the cost of computing dist is super-linear on the image size, the preprocessing overhead is proportionally less. Namely, if $\operatorname{dist}(A, B)$ costs approximately Dn^r for images with n pixels, then the cost of computing the interval pyramid is $C/4^r + C/4^{2r} + \cdots + C/4^{mr})$ which is less than $C/(4^r - 1)$.

However, the *average case* performance may be much better. Each tuple $(B, \mathbb{B}, k, d)$ with $k > 0$ that is eliminated in step 4 will have cost $CDn(1/4^k + 1/4^{k+1} + \cdots + 1/4^m) \leq (C/3/4^{k-1})Dn$ operations, but its elimination will save us from computing the distance $\mathrm{dist}(A, B)$ at full resolution, which is Dn. The ratio of costs is $C/3/4^{k-1}$, with 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.

3.3 Which candidate to expand

In step 3, there are many possible strategies that can be used to select the candidate tuple $(B, \mathbb{B}, k, d)$ that is to be refined. We use the following heuristic: let $(B', \mathbb{B}', k', d')$ and $(B'', \mathbb{B}'', k'', d'')$ be the two candidates with smallest $d{\downarrow}$. If their levels k' and k'' are different, select the one with *largest* k. If the levels are equal, select the tuple with smallest $d{\downarrow}$.

Note that one cannot have $k' = k'' = 0$ 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.

3.4 Influence of the image metric

The correctness of the MuSIS algorithm is independent of the image distance function dist. In fact, dist needs not be a metric in the mathematical sense of the term. Also, replacing dist by any monotonic function $\mathrm{dist}'(A, B) = f(\mathrm{dist}(A, B))$ of it will result in equivalent output.

However, the choice of dist may have a significant impact on the algorithm's efficiency. If dist is entirely dependent on detail that is seen only at scales smaller than a certain scale r, then no quadruplets will be discarded from $\mathcal{C}$ until they have all have $k \leq r$. Thus, MuSIS as described above is ineffective for queries in databases of fingerprints, or of scanned novels.

4 Image distance

For our tests we used the most basic metric for evaluating the discrepancy between two images, the *normalized Euclidean distance*. If the images A, B have the same domain P, it is defined by

$$\mathrm{dist}_2(A, B) = \frac{1}{\#P} \sqrt{\sum_{p \in P} |A[p] - B[p]|^2} \tag{11}$$

In other words, $\mathrm{dist}_2(A, B) = \mathrm{rms}(A - B)$. If the pixel values are real numbers between 0 and 1, the value of $\mathrm{dist}_2(A, B)$ is also between 0 and 1; and the distance is 0 if and only if the images are identical.

This metric assumes that the desired image B^* is approximately aligned with the query image A so that corresponding pixels in the two images refer to the same point (or nearby points) on the objects, and the photometric parameters (lighting conditions, brightness, contrast, camera response, etc.). When these conditions are not satisfied, one may still be able to use the Euclidean distance with a suitable preprocessing of the images (e.g. replacing each image by its gradient image, in order to reduce the effect of lighting variations).

4.1 Extension for images with different domains

In many databases the images have different sizes and aspect ratios, and/or are accompanied by binary *masks* that specify the actual domain. To extend the Euclidean metric to such images, one may assume that any pixel $A[p]$ that lies outside $\mathcal{D}\,A$ is a special "null" color value $\emptyset$, such that $|\emptyset - v| = |u - \emptyset| > |u - v|$ for any non-null colors u, v, and $|\emptyset - \emptyset| = 0$. Then formula (11) may be used, with P being the union of the domains of all images in the database.

The Euclidean metric is trivially extended to color images, by treating the channel index as just another image dimension — that is, by computing the man of the squared differences over all samples in all three channels.

4.2 Distance estimation

For many kinds of images, such as outdoor photographs, the Euclidean distance $\mathrm{dist}_2(A, B)$ between two images can be effectively estimated from their interval-reduced versions $\mathbb{A}^{(k)}, \mathbb{B}^{(k)}$ at any scale k. It suffices to evaluate $\mathrm{dist}_2(\mathbb{A}^{(k)}, \mathbb{B}^{(k)}) = \mathrm{rms}(\mathbb{A}^{(k)} - \mathbb{B}^{(k)})$ as in formulas (7), with the interval arithmetic operations (8)– (9). Namely, we compute

$$d{\downarrow} = \sqrt{\frac{1}{\#P} \sum_{p \in P} (\mathrm{dst}\downarrow(\mathbb{A}^{(k)}[p], \mathbb{B}^{(k)}[p]))^2} \tag{12}$$

$$d{\uparrow} = \sqrt{\frac{1}{\#P} \sum_{p \in P} (\mathrm{dst}\uparrow(\mathbb{A}^{(k)}[p], \mathbb{B}^{(k)}[p]))^2} \tag{13}$$

where

$$\mathrm{dst}\downarrow(a, b) = \begin{cases} 0 & \text{if} [a{\downarrow}, a{\uparrow}] \cap [b{\downarrow}, b{\uparrow}] \neq \{\}\,, \\ |a{\downarrow} - b{\uparrow}| & \text{if} a{\downarrow} > b{\uparrow}, \\ |a{\uparrow} - b{\downarrow}| & \text{if} a{\uparrow} < b{\downarrow} \end{cases} \tag{14}$$

$$\mathrm{dst}\uparrow(a, b) = \max\{a{\uparrow} - b{\downarrow},\ b{\uparrow} - a{\downarrow}\} \tag{15}$$

These formulas give an interval d that is guaranteed to contain the exact distance $\mathrm{dist}_2(A, B)$ at the original scale, and is often narrow enough to allow us to decide which of two candidates B', B'' is closer to the query A.

Figures 3(a) through 3(d) show these interval estimates (bars) between an image $\mathbb{A}^{(k)}$ (top) and two other images $\mathbb{B}^{(k)}$, $\mathbb{C}^{(k)}$ (middle and bottom), at various scales of resolution — from $k = 7$ (1×1 pixel) to $k = 7$ (original, 128×128 pixels). Note that the interval computed at scale $k = 2$ is already sufficient to decide that $\mathrm{dist}_2(A, B) < \mathrm{dist}_2(A, C)$, so the latter does not have to be computed at scales 1 and 0.

For the purposes of the MuSIS algorithm, it is more convenient to work with the square of the distance, $\mathrm{dist}_2^2(A, B) = \mathrm{msq}(A - B)$.

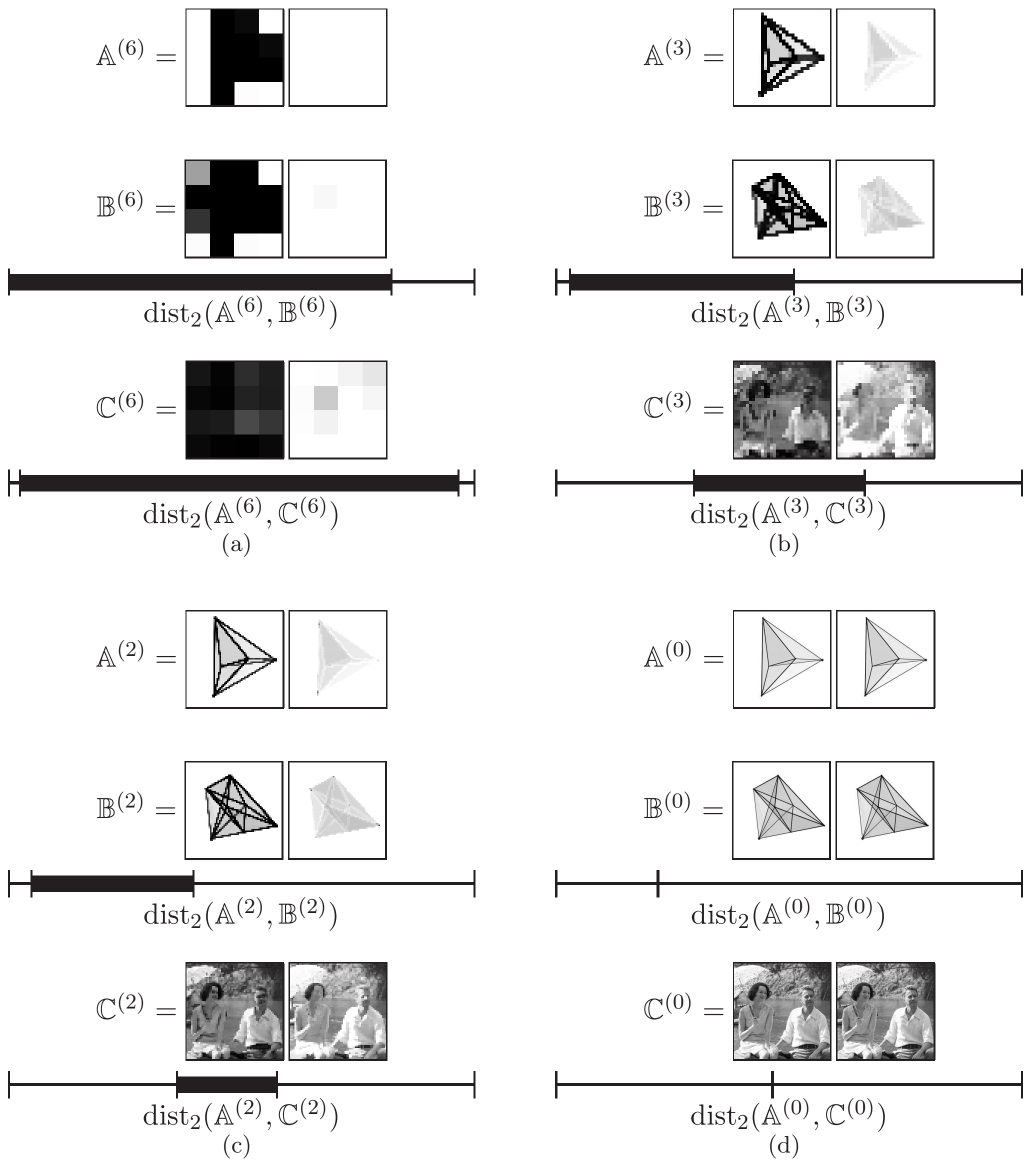
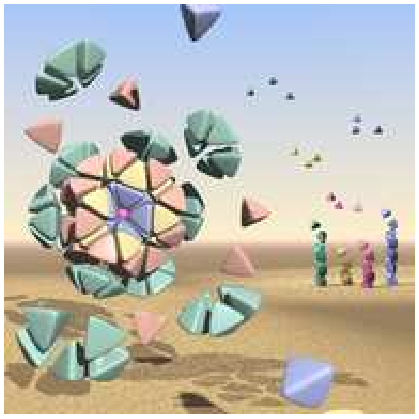

$\mathbb{A}^{(6)} =$
$\mathbb{B}^{(6)} =$
$\mathrm{dist}_2(\mathbb{A}^{(6)}, \mathbb{B}^{(6)})$
$\mathbb{C}^{(6)} =$
$\mathrm{dist}_2(\mathbb{A}^{(6)}, \mathbb{C}^{(6)})$
(a)
$\mathbb{A}^{(3)} =$
$\mathbb{B}^{(3)} =$
$\mathrm{dist}_2(\mathbb{A}^{(3)}, \mathbb{B}^{(3)})$
$\mathbb{C}^{(3)} =$
$\mathrm{dist}_2(\mathbb{A}^{(3)}, \mathbb{C}^{(3)})$
(b)
$\mathbb{A}^{(2)} =$
$\mathbb{B}^{(2)} =$
$\mathrm{dist}_2(\mathbb{A}^{(2)}, \mathbb{B}^{(2)})$
$\mathbb{C}^{(2)} =$
$\mathrm{dist}_2(\mathbb{A}^{(2)}, \mathbb{C}^{(2)})$
(c)
$\mathbb{A}^{(0)} =$
$\mathbb{B}^{(0)} =$
$\mathrm{dist}_2(\mathbb{A}^{(0)}, \mathbb{B}^{(0)})$
$\mathbb{C}^{(0)} =$
$\mathrm{dist}_2(\mathbb{A}^{(0)}, \mathbb{C}^{(0)})$
(d)

Figure 3: Interval estimates at various scales of resolution.

5 Results

To illustrate the algorithm,we present the results of a searching for the closest image among a set of 999 RGB images with the MuSIS algorithm, compared to the cost of a brute-force search. All images have 192×192 pixels and the comparisons started at level 6, where the images have only 3×3 pixels. The query image and the returned image are shown in figure 4(a) and 4(b), respectively.

Table 1 shows the computation costs for this test. The table assumes that the cost of a distance computation at scale $k > 0$ (including the cost of fetching the candidate image from the database) is $2 \cdot 4^{-k}$ times the cost of computing the distance at scale 0. The factor 4^{-k} accounts for the relative number of pixels, and the factor of 2 accounts for the overhead of fetching the two versions ($\mathbb{B}^{(k)}\downarrow$ and $\mathbb{B}^{(k)}\uparrow$) computing the distance with interval arithmetic instead of plain floating point.


(a) Query image for the test.

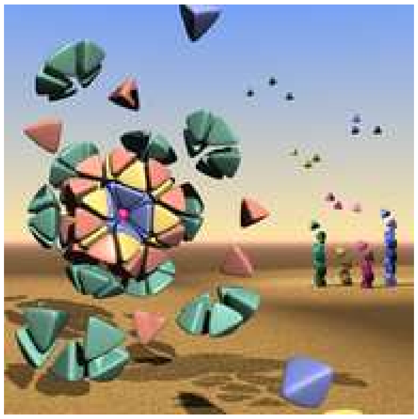

(b) Closest image returned for the query.

Figure 4: Results of a searching for the closest image among a set of images.

Table 1: Number N_k of Euclidean distances computed at each scale k, cost C_k of those comparisons (1 = one distance at full scale), total cost of search, and ratio of total cost to brute-force cost.

k	N_k	C_k
06	999	0.487793
05	978	1.910156
04	541	4.226562
03	44	1.375000
02	8	1.0000000
01	1	0.500000
00	1	1.0000001
tot		10.499512
rel		0.010500

6 Conclusions

These preliminary tests show that the MuSIS algorithm, even in its simplest implementation, can 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] M. Flickner and H. Sawhney and W. Niblack and J. Ashley and Q. Huang and B. Dom and M. Gorkani and J. Hafner and D. Lee and D. Petkovic and D. Steele and P. Yanker, *Query by Image and Video Content: The QBIC System*. IEEE Computer Society, Los Alamitos, CA, USA (1995).

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

[3] R. E. Moore, *Interval Analysis*. Prentice-Hall, (1966).

[4] 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).