Asociación Argentina de Mecánica Computacional

Mecánica Computacional Vol XXIX, págs. 6045-6064 (artículo completo)
Eduardo Dvorkin, Marcela Goldschmit, Mario Storti (Eds.)
Buenos Aires, Argentina, 15-18 Noviembre 2010

ANISOTROPIC K-NEAREST NEIGHBOR SEARCH USING COVARIANCE QUADTREE

Eraldo P. Marinho and Carmen M. Andreazza

Universidade Estadual Paulista (São Paulo State University), Departamento de Estatística, Matemática Aplicada e Computação – Rio Claro, São Paulo, Brazil

Keywords: k nearest neighbors, quadtrees, hyper-quadtrees, principal components analysis, anisotropy

Abstract. We present a variant of the hyper-quadtree that divides a multidimensional space according to the hyperplanes associated to the principal components of the data in each hyperquadrant. Each of the hyper-quadrants is a data partition in a multidimensional subspace, whose intrinsic dimensionality is reduced from the root dimensionality by means of the principal components analysis, which discards the irrelevant eigenvalues of the local covariance matrix. In the present method a component is irrelevant if its length is smaller than, or comparable to, the local inter-data spacing. Thus, the covariance hyper-quadtree is fully adaptive to the local dimensionality. The proposed data-structure is used to compute the anisotropic K nearest neighbors (kNN), supported by the Mahalanobis metric. As an application, we used the present k nearest neighbors method to perform density estimation over a noisy data distribution. Such estimation method can be further incorporated to the smoothed particle hydrodynamics, allowing computer simulations of anisotropic fluid flows.

1 INTRODUCTION

There is a number of problems in pattern recognition that requires efficient algorithms to search for the k nearest neighbors (kNN) in multidimensional feature spaces, (e.g., McNames, 2001; Yu et al., 2001; D'haes et al., 2003; Mouratidis et al., 2005). For instance, color segmentation of images requires a kNN algorithm to select parts of the context whose color range matches a query in the color space (e.g., Rehrmann and Priese, 1998). Algorithms for cluster analysis of multivariate data are better improved if based upon kNN techniques for estimating both the class-independent and the class-conditioned probability densities (e.g., Tran et al., 2006). Moreover, many pattern recognition problems might require density estimation techniques, which are in turn efficiently performed if combining kNN algorithms with kernel interpolation, which is a generalization of Parzen's windows (e.g., Bishop, 1995; Duda and Hart, 1973).

Pattern recognition techniques are applied even in contexts with concerns other than of the usual information retrieval or scene analysis problems but as a helper on solving the conservation equations in computer simulation of fluid dynamics. For instance, *smoothed particle hydrodynamics, SPH* (e.g., Gingold and Monaghan, 1977; Lucy, 1977) is a kernel-based technique to perform computer simulation of fluid dynamics using particles as input vectors. The technique simulates the continuum by means of particle interpolated quantities, using similar approaches of kernel based density estimation.

Usually smoothed particle hydrodynamics codes require a kNN approach to decide the kernel parameters (e.g., Marinho and Lépine, 2000; Marinho et al., 2001), and references therein, in order to perform both density estimation and interpolated spatial derivatives of the fluid quantities as, for instance, pressure gradient, stress-tensor divergence and Maxwell's stress divergence. Also SPH is useful in plasma simulations as shown in Jiang et al. (2006). Nowadays, SPH is an important particle method used even in incompressible fluid simulations as presented in Xu et al. (2009). Recent works have focused the full adaptivity of the density estimation technique of SPH concerning the anisotropy in interface regimes as discussed formerly by Owen et al. (1998) and more recently by Liu et al. (2006).

We introduce the concept of covariance hyper-quadtree as an extension of the traditional hyper-quadtree data structure, but now taking into account that the orthogonal directions through which the space is subdivided into hyper-quadrants are set by the covariance eigenvectors. Moreover, this novel approach includes the principal components analysis method on the reduction of the data dimensionality in each node.

Covariance trees are a relatively novel concept on hierarchical data decomposition. However, we may find in the literature that such terminology is somehow mismatching both on its purpose and in its data structure definition as it has been shown for instance in the works of Burschka et al. (2004); Ma and Ji (1998).

From the Burschka et al. (2004) view point, a covariance tree is a variant of the k-D tree, in which the uniform orthogonal coordinate system is replaced by the local coordinate system of the covariance eigenvectors, centered on the center of mass of the data partition (node). In other contexts, covariance trees are defined to map the strategies of multiscale stochastic processes (e.g., Ribeiro et al., 2006).

A term nearly similar to covariance quadtrees was presented in the literature in a previous work of Minasny et al. (2007), but in a quite different approach, and still preserving the traditional concept of quadtrees, in which the image tessellation is taken along a fixed set of orthogonal directions.

In the present work, we get ride of the Burschka et al. (2004) idea of covariance tree, which is a strict binary tree, to introduce the concept of *covariance quadtree* as a generalization of the traditional quadtree rectangular image tessellation Finkel and Bentley (1974). In the present approach, each data partition has its own local orthogonal coordinate system centered on the local data expectation with the coordinate axes defined by the local covariance eigenvectors, which in turn are the normal directions of the clipping hyperplanes.

We apply the covariance quadtree to the kNN search using a bimodal metric, which is a variant of both McNames (2001) and (D'haes et al., 2003, hereafter D'DR) scheme. In the D'DR method, the search data structure is limited to a balanced binary tree, which maps the recursive split of each data partition by the median hyperplane whose normal is the covariance's principal component (see Figures 1 and 2 of D'DR). We extended this interesting idea to the covariance quadtree, whose number of derived nodes spans from 2 up to 2^D children, where D is the space dimensionality.

The main purpose of the present paper is the presentation and validation of an efficient algorithm of finding the exact K nearest neighbors from a query vector in a multidimensional data space, either considering isotropic or anisotropic searches. The concept of anisotropy is presently interpreted in terms of the Mahalanobis metric. Since the k-nearest neighbors have an ellipsoidal support, set by the covariance matrix for this region, the anisotropic search requires a bootstrapping scheme of finding the optimal k-nearest neighbors morphology. The applications of the present method for both image processing and anisotropic SPH simulations shall be presented in future works.

The paper is structured as follows. In Section 2 is introduced the concept of covariance quadtree, whose geometrical principles are discussed in subsection 2.1, and the computational aspects of its data structure are discussed in subsection 2.2. In subsection 2.3 is discussed the idea of intrinsic dimensionality which may change along the tree construction if regarding each node as vector subspace spanned from the node expectation, having the principal components as the orthonormal basis of the subspace. The 2D image interpretation of the covariance quadtree is illustrated with some examples in subsection 2.4, which helps the reader to interpret correctly the main idea of the covariance quadtree. In Section 3 are discussed the methods of finding the isotropic (subsection 3.1) and the anisotropic (subsection 3.2), Mahalanobis based, k-nearest neighbors. The benchmark is discussed in Section 4 for the isotropic case. The anisotropic search has the same time complex multiple of the isotropic case, which depends on the number of the required iterations until convergence. One simple application for image construction from noise images is shown in section 5. Further considerations, perspectives and conclusions on the introduced method are made in Section 6.

2 COVARIANCE QUADTREE

2.1 Principles

In order to introduce the concept of covariance quadtree we recall first to the classical idea of a quadtree. Historically Finkel and Bentley (1974) a quadtree is the hierarchical data structure which maps the division of a rectangle into four subrectangles, so that each internal node derives four child nodes, and an external node derives a terminal partition, or image segment, which is a rectangular atom of the image. A simple example of a quadtree image tessellation is given in Figure 1 of the original image of Lenna shown in Figure 2.

An immediate generalization of quadtree is its multidimensional form, which divides a d-dimensional hyper-rectangle into 2^d child hyper-rectangles, or hyper-quadrants. Hyper-quadtrees

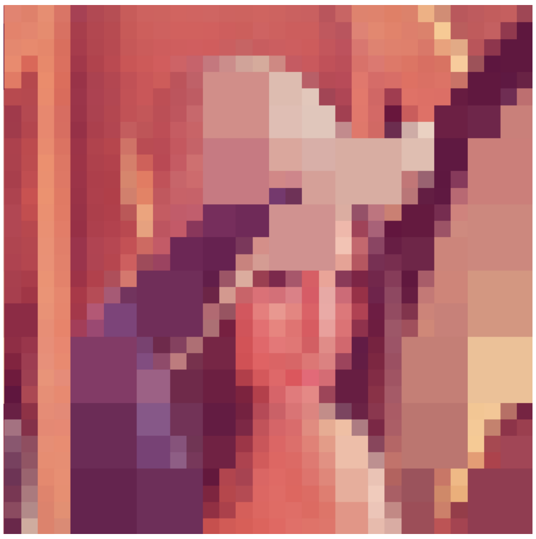

Figure 1: The quadtree tessellated image of Figure 2 for about 10% tolerance in the gray-scale standard deviation.

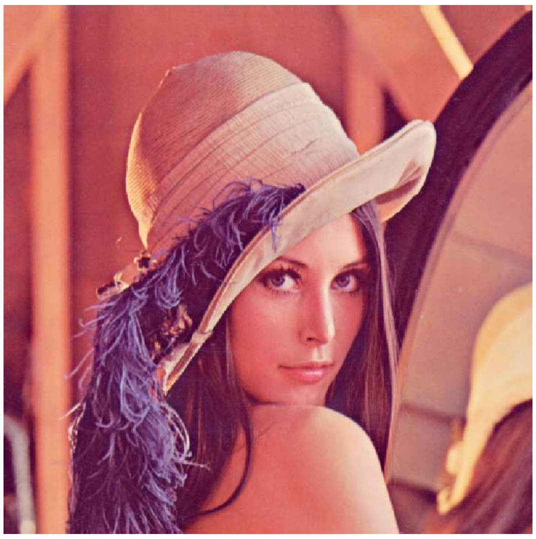

Figure 2: The standard 512×512, 24 bit, Lenna's portrait.

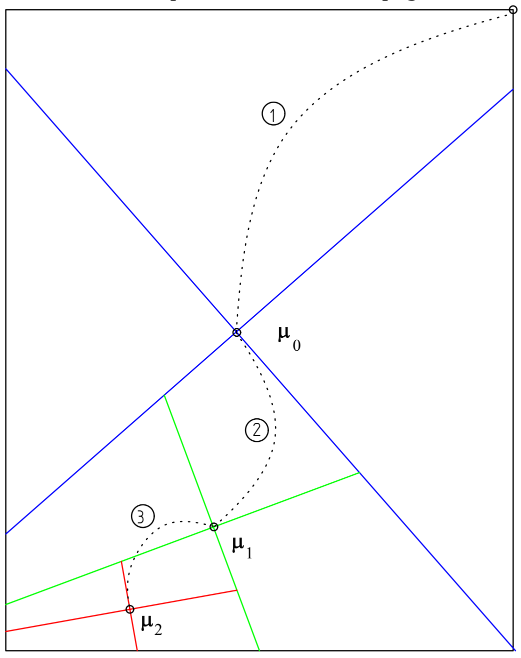

Figure 3: The first three steps of the top-down construction of a covariance quadtree. The outer rectangle is the image frame, and the orthogonal strait lines, crossing the local expectation, are the principal directions of the image segment. The dotted line illustrates the top-down path.

preserve the aspect ratio of the the hyper-quadrants with the root.

For considerations of brevity, we assume hereafter that the term quadtree applies both for the 2D case and for multidimensional case.

A covariance hyper-quadtree, or simply covariance quadtree, is a generalization of the traditional quadtree, with the difference that covariance hyperplanes cuts the data space through the data expectation, as sketched in Figure 3. Henceforth, we call such a spatial decomposition as *covariance tessellation* to make distinction from the classic quadtree tessellation.

The covariance quadtree is the Gaussian multivariate extension of the traditional quadtree data structure, but now with the difference that each node spans from the local expectation μ a vector space $\mathbb{S}$, whose basis is the set of the local principal eigenvectors $\{\mathbf{v}_1, \ldots, \mathbf{v}_\lambda\}$. Any input vector $\mathbf{x}$ is then written down as $\mathbf{x} = \mathbf{\Delta} + \mu$, where $\mathbf{\Delta}$ is a vector in $\mathbb{S}$. Moreover, each covariance node carries out the local aspect ratio in the form of the λ principal eigenvalues $\{\alpha_1, \ldots, \alpha_\lambda\}$, which by their own define the local data anisotropy as well as its intrinsic dimensionality λ, where λ is the number of principal components of the covariance matrix.

Each of the principal covariance eigenvectors sets up an orthogonal hyperplane passing through the expectation μ. Such a hyperplane is called a covariance hyperplane, and for consistency the subspace $\mathbb{S}$ is the covariance subspace. Each covariance hyperplane $\Pi_\mathbf{v}$ splits $\mathbb{S}$ into two adjacent regions having $\Pi_\mathbf{v}$ as interface and the expectation μ as a common vertex. Thus, the λ-covariance hyperplanes divide the covariance subspace into 2^λ partitions or hyper-quadrants, which are assigned to 2^λ child nodes.

Each internal node obeying some division condition is subdivided by the covariance hyperplanes into 2^λ children. The covariance hyperplane is defined as the hyperplane which crosses the expectation and is directed by some of the covariance eigenvectors. as its normal direction. The number λ of principal components is called the intrinsic dimensionality of the local data distribution.

If the number n of input vectors associated to the node is smaller than the original data dimensionality d then the intrinsic node dimensionality λ is limited to $0 \leq \lambda \leq n-1$.

2.2 Tree build

2.2.1 Covariance quadtree data structure

The covariance quadtree is a hierarchical data structure formed by nodes whose attributes are the following:

1. a data subset represented as a list of the input vectors embedded in the space partition;
2. the expectation and the λ-principal components evaluated over the local data content;
3. a lower-bounding corner formed by the parent-evaluated expectation and the principal components renormalized in order to point outward the space partition;
4. a link to visit the 2^λ possible children.

The parental link is implemented by assigning a 2^λ-dimensioned array of pointers to child nodes. Given the data set $\mathbb{X}_\nu$, which is assigned to the node ν, there are 2^λ disjoint subsets $\mathbb{X}_{\nu_\beta}$, each of which is assigned to the respective child node ν_β.

Each of the n_ν input vectors in $\mathbb{X}_\nu$ is assigned (or moved) to one of the 2^λ partitions $\mathbb{X}_{\nu_\beta}$ $(\beta \in \mathbb{Z}_{2^\lambda})$, which means that there is a map $\beta : \mathbb{R}^\lambda \mapsto \mathbb{Z}_{2^\lambda}$ to derive a childhood from the interior node ν:

$$\mathbb{X}_\nu \xrightarrow{\beta} \left(\mathbb{X}_0 \quad \ldots \quad \mathbb{X}_{2^\lambda - 1} \right)$$

where $\mathbb{Z}_{2^\lambda} = \left\{0, \ldots, 2^\lambda - 1\right\}$.

We adopt the following hash function, namely the hyperquadrant classifier or child index, to map a input vector from its origin data set $\mathbb{X}_\nu$ to a (child) data partition $\mathbb{X}_{\nu_\beta}$:

$$\beta(\mathbf{\Delta}) = \sum_{j=1}^{\lambda} 2^{\lambda-j}\, H(\mathbf{\Delta}^{\mathrm{T}}\, \mathbf{v}_j), \tag{1}$$

where $\mathbf{\Delta}$ is the data deviation about the expectation μ_ν, $\mathbf{v}_j$ is the j-principal eigenvector, and H is the following step function:

$$H(x) = \begin{cases} 1, & x \geq 0; \\ 0, & x < 0. \end{cases} \tag{2}$$

A 2D instance of the child indexing scheme can be seen in Figure 4.

After successive divisions each node is bounded not only by its lower-bounding corner but also by the lower-bounding corners from all of its ancestors but the root. The latter can be unbounded unless for the sake of the problem details which can impose an input frontier as e.g. the frame illustrated in Figure 3. In this case, any interior node is wrapped by a convex hull, formed by the innermost of the corners collected from all of the node ancestors. As a result, the node boundary is then formed by the innermost hyperplanes from both the local-lower bounding corner and the parent boundary. The node boundary is then a synthesized attribute combined from the inherited parent boundary and from the lower bounding corner.

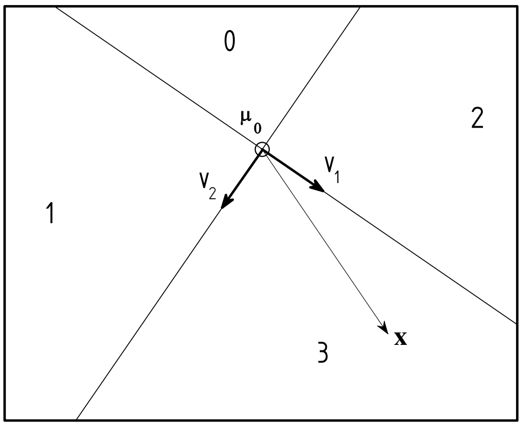

Figure 4: Child numbering order with respect to the principal components detected in the dividing node according to equations (1) and (2).

In multidimension spaces it is somehow entangling to assemble a data structure to represent the node boundary. However, in 2D-spaces it is too simple to determine the boundary in terms of minimal polygons formed by the straight lines orthogonal to the principal components.

If the root is unbounded, its childhood is lower-bounded by infinite hyperpyramids having the root expectation as their common vertex. An important result, not proven here, is that if the root is either unbounded or it has a convex boundary, any interior node boundary is either an infinity hyperpyramid or a convex hyperpolyhedron.

Since the set of covariance hyperplanes forms 2^λ lower bounding corner with the expectation as the common vertex, each of these principal hyperplanes isolates pair of children. Thus, each child has λ common face siblings. A 2D sketch of the covariance-quadtree tessellation was given in Figure 3. From the child point of view, any interior input vector faces the concave part of the lower bounding corner, which requires a direction multiplier $\phi \in \{-1, +1\}$ in order to have the child normal vectors ever pointing outward its region.

From equation (1), we have that the leftmost bit of the child index β (unsigned integer) represents the data point orientation with respect to the principal hyperplane ($\mathbf{v}_1$-direction): 0, if the eigenvector is pointing outward the data region, and 1, if $\mathbf{v}_1$ is pointing inward. Thus, the direction multiplier ϕ_j, as a function on both the j-principal eigenvector and the child index β, is given by

$$\phi_j = (-1)^{\,\mathrm{bit}(j,\,\beta)}, \tag{3}$$

where the function $\mathrm{bit}(j,\ \beta)$ returns the j th left-to-right bit of the child index β. Consequently, for each eigenvector $\mathbf{v}_j$, given the child index β, corresponds the lower-bounding corner normal vectors $\mathbf{u}_{\beta_j}$ given by

$$\mathbf{u}_j = \phi_j\ \mathbf{v}_j \tag{4}$$

2.3 Intrinsic dimensionality

An interesting aspect of the covariance quadtree is that its spatial tessellation keeps track of the residual covariance left on each child partition. This is an interesting development since it reveals the natural adaptivity of the decomposition method to the partition's intrinsic dimensionality, yielded from the principal components analysis, PCA, aka *the Karhunen-Loève transform* (e.g., Jollife, 1986; Bishop, 1995).

The proposed decision rule for the principal components analysis is that the next training

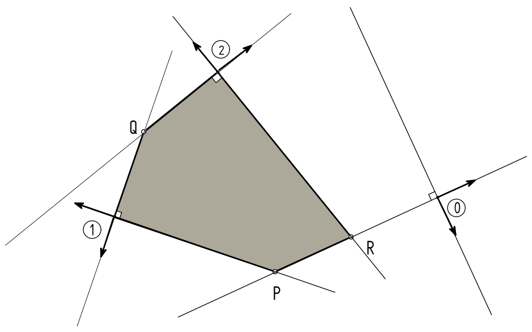

Figure 5: A simplified situation in which the node boundary is assembled by combining the parent boundary with the local lower- bounding corner. The parent boundary is an inherited attribute from its ancestors which is presented in the form of a polygonal. The lower-bounding corner intercepts some straight lines of the parent boundary which has at least two solutions. The solution is the innermost one which is faced by the concave part of the lower-bounding corner.

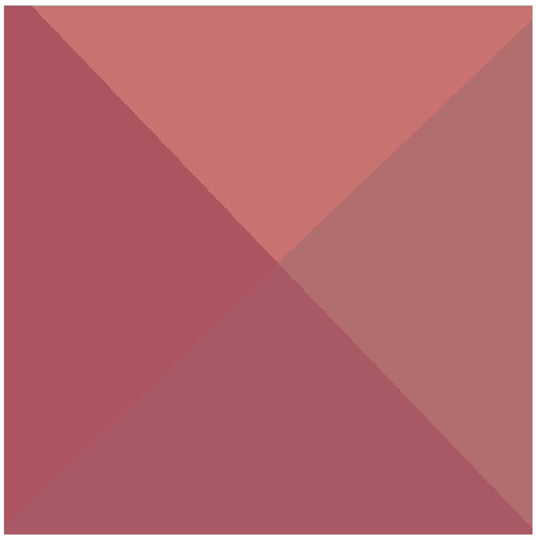

Figure 6: The first level covariance quadtree partition from the original image in Figure 2. Each partition is filled up with the mean RGB color. Compare this image with the sketch in Figure 3.

component, which is ever smaller than or equal to the previous component, must be greater than the mean data spacing interior to the hyper-rectangle having as edges the square root of the previously estimated components.

The intrinsic dimensionality is decided either by limiting the number of relevant components, or by spatial singularities such a data distribution having no dispersion in some dimensions as the case of the local data being entirely distributed in a surface. As the number n of input vectors of a given partition becomes smaller than the original data dimensionality d, the intrinsic dimensionality λ is geometrically constrained to $\lambda \leq n - 1$. Thus, a two-point partition has intrinsic dimensionality ONE ($\lambda = 1$).

Top-level space partitions (nodes) likely have higher intrinsic dimensionality than lower-level partitions. Moreover, our space division is performed through the expectation, which implies that the division centers (node vertices) tend to be closer to the denser child partitions than to the rarefied ones.

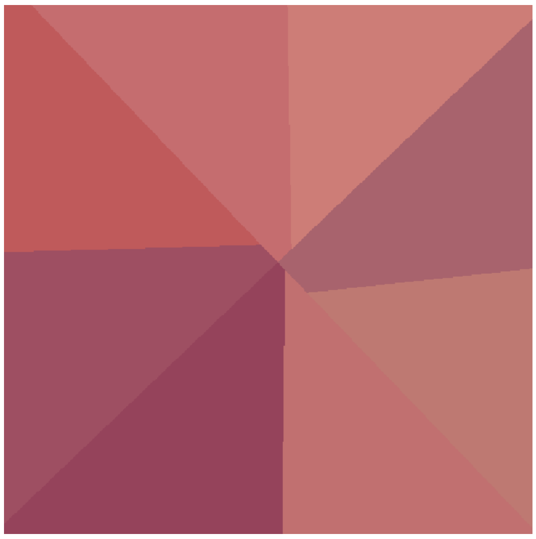

Figure 7: The second level covariance quadtree partition from the original image in Figure 2. The local dimensionality is reduced if the minor component is below 50% the major one.

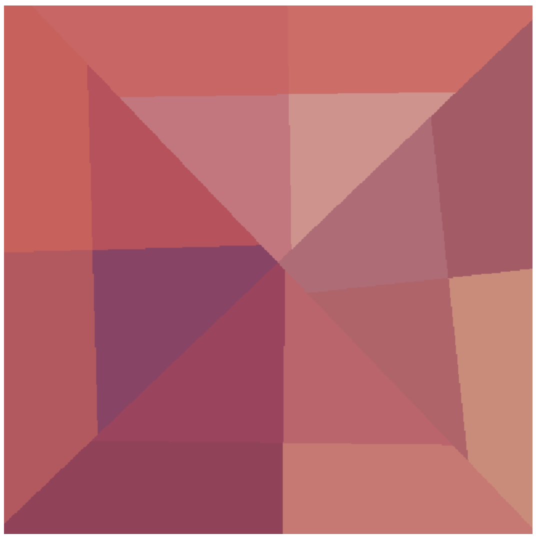

Figure 8: As in previous figure, but with no dimensionality detection.

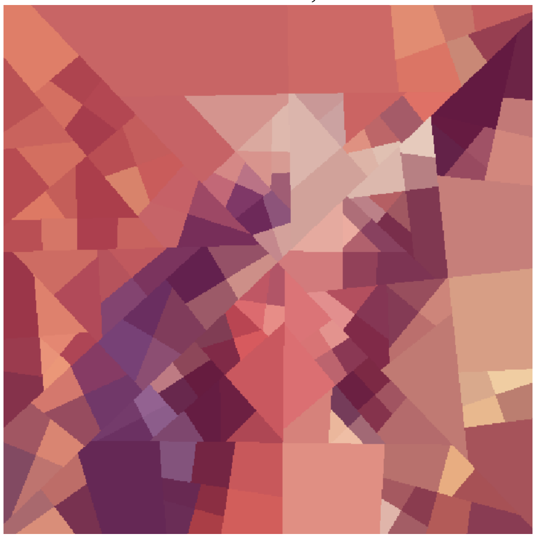

Figure 9: Level 4 covariance quadtree, within 50% tolerance for the dimensionality detection.

2.4 Two-dimensional case: image examples

To better illustrate the covariance quadtree datastructure, we three examples of image tessellation by a hierarchy of grey scale covariance. in this case, the root image is divided into 2 or 4 partitions, depending on the dimensionality reduction criterion applied.

In the following examples we assume a simple dimensionality reduction scheme, where an image partition is considered 2D if the minor-to-major component ratio is above a prefixed threshold, and then is divided into 4 partitions directed by the eigenvectors. Otherwise, the partition is cut perpendicularly to the principal component.

Each of the λ hyperplanes from the lower-bounding corner intercepts other λ hyperplanes from the parent boundary to determine a vertex on the λ-dimension hyperpolyhedron. In the particular case of 2D images, as illustrated in Figure 2, we have the situations depicted in Figures 6 through 10.

Figure 6 illustrates the first level in the covariance quadtree for Figure 2, assuming no dimensionality threshold used on choosing the covariance principal components. Each image partition is filled up with the mean RGB-color. The descent stop condition is that the partition's greyscale dispersion is below a fraction (tolerance parameter) of the parent's greyscale dispersion.

Figure 7 resumes the image tessellation from the previous Figure 6, but assuming a simple dimensionality reduction. If the minor-to-major components ratio is smaller than 0.5, the dimensionality is reduced to ONE ($d = 1$).

Similar to Figure 7, Figure 8 resumes the image tessellation from the previous Figure 6, but assuming no dimensionality reduction ($d = 2$, regardless the tree depth).

Resuming from Figure 7, we have the covariance quadtree situation for level 4 shown in Figure 9. Now, some partitions are considered 1D, and other ones are considered 2D, depending on the aspect ratio of each resulting partition. Finally, in the level 7 of the covariance quadtree, we have the tessellated image shown in Figure 10.

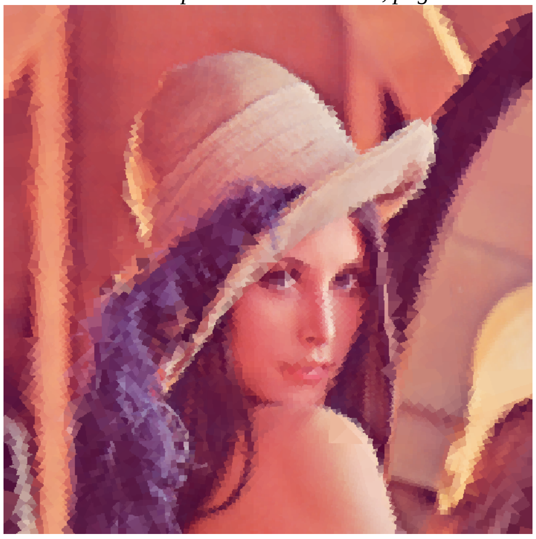

Figure 10: Resuming from the tree situation shown in Figure 9, until Level 7 in the covariance quadtree, also adopting 50% tolerance for the dimensionality detection.

3 COVARIANCE-BASED KNN

3.1 Isotropic search

The present covariance-based K nearest neighbors algorithm is a top-down search that recursively proceeds the descent through nodes which may have some point closer than any of the k ($\leq K$) collected nearest candidates until that level in the tree.

The adopted query-to-node distance is one of many forms of expressing distance of a point to a set. To compose the query-to-node distance, it is first required to define a pseudo distance, namely the direct query-to-node distance $D(.)$ as the following function:

$$D(\mathbf{x}, \texttt{node}) = \max_{j=1}^{\lambda} \left\{ (\mathbf{x} - \mu)^{\mathrm{T}} \, \mathbf{u}_j \right\}, \tag{5}$$

where μ is the node's lower-bound corner vertex and $\{\mathbf{u}_1, \ldots, \mathbf{u}_\lambda\}$ the set of its normal unit vectors as discussed in Subsection 2.2. Since the internal product $(\mathbf{x} - \mu)^{\mathrm{T}} \, \mathbf{u}_j$ is positive only if the query point is facing the j-hyperplane from outside the node, the direct distance defines a lower boundary for a collection of candidate vectors to comprise the list of the k nearest neighbors.

Further examining equation (5) one may conclude that if the query vector is facing the hyperplanes from inside, then $D(\mathbf{x}, \texttt{node}) \leq 0$. The equality occurs if, and only if, the query is lying in one of the node's boundaries.

Since the covariance nodes but the root are bounded not only by its principal hyperplanes but also by at least one of its ancestors boundary, we write the complete query-to-node distance as the following recursive function:

$$d(\mathbf{x}, \texttt{node}) = \max \left\{ d(\mathbf{x}, \mathrm{parent}(\texttt{node})), D(\mathbf{x}, \texttt{node}) \right\}, \tag{6}$$

with the breaking condition that $d(\mathbf{x}, \texttt{node}) = 0$, if $\texttt{node} = \texttt{root}$.

The distance calculation in equation (6) is performed along the tree descent, and the returned value from the recursive call for $d(.)$ in the RHS is an inherited attribute from the node's ancestor.

It sounds like a theorem that the distance from any query vector to the covariance node is zero if the query is included in the node. In fact, the direct distance $D(\mathbf{x}, \texttt{node})$ is negative or null if the query is interior or it is in some of the node's principal hyperplanes, which is valid to the distance from the query to some of its ancestors until the root, where the distance given by equation (6) is trivially zero. Since the maximum from zero and a non-positive number is zero, the distance from the query to any node which encloses it is zero.

Algorithm 1 The C-code for the recursive top-down kNN algorithm.

```
extern query_type q;
extern list_type kNN_list;

void kNN_find (node_t *cell, double d)
{
    d = query_to_node_distance (q, cell, d);
    if ( kNN_list.N == K && d > kNN_list.top_dist)
        return;
    else {
        if (cell->N > 1) {
            int  b;
            for (b = 0; b < cell->nchilds; b++)
                if (cell->child[b])
                    kNN_find (cell->child[b], d);
        }
        else
            kNN_insert (cell, d);
    }
}
```

The present kNN scheme requires an upper-bounded neighbor list (kNN list), which stores each input vector according to its query-to-data distance, so that the end term is the outermost one from the query neighborhood. The closest neighbor is of course the first element of the list after the complete search. Each input vector is pointed to by a data descriptor, which is stored both in the neighbor list as well as in the covariance quadtree nodes as previously discussed.

For the present, a neighbor is an input vector, which is assigned to a unit node (leaf). The kNN list has K members at most, where K is the user assigned number of nearest neighbors to find. The list is initially empty and it progressively grows with the successive insertion of nearest neighbor candidates along the descent until the list is fully populated with K members. Henceforth, any new insertion implies in removing the farthest element from the list.

Positive query-to-node distance denotes the maximum of the distances from the query to the hyperplanes bounding the node. On the other hand, the distance from the query to a hyperplane is the Euclidean distance from the query to the closest point in the hyperplane. Thus, if the farthest of the kNN candidates is still closer than the focused node, then no closer neighbors can be found in that node. This is the refutation criterion to be applied along the tree descent if the kNN list is fully populated with the K nearest candidates.

Conversely, a given node potentially has kNN candidates if either the kNN list is not full yet,

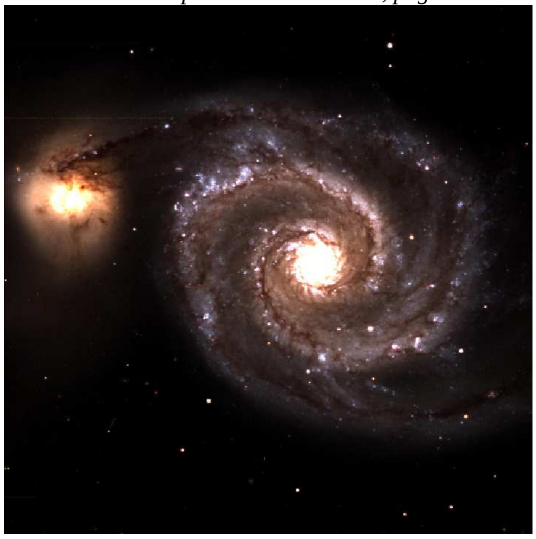
Figure 11: The original RGB image of the M51 galaxy, whose gray-scale (R+G+B) is used to drive the random distribution shown in the next two figures.

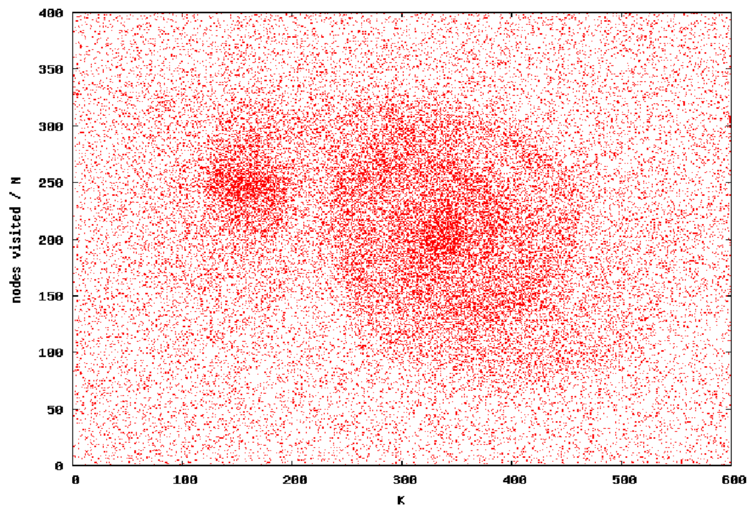

Figure 12: A Plot of 44,081 random points sampling the gray-scale image of the galaxy M51.

or the query-to-node distance is below or equal to the distance from the query to the outermost candidate in the kNN list. Particularly in this case, if the node is unit, the calculated distance is the Euclidean point-to-point distance, and then the insertion method is called to solve in what position in the list, if any, the new candidate might be inserted.

Both paragraphs above are summarized in the form a C code, listed in the Algorithm 1. The proposed method is a recursive descent, with a prefixed call to the query-to-node distance function given in equations (5) and (6). The second argument, `d`, in `kNN_find` is initially the distance from the query to the parent node, which might be modified in the assignment statement at line 6 conform equation (5).

According to the algorithm, the stop condition for tree descent is that the kNN list is full and that the query-to-node distance is greater than the distant to the outermost list member, ending the recursion through line 8. Line 10 decides whether or not the algorithm recursively descends (lines 11-14) or it includes the new neighbor candidate to the kNN list (line 17).

As an example of the search for exact kNN, we performed the search on a random distri-

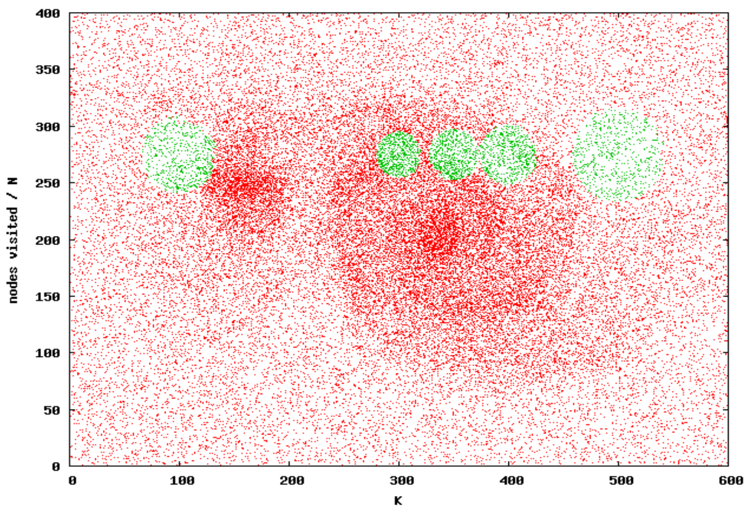

Figure 13: A plot of K=320 nearest neighbors (in green) for 5 arbitrarily chosen points from previous figure.

bution of points which was drawn by the probability density set proportional to the gray-scale distribution of M51 spiral galaxy image (see Figure 11), whose data points are plotted in Figure 12. Five query points were arbitrarily chosen as shown in Figure 13.

3.2 Anisotropic search

The algorithm discussed in previous section performs an isotropic k-nearest neighbor search using the Euclidean metric. However, as the covariance quadtree itself suggests, the search algorithm is worth applied to detect a fixed number of the nearest neighbors accordingly to the Mahalanobis metric Mahalanobis (1936), which is the natural choice to find the nearest vectors accordingly to the covariance matrix estimated for the neighborhood found.

The Mahalanobis distance ξ from a query vector $\mathbf{x}$ to the expectation μ of a given data distribution explained by a covariance $\mathbf{\Sigma}$ is defined as the following positive definite bilinear form:

$$\xi^2 = (\mathbf{x} - \mu)^{\mathrm{T}} \mathbf{\Sigma}^{-1} (\mathbf{x} - \mu) \tag{7}$$

In the present case, equation (7) is valid for a data partition assigned to as a covariance quadtree node. Thus, μ is the node vertex about which further division might occur.

Let $\{\mathbf{u}_j \,|\, j = 1, \ldots, D\}$ be the set of all eigenvectors of the covariance matrix, evaluated for the K-nearest neighbors, and $\left\{\sigma_j^2 \,|\, j = 1, \ldots, D\right\}$ its eigenvalues. Then, the inverse of the covariance matrix $\mathbf{\Sigma}^{-1}$ may be rewritten in the diagonal form:

$$\mathbf{\Sigma}_{\mathrm{kNN}}^{-1} = \sum_{j=1}^{D} \frac{1}{\sigma_j^2} \mathbf{u}_j \mathbf{u}_j^{\mathrm{T}} \tag{8}$$

If regarding the principal components analysis to perform the dimensionality reduction, the number of relevant eigenvectors λ might be smaller than the input space dimensionality D, as discussed in subsection 2.3.

The query-to-point distance $\xi^2_{(\mathbf{x},\mathbf{x}_j)}$ is a modification in the Mahalanobis distance so that

$$\xi^2_{(\mathbf{x},\mathbf{x}_j)} = (\mathbf{x} - \mathbf{x}_j)^{\mathrm{T}} \mathbf{\Sigma}_{\mathrm{kNN}}^{-1} (\mathbf{x} - \mathbf{x}_j) \tag{9}$$

The distance from the query to a covariance quadtree node defined in previous subsection is just the maximum Euclidean distance from the query to the node hyperplanes. The modification of this distance to the Mahalanobis metric is then

$$\xi^2_{(\mathbf{x},\mathrm{node})} = d^2_{(\mathbf{x},\mathrm{node})} \mathbf{u}^{\mathrm{T}}_{j_{\max}} \boldsymbol{\Sigma}^{-1}_{\mathrm{kNN}} \mathbf{u}_{j_{\max}}, \tag{10}$$

where $d^2_{(\mathbf{x},\mathrm{node})}$ is the query-to-node distance calculated via equation (6), and $\mathbf{u}_{j_{\max}}$ is the node's normal vector that gave the maximum contribution in evaluating the RHS of the equation (6).

Algorithm 1 is robust regardless the metric chosen to classify the covariance quadtree nodes as kNN candidates. Thus, no changes are required in the algorithm of the previous subsection. Only the distance definition must be modified to the Mahalanobis metric to perform the anisotropic search. Moreover, since we do not know prior the covariance of the not yet known K nearest anisotropic neighbors, the kNN modification requires bootstrapping.

The bootstrapping algorithm to perform the anisotropic search for the k nearest neighbors under the Mahalanobis metric works as follow:

1. Initialize the kNN covariance matrix with the identity matrix.
2. Repeat:
   (a) Perform the kNN via Algorithm 1 with the distances modified according to equations (9) and (10);
   (b) For the present kNN list, calculate the covariance matrix;
3. Until convergence occurs on the covariance eigenvalues.

Convergence occurs in few steps even if we adopt a tight tolerance as for instance 10^{-19} used in the runs of present work.

To illustrate the anisotropic search, we performed the method above for the same data used in previous subsection. Figure 14 shows the points plotted in green corresponding to the $K = 410$ neighbors found for 9 queries conveniently chosen to reveal morphological aspects of the galaxy noise image. The leftmost upper green spot is almost isotropic since corresponds to the image background. However, in denser regions, the kNN profiles follow the spiral pattern of the galaxy M51.

4 BENCHMARK

The time complexity of the top down kNN search shown in Algorithm 1 is measured by counting the total number of nodes visited and the total number of insertions of unit nodes into the kNN list per query. This latter corresponds to the number of individual points which were inserted into the kNN list regardless they remained in the list until the search ended. The number of node candidates is greater than the desired number K of nearest neighbors of the query point.

There are two critical costs on the present search complexity. The first one concentrates on the total number of internal nodes visited along the entire search per query (Figure 15). The other one is the fraction of visited nodes which has in fact a kNN candidate – it means that the more efficient is the search the smaller is the proportional number of rejected nodes (Figure 16).

Examining Figure 16 one may see that the search efficiency increases with the increase of the prefixed number K of the nearest neighbors found. The plausible explanation for this speedup is that the K/N ratio is roughly the probability of finding some of the K nearest neighbors of a given query point.

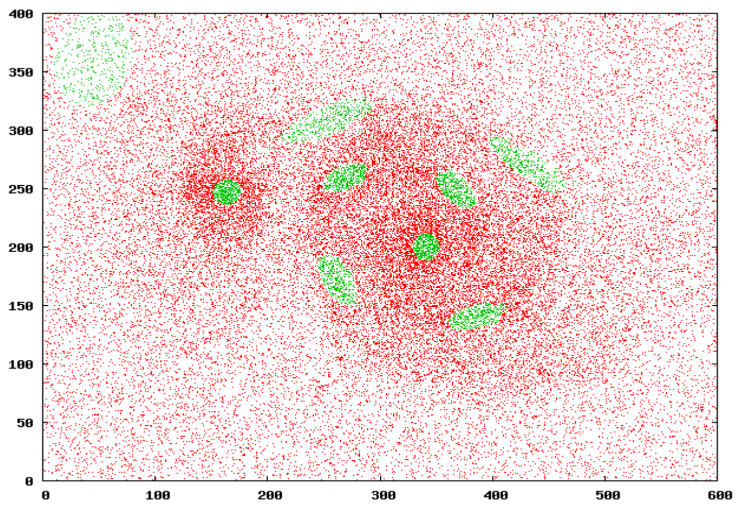

Figure 14: A plot of the K=410 nearest neighbors (in green) for 8 arbitrary queries for the data in figure 12 but performing a Mahalanobis based anisotropic search. Note that the green spots align with the preferential directions of the data distribution.

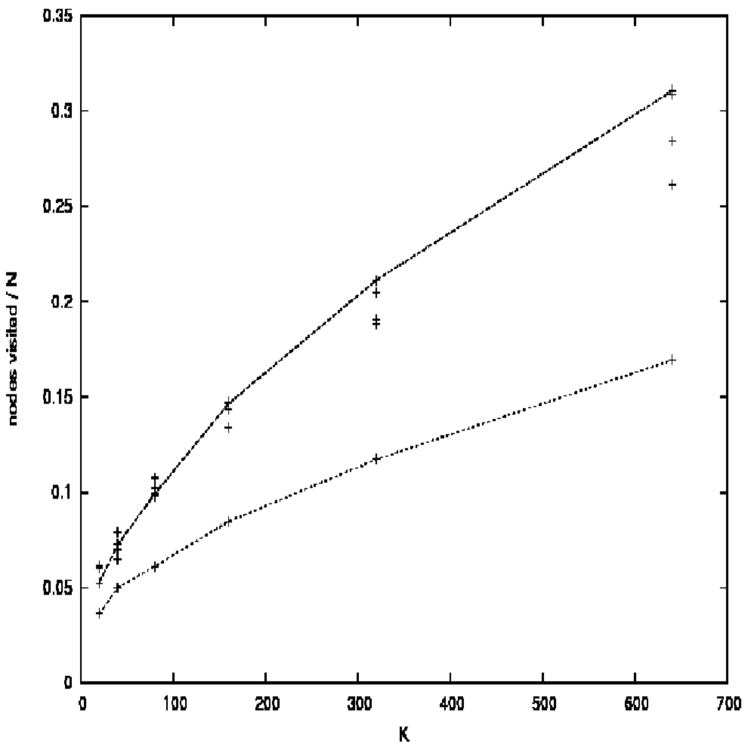

Figure 15: Number of nodes visited per data point versus the number of K nearest neighbors found. Upper and lower curves are plotted for the maximum and minimum number of candidate nodes found, respectively.

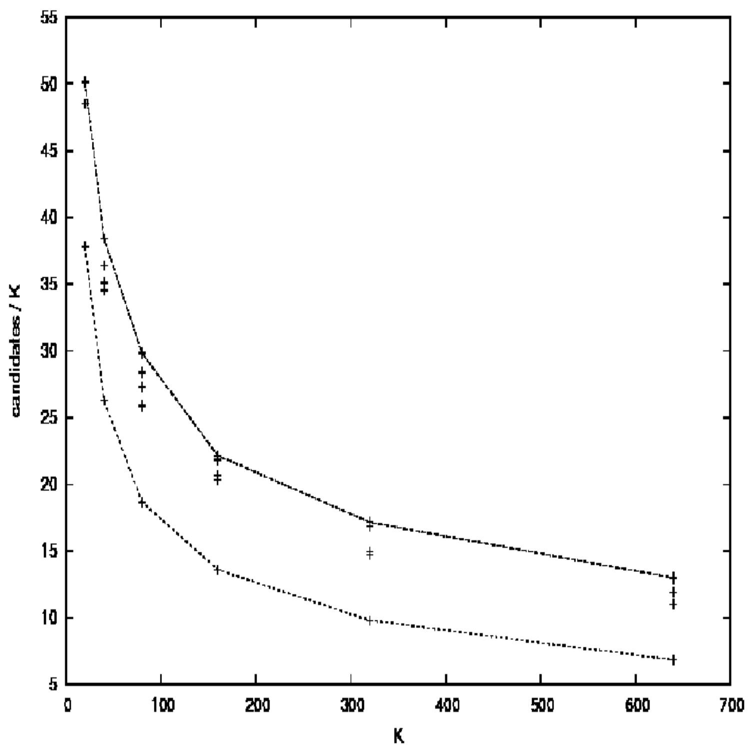

Figure 16: Number of candidate nodes visited per neighbor found versus the number of K nearest neighbors. Upper and lower curves matches the maximum and minimum candidates found.

The search complexity depends on the data distribution, so that in low contrast regions the time complexity is smaller than finding the kNN in higher contrast distributions. This effect is responsible for the dispersion between the upper and lower bounding curves shown in both figures 15 and 16.

5 APPLICATION. ANISOTROPIC INTERPOLATION

To illustrate one of a number of purposes of the covariance quadtree method, we show an image reconstruction from a noise image, as in the data points shown in Figure 12. The reconstruction is performed using a density estimation technique based on compact support kernels. Usually, a kernel is a good function, which is a member from a Dirac's δ-sequence.

The adopted kernel is a radial basis function, whose support radius is set by the distance from the origin to the outermost point from the K-nearest neighbors. Adopting the Mahalanobis distance, the kernel support is an ellipsoid whose semi-major axes are defined by the square root of the principal components of the covariance matrix estimated for the K-nearest neighbors.

The anisotropic density is estimated from the following summation:

$$\rho(\mathbf{x}) = \frac{1}{N} \sum_{j} \frac{K(\xi_j(\mathbf{x}))}{\sigma_a \sigma_b} \tag{11}$$

where the metric function ξ_j is given by

$$\xi_j^2(\mathbf{x}) = \frac{|(\mathbf{x} - \mathbf{x}_j)^{\mathrm{T}} \mathbf{u}_a|^2}{\sigma_a^2} + \frac{|(\mathbf{x} - \mathbf{x}_j)^{\mathrm{T}} \mathbf{u}_b|^2}{\sigma_b^2} \tag{12}$$

$\mathbf{u}_a$, $\mathbf{u}_b$ are the two-dimensional eigenvectors, σ_a^2, σ_b^2 are the eigenvalues, $\mathbf{x}$ is the query point, in which the estimation is made, and $\mathbf{x}'$ is the neighboring position. The kernel function K is

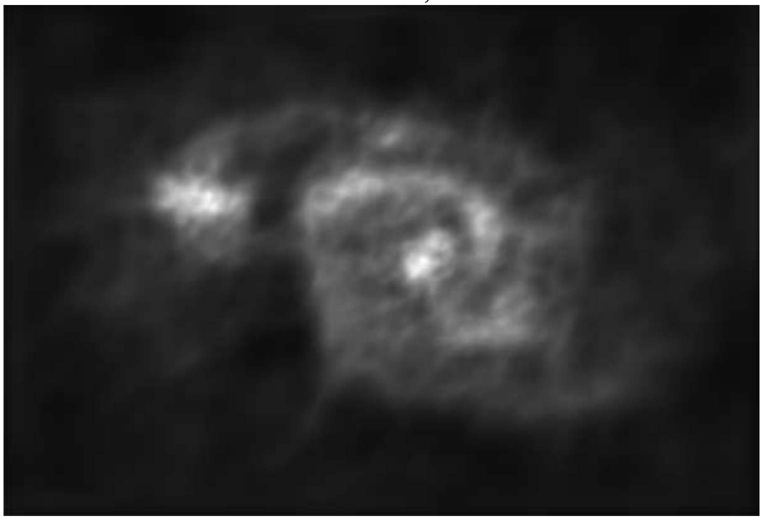

Figure 17: The anisotropically interpolated image of theM51 galaxy, using only 4,401 randomly chosen points from the 44,081 shown in Figure 12.

normalized in order to give

$$\int \rho(\mathbf{x})\mathrm{d}x^2 = 1. \tag{13}$$

To illustrate the anisotropic density estimation we recall the examples previously given with a random distribution of 44,081 points drawn from the gray-scale image of the spiral system M51. The integral given in equation (11) is easily translated to a grid image so that densities are normalized to comprise the 8-bit color range of the output image in Figure 17. The density estimation shown in Figure 17 was performed over 4,401 ($\sim 10\%$) of the points shown previously in Figure 12, which reduced considerably the computational cost of the gray-scale construction.

The integral in (11) is effectively performed over the kernel support, which is the ellipsoidal region encompassing the $K = 200$ nearest neighbors found with the method described in Subsection 3.2. The adopted kernel model to perform the density estimation shown in Figure 17 was the cubic B-spline, largely used in the SPH literature (e.g., Gingold and Monaghan, 1977; Owen et al., 1998; Pelupessy et al., 2003).

6 CONCLUSION

The present work is a preliminary approach to introduce and validate a fast algorithm to perform anisotropic search for the exact k nearest neighbors. The adopted concept of anisotropy was based on the Gaussian multivariate interpretation of the local data distribution assigned to the covariance quadtree nodes. The method detects spatial structures by conforming the kNN ellipsoid to the principal directions of the local data dispersion.

Differently from the work of D'haes et al. (2003), the modified query to node distance depends not only on the principal component but on intermediate covariance component which gives the the maximum contribution to the distance estimation, which improves the search on choosing the closest covariance cell (covariance quadtree's node) in a more refined criterion, avoiding deeper descents along the kNN search, typical of strictly binary trees.

Both the total performance and the search efficiency are approximately logarithmic per search if the number of neighbors K is relatively large ($K \sim 2\sqrt{N}$). The search efficiency was measured as the ratio of the number of definitive neighbors to the number of insertions C in the kNN list. The speedup with the number K of neighbors can be explained by the increase on the probability $\sim K/C$ that an insertion in the kNN list is definitive.

We tested the present method with a simple application of the gray-scale image retrieval from a noise data distribution, which revealed a good quality image even using only 10% of the data points.

The proposed technique might be extended to the case of any other positive-definite bilinear form. For instance the inertia tensor might be used to define the principal directions to divide a solid into small pieces separated from each other by normal cut planes. In this case, if the solid were represented by particles, as in the SPH scheme, the anisotropic search should be performed using the inverse inertia tensor in place of the covariance matrix modifying the Mahalanobis metric in the Algorithm 1.

Future applications of the present anisotropic kNN approach shall be concentrated in the investigation of SPH simulation of strongly compressive regimes, in which the adopted metric tensor presently used in the Mahalanobis distance must be replaced by the stress tensor.

ACKNOWLEDGMENTS

The authors are grateful to Fundação de Amparo à Pesquisa do Esatdo de São Paulo (FAPESP) for the partial support.

REFERENCES

Bishop C.M. *Neural Networks for Pattern Recognition*. Oxford University Press, Bookcraft (Bath) Ltd., Midsomer Norton, Avon, Great Britain, 1995.

Burschka D., Li M., Taylor R., and Hager G.D. Scale-Invariant Registration of Monocular Stereo Images to 3D Surface Models. In *Proceedings of 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems, September 28 - October 2, 2004, Sendai, Japan*, pages 2581–2586. IEEE, 2004.

D'haes W., van Dyck D., and Rodet X. Pca-based branch and bound search algorithms for computing k nearest neighbors. *Pattern Recogn. Lett.*, 24(9-10):1437–1451, 2003. ISSN 0167-8655. doi:http://dx.doi.org/10.1016/S0167-8655(02)00384-7.

Duda R.O. and Hart P.E. *Pattern Classification and Scene Analysis*. New York: Wiley & Sons, 1973.

Finkel R. and Bentley J. Quad trees: A data structure for retrieval on composite keys. *Acta Informatica*, 4 (1)(1):1âĂŞ9, 1974. doi:10.1007/BF00288933.

Gingold R.A. and Monaghan J.J. Smoothed particle hydrodynamics - Theory and application to non-spherical stars. *Monthly Noticies of the Royal Astronomical Society*, 181:375–389, 1977.

Jiang F., Oliveira M.S., and Sousa A.C. Sph simulation of transition to turbulence for planar shear flow subjected to a streamwise magnetic field. *Journal of Computational Physics*, 217:485–501, 2006. doi:10.1016/j.jcp.2006.01.009.

Jollife I.T. *Principal Component Analysis*. New York: Springer-Verlag, 1986.

Liu M.B., Liu G.R., and Lam K.Y. Adaptive smoothed particle hydrodynamics for high strain hydrodynamics with material strength. *Shock Waves*, 15(1):21–29, 2006.

Lucy L.B. A numerical approach to the testing of the fission hypothesis. *Astronomical Journal*, 82:1013–1024, 1977.

Ma S. and Ji C. Modeling video traffic in the wavelet domain. In *INFOCOM (1)*, pages 201–208. IEEE, 1998.

Mahalanobis P. On the generalized distance in statistics. *National Institute of Science in India*, 12:49–55, 1936.

Marinho E.P., Andreazza C.M., and Lépine J.R.D. SPH simulations of clumps formation by dissipative collisions of molecular clouds. II. Magnetic case. *Astronomy and Astrophysics*, 379:1123–1137, 2001. doi:10.1051/0004-6361:20011352.

Marinho E.P. and Lépine J.R.D. SPH simulations of clumps formation by dissipative collision of molecular clouds. I. Non magnetic case. *Astronomy and Astrophysics Supplement*, 142:165–179, 2000.

McNames J. A fast nearest-neighbor algorithm based on a principal axis search tree. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(9):964–976, 2001.

Minasny B., McBratney A.B., and Walvoort D. The variance quadtree algorithm: Use for spatial sampling design. *Computers & Geosciences*, 33:383âĂŞ392, 2007.

Mouratidis K., Papadias D., and Tao Y. A threshold-based algorithm for continuous monitoring of k nearest neighbors. *IEEE Transactions on Knowledge and Data Engineering*, 17(11):1451–1464, 2005. ISSN 1041-4347. doi:http://dx.doi.org/10.1109/TKDE.2005.172. Member-Spiridon Bakiras.

Owen J.M., Villumsen J.V., Shapiro P.R., and Martel H. Adaptive Smoothed Particle Hydrodynamics: Methodology. II. *Astrophysical Journal Supplement*, 116:155–+, 1998. doi:10.1086/313100.

Pelupessy F.I., Schaap W.E., and van de Weygaert R. Density estimators in particle hydrodynamics. dtfe versus regular sph. *Astronomy & Astrophysics*, 403:389–398, 2003. doi:10.1051/0004-6361:20030314.

Rehrmann V. and Priese L. Fast and robust segmentation of natural color scenes. In *ACCV (1)*, pages 598–606. 1998.

Ribeiro V.J., Riedi R.H., and Baraniuk R.G. Optimal sampling strategies for multiscale stochastic processes. In *Institute of Mathematical Statistics Lecture Notes - Monograph Series*, pages 1–29. IEEE, 2006.

Tran T.N., Wehrens R., and Buydens L.M. Knn-kernel density-based clustering for high-dimensional multivariate data. *Computational Statisticas & Data Analysis*, 51:513–525, 2006.

Xu R., Stansby P., and Laurence D. Accuracy and stability in incompressible sph (isph) based on the projection method and a new approach. *Journal of Computational Physics*, In Press, Accepted Manuscript:–, 2009. ISSN 0021-9991. doi:DOI:10.1016/j.jcp.2009.05.032.

Yu C., Sharma P., Meng W., and Qin Y. Database selection for processing k nearest neighbors queries in distributed environments. In *JCDL '01: Proceedings of the 1st ACM/IEEE-CS joint conference on Digital libraries*, pages 215–222. ACM Press, New York, NY, USA, 2001. ISBN 1-58113-345-6. doi:http://doi.acm.org/10.1145/379437.379504.