Z-TREES AND PEANO RASTERS FOR SCAN ADAPTIVE IMAGE PROCESSING

Bertrand Zavidovique[1] and Guna Seetharaman[2]
[1]Institute for Electronics Fundamentals, Bldg. 220
University of Paris XI, 91405 Cedex, FR
zavido@ief.u-psud.fr
[2]Dept of Electrical and Computer Engineering,
Air Force Institute of Technology, Dayton, OH 45433
guna@ieee.org

July 12, 2007.

Abstract

Given a specific image processing task and an image, its effectiveness is influenced by proximity driven properties captured in the image scan. One could exploit this effect and maximize the performance by adapting the scan and/or suitably adjusting the original operator. This two part sequel explores the balance between the effort required in scan adaptation and operator reformulation. The definition, basic properties and the application of Peano rasters and Z-trees are presented. Peano raster is a specific space filling curve generated using bit splicing of its grid coordinates. The Z- tree is a binary tree whose terminal nodes follow a Peano raster. A rotation operator is designed to swap specific grand children of a node in the Z-tree, which would effectively adapt the scan locally. Analytical expressions are presented to fully characterize the mapping between the exact coordinates of a pixel and its rank in the image scan before and after such rotations. The effectiveness of scan adaptation as a preprocessing step is illustrated in a number of applications, as we develop a framework of data-driven performance maximization of the underlying image processing task. This approach can be extended to 3-D images and higher dimensional grids as are the peano rasters and Z-trees. Part 2 of the sequel is focused on generalizing the scans made of 12-distinct motifs and thus the Z-tree into *-trees.

AMS Subject Classification: ...

Key Words and Phrases: Image Data-structure, Adaptive Pyramid Algorithms, Perceptual Grouping of Pixels.

1 INTRODUCTION

Multi dimensional data analysis is inherently sequential due to the operational nature of computing and communication devices used. Morton[11] is the first work to study various options for storing images and multidimensional data in the form of 1-D sequence. The widely employed *image raster* - left to right and top to bottom scan - has its origin governed by the operational nature of video cameras and video image displays. The width of the image must be known *a priori* and the mapping does not depend on its height. This asymmetrical nature makes the representation unsuitable for dynamically scalable images and unsuitable for direct manipulation by recursive operators. We establish *Peano scans* as versatile replacements for the raster scans. A Peano scan [12] does not explicitly depend on the width or the height of the image. It lends itself to the pyramidal or hierarchical processing of images. Also, it is scalable to higher dimensional images.

Applications involving pyramid representation of images [1, 7] have increased steadily over the last three decades. Two seminal works on applying the quad tree representation are found in image segmentation [4], and in computer graphics [1]. Though the quadtrees work directly in a 2-D representation, they are less flexible than binary trees whose storage complexity is roughly the same. Computer systems built with binary tree representation are much simpler to construct [8] than the quadtree based pyramid[9, 10] machines. In contrast, a binary tree is more versatile than the quadtree.

The Z-trees introduced in this paper are essentially binary trees. They permit a unified framework to represent, analyze and process two and higher dimensional data sets as a pyramid. The analysis can be made more effective if the scan is adaptive and content driven. The Z-trees enable computational assessment of the tradeoff between the scanning and adaptation in the context of a given image processing task. This entails two new operators called rotate and switch, to adapt the scan over a family of space filling curves.

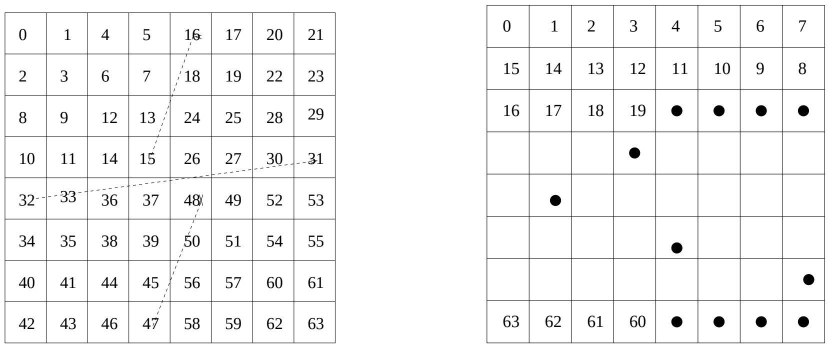
In section 2 we introduce the Peano-scan-index of a pixel computed from its coordinates and describe a hardware operation[5] for the same. We introduce a Z-tree representation of images in which each node in the underlying binary tree captures the interconnection and labelling of processors, or pixels, in a SPHINX like pyramidal architecture [3]. The computation of Peano functions, peano transformation - a vectorization in essence - and the binary tree representation of the vector, can all be extended to three and higher dimensional images. The $\mathcal{Z}-$tree contains all nodes of the quad-tree and octree of two and three dimensional images respectively. This is illustrated in section 3 where we also introduce a local transformation of the tree, called rotation, to modify the raster scan. Computations are provided to map the pixel coordinates on to its rank in the scan before and after rotation. Then, it is shown that the underlying data transformations bring forth the subsequent processing as pseudo inverses. Inspired by the above process, we proceed to generalize raster scans in section 4 by introducing another local transform of the tree called switch, which in turn permits adapting the scan over a complete family of space filling curves. Thus, the $\mathcal{Z}$-trees become *-trees. The pseudo inverse setting manifests once again. That is building a data structure by map products of: scan, tree construct, tree modify, then processing, and eventually inverting the data structure to rebuild the processed image from the 1D sequence. Note that several couples: *i.e.,* data-structure formation and image reconstruction as a pair, can be chained as well. Section 5 gives a few primary examples of the type of image processing enabled by Peano scans.

The ideas developed in this paper have been successfully applied to a set of problems[13, 14, 15, 16, 17]:1) The pipelined computation of orthogonal transform with four taps. 2) Image filtering with an without preprocessing in the Z-tree based framework. 3) Run length coding. 4) Content driven selection of adaptively sub-sampled exemplar and suitable zoom. 5) Image mixing if morphing style. 6) Concurrent edge/region detection; and, in a broad sense band pass filtering. And, 7) Content based image retreival using peano scans.

2 PEANO RASTER BASED TREE REPRESENTATIONS

2.1 Peano code, Peano-scan-index and Peano Transformed Images.

Points defined over a multi dimensional grid can be ordered in an unique sequence called a Peano scanned sequence. The Peano sequence of an 8×8 image is shown in Figure 1. It is continuous, recursive and spans over the entire image.



(a) (b)

Figure 1. The peano-scan of a 8×8 image grid, and a method of displaying the 64×1 scan are illustrated in (a) and (b) respectively.

Definition 1. Given $x \in \mathbb{Z}$,we define its Peano code $p_2(x)$ of spread factor 2 as, $p_2(x) := 2^{2\lfloor \log_2 x \rfloor} + p_2(x - 2^{\lfloor \log_2 x \rfloor}); p_2(1) := 1$, and $p_2(0) := 0$. Note, $p_2 : \mathbb{Z} \to \mathbb{Z}$, is one to one and invertible.

Let $x = \sum_{i=0}^{m} b_i(x)2^i, b_i(x) \in \{0, 1\}$, and $m = \lfloor \log_2 x \rfloor$ denote the binary representation of x. Then, the bit-spreading operator will produce, $p_2(x) = \sum_{i=0}^{m} b_i(x)2^{2i}$. It takes $O(\log_2 x)$ steps to compute $p_2(x)$ sequentially; however, a direct hardware implementation would take one clock-cycle.

Definition 2. Given a two dimensional vector, $(x, y) \in \mathbb{Z}^2$, typically the coordinates of a pixel in an image, its Peano scan index $P(x, y)$, is defined s.t., $P(x, y) = p_2(x) + 2\ p_2(y)$. The mapping $P : \mathbb{Z}^2 \to \mathbb{Z}$ is one-to-one, and fully invertible.

The definition above depicts the X-major setting. In contrast, the Y-major scanning would entail: $P\prime(x, y) = p_2(y) + 2\ p_2(x)$.

Definition 3. Given an image, $f(x, y), 0 \le x, y < \sqrt{n}$, then, its Peano Transform is an one-dimensional vector, $F[k]$, expressed as:

$$f(x, y) \quad \longrightarrow \quad F[k = P(x, y)]; \qquad 0 \le x, y < \sqrt{n}; 0 \le k < n.$$

More specifically, it is a unique sequence of all pixels. The coordinates (x, y) of a pixel and its index k within the sequence are related by $P : (x, y) \to k$. The above Peano transformation is an unique, one-to-one, and an invertible mapping. The inverse Peano transformation, $F(k) \to f(x, y)$ can be achieved by computing $P^{-1}(k) = (x, y)$. The $n \times 1$ vector $F[k]$ is best visualized in the form of a matrix as shown in Figure 2b, as suggested in Figure 1b.

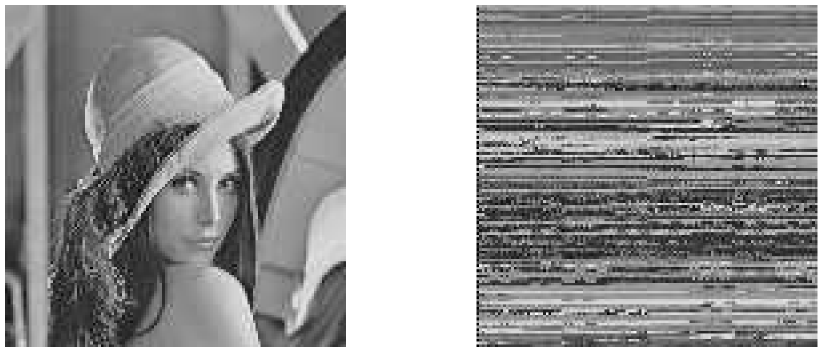

(a) (b)

Figure 2. The image shown in (a), is made of 128 × 128 pixels. Its peano transform shown in (b) is made of the 16384 × 1 pixels arranged in a specific order illustrated in Figure 1(b).

The computation of peano code, $p_2(x)$ and peano scan index P can be extended into higher dimensions. One could define, $p_3(x) = \sum_{i=0}^{m} b_i(x)2^{3i}$, and $p_d(x) = \sum_{i=0}^{m} b_i(x)2^{di}$, etc. Then, the Peano scan index of a three-dimensional grid-coordinate would be defined as: $P(x, y, z) = p_3(x) + 2\ p_3(y) + 4\ p_3(z)$. Note,

$p_3(x)$, will insert two blank bits between two consecutive bits of x. Also, the Peano scan index of a n-D grid location would be of the form: $P(x_1, x_2, \cdots, x_n) = p_n(x_1) + 2p_n(x_2) + 2^2 p_n(x_3) + \cdots 2^{n-1} p_n(x_n)$.

2.2 Compact Binary Trees

It is proposed to use $\mathcal{Z}$-trees as a new tool for representing images. It is a binary tree constructed based on the peano traversed data. A compact binary tree is used. Popular trees such as quadtrees and octrees are inherently embedded into $\mathcal{Z}$-trees. Some basic definitions pertaining to compact k-trees are presented before describing the construction of the $\mathcal{Z}$-tree.

In the context of tree based representation of images, the term *node* is used to denote an abstract region, usually a rectangle or a square in the image. A *k-tree* is a tree in which each non terminal node has at most k nodes as its children. A *complete* $k-tree$ is a tree in which each non terminal node has exactly k children. The number of terminal nodes at the base of a complete $k-tree$ is k^h, where, h is the height of the tree. Then, the number of internal nodes will be: $\frac{n-1}{k-1}$, where $n = k^h$.

Definition 4. A compact tree is made of a linear array $T = [t_1, t_2, \cdots]$, and an implicit hierarchy of the form: $H_k(i) = \{t_i, c_1(i), c_2(i), \cdots, c_k(i)\}$, such that, $c_j : t_i \to t_m; t_m \in \{t_0 = \phi\} \cup \{T\}$, where $m = (i-1)k+1+j$, if t_i is a non-terminal node, and $m = 0$, otherwise.

The rule H_k embeds a pointerless complete $k-tree$ onto the array T. More importantly, the array T depicts the exact *breadth-first traversed* sequence of nodes in the tree. The nodes, $t_i, 1 \le i \le \frac{n-1}{k-1}$, are internal nodes, and the nodes $t_j, n \le (k-1)j \le kn-1$ are terminal nodes.

Given a sequence $F(p), 0 \le p < 2^h; n = 2^h$ a compact binary tree T_F can be constructed by a bottom-up process as follows:

```
Initialize:   for $j = n$ to $2n-1$ do
              $t_j \leftarrow import(F(j-n))$
Construct:    for $i = (n-1)$ down to 1, do
                  $t_i := construct\_ancestor(t_{2i}, t_{2i+1})$
```

The *construct_ancestor* is essentially an aggregation operation. It would involve any filter on the "children", both linear or non linear ones, e.g. weighted mean, median, sup, inf etc. The abstract relations $parent(\cdot)$, $ancestors(\cdot)$ and, $descendents(\cdot)$ relate various elements within the tree. Given, $t_j \in T$ its ancestors can be defined as: $A_T(t_j) = \{t_a | a = \lfloor j/2^i \rfloor; i = 1, 2, \cdots, \lfloor \log_2 j \rfloor\}$. Also, its descendents at relative depth δ are: $D_T^\delta(t_j) = \{t_d | d = j2^\delta + \epsilon; 0 \le \epsilon < 2^\delta\}$, where, $1 \le \delta \le \log_2 n - \lfloor \log_2 j \rfloor$. Any modification of top-down nature at j would affect specific attributes of all descendent nodes $D_T(j)$. For example, breadth first labelling of the nodes in a tree is a top-down process. Change in the labelling order at a given level will impact the labels of all descendent nodes.

Definition 6. The *label* j of a node was implicitly defined above to be the position of the node in the tree. Thus, if $\ell(j) := j$, describes the labelling, it is expressed in a recursive form as follows:

$$\ell(j) = 2\ell\left(\left\lfloor \frac{j}{2} \right\rfloor\right) + j \ \mathbf{mod}\ 2; \quad \text{with} \quad \ell(1) := 1, \ \text{and } 1 \leq j < 2n.$$

We now merge the 1-D peano traversal with the compact binary tree, in defining the $\mathcal{Z}-$tree to represent images as follows.

Definition 7. The $\mathcal{Z}-$tree is compact binary tree built on the Peano raster defined earlier. Thus, $Z_f = T \cdot P \cdot f(x, y)$. An implicit labelling $\ell(t_j) := j$ would produce a labelled tree depicted in Figure 3.

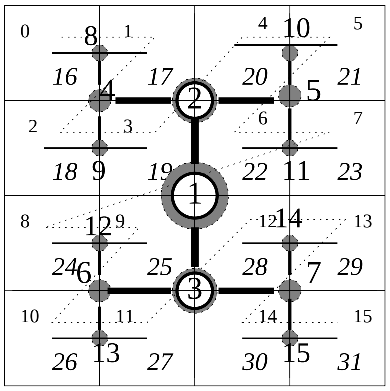


Figure 3. The $\mathcal{Z}-$tree of a 4×4 grid. Internal nodes are shown as labelled circles. Node 1 depicts the entire image, and nodes $2, 3$ depict upper and lower halves, while the nodes $4, \cdots, 7$ depict four quadrants etc. Nodes $16 \cdots 31$ are in fact pixels. The rank of each pixel in the peano scan is shown in small letters, and its position in the tree is shown in italics.

Other basic properties of a binary tree can be suitably exploited to process images. A *cutset* can be used to describe the extent of homogeneity in a specific instance. A segmented image is often represented by a minimal set of homogeneous nodes in T.

Definition 8. A subset $C \subset T$, is called a cutset of the tree T if it satisfies the following conditions:

(a) For each element $t_k \in C$, there exists no other element in C which would be on the path between t_k, and the root t_1.

(b) For each terminal node t in the tree, there exists an ancestor of t in the set C.

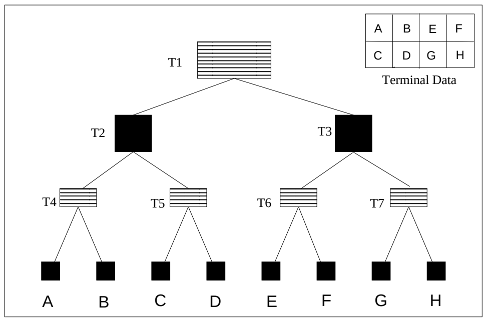
3 LOCAL TRANSFORMATIONS FOR TREE ADAPTATION

We now introduce operations that will maximize localized properties of one or more subtrees based on homogeneity, similarity of texture index, least or maximum differences etc, among the nodes at the base of each the subtree. For example, let $T = \{t_1, t_2, t_3, A, B, A, C\}$ describe a tree made of a 2×2 image. Let $T\prime = \{t_1, t_2, t_3, A, A, B, C\}$. It is observed that the cutset required to represent $T\prime$ is minimal. Thus, we are motivated to have a mechanism to permute, or locally altering the parent_of(.) relation, where it is desirable. The net impact is to permute the terminal level nodes of $T\prime$ given T.

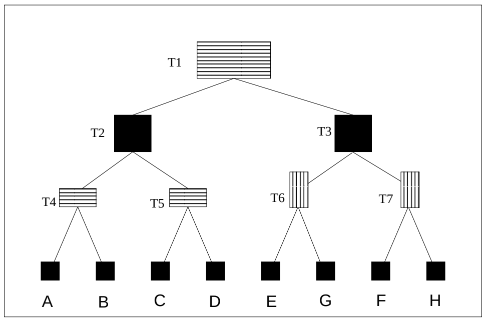
3.1 Rotation Operator: Adaptation of the Scan Orientation

The rotation operator will swap two grandchildren of a node in the binary tree, as defined below. For a square node, this operation will locally change the direction of the underlying scan from X major scan to a Y major scan.

Definition 9. The rotation operator R, on a node t_i, locally modifies the implicit hierarchy between its two children t_{2i}, t_{2i+1} and its four grand children $t_{4i}, t_{4i+1}, t_{4i+2}, t_{4i+3}$ respectively. The unrotated instance is represented by the sequence: $I_i = \{t_i, t_{2i}, t_{2i+1}, t_{4i}, t_{4i+1}, \quad t_{4i+2}, t_{4i+3}\}$. And, the rotated instance is given as: $I\prime_i = \{t_i, t_{2i}, t_{2i+1}, t_{4i}, t_{4i+2}, t_{4i+1}, t_{4i+3}\}$.



(3-a)



(3-b)

Figure 4. The binary tree of a 2×4 image, and the labels of each pixel are shown in (a). The exact configuration of the tree after a rotation $R(t_3)$ is displayed in (b).

To illustrate the above concepts, we consider a 2×4 image, made of 8 pixels, labelled, $A, B, \cdots, H$, as shown in the top right corner of Figure 3. The labelling follows the Peano-transformed order, introduced in section 2.1. A binary tree made of eight terminal nodes, $A \cdots, H$ and seven internal nodes, $t_1, \cdots, t_7$, is depicted in Figure 3a. And, Figure 3b shows the tree after it has been rotated at the node t_3. In this example, the local instance $I_3 = \{t_3, t_6, t_7, E, F, G, H\}$ has been modified by a rotation, $R_{(t_3)}$, into $I_3\prime = \{t_3, t_6, t_7, E, G, F, H\}$.

The rotation operator $R_{(i)}$, - is its inverse; thus, $R_{(i)} \cdot R_{(i)} \cdot T = T$. Also, local rotation at two distinct locations will not commute; *i.e.,* if, $i_1 \neq i_2$, then, $R_{(i_1)} \cdot R_{(i_2)} \ \cdot T \neq R_{(i_2)} \cdot R_{(i_1)} \ \cdot T$. However, they commute if neither i_1 nor i_2 is an ancestor of the other. Only then the rotations $R_{(i_1)}$ and $R_{(i_2)}$ can be executed in parallel, or in any order. An explicit precedence must be imposed in order to uniquely define multiple rotations and thus enable their unique inversion.

Definition 10. Given, $i_1 < i_2 \cdots < i_n$, then, by definition, $R_{(i_1, i_2, i_3, \cdots, i_n)} \ T = R_{(i_1)} \cdots R_{(i_n)} \ T$. And, the inverse of the composite rotation, $T\prime = R_{(i_1, \cdots, i_n)} \ T$ is expressed as: $T = R^{-1}_{(i_n)} \cdots \quad R^{-1}_{(i_1)} \ T\prime$.

3.2 Block Permutation of The Image

The ordered breadth first traversal of the original and rotated trees in Figure 3 would produce the sequences T and $T\prime$ respectively, where, $T = \{t_1, t_2, \cdots, t_7, A, B, C, D, E, F, G, H\}$, and, $T\prime = \{t_1, \cdots, t_7, A, B, C, D, E, G, F, H\}$. We observe that the image underlying $T\prime$ is made of the sequence: $\{A, B, C, D, E, G, F, H\}$, permitting two interpretations: 1) $T\prime$ bears a different scan over the input. And, 2) the image at the base of $T\prime$ is a decimated and shuffled version of the input

Definition 11. A cascaded operation of the form: $T^{-1}RT$, is defined as *Image Permutation*, where T represents the construction of the binary-tree from an image, and R the rotation vector applied on the tree.

The permutation operation as defined above follows a pseudo inverse form. Rotation is valid for any node in the $\mathcal{Z}-$tree which has four or more terminal descendents at the terminal level. It decimates the terminal-descendents into 4 subsequences, and block-swaps the second and third blocks. $T^{-1} \cdot R(*) \cdot T$, is rich enough for expressing textural patterns, perceptual grouping of pixels etc, used in image segmentation, illustrated below.

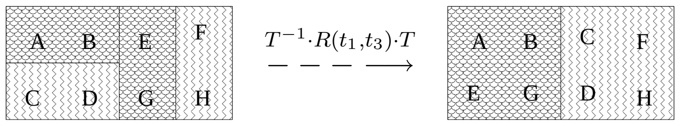


Figure 5. The effect of rotation is to consolidate blocks of similar pixel properties.

A textural pattern involving A, B, E, G and C, D, F, H is captured by $R(t_1, t_3)$. Also, the scan A, C, E, G and B, D, F, H will be characterized by: $R(t_1, t_2, t_3)$. Thus, multiple rotations applied to the subtree under a node facilitate a rich set of balanced partitions depicting various textural patterns.

The operation $T^{-1} \cdot R \cdot T$ does not span over the whole set of permutations of image pixels. R would shuffle selected sub-sequences of (length 2^*). For an image of size $\sqrt{n} \times \sqrt{n}$ pixels there are exactly $\frac{n}{2} - 1$ internal nodes which are valid operands for the rotation operator. Thus, there are at most $\frac{n}{2} - 1$ independent rotations which can be applied on the tree, up to $2^{0.5(n-2)}$ distinct permutations of the input image to be generated by the rotation operator.

3.3 Maximal Permutation: Localized Rotations Everywhere

It is useful to gain a geometric insight to the underlying permutations. We consider an extreme case where systematic rotations are performed at every node. A systematic methods is used to construct: $T^{-1} \cdot R(t_1, t_2 ..., t_{0.5n-1}) \cdot T$. The objective is to develop a closed form expression as well as labelling procedure to compute $T^{-1} \cdot R(\cdots) \cdot T$ after an arbitrary number of data driven rotations.

First, we point out that the scan underlying each square node in the $\mathcal{Z}-$tree, follows the Z order$\begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix}$ and, that of a rectangular node follows the И order$\begin{bmatrix} 0 & 2 \\ 1 & 3 \end{bmatrix}$. If a rotation is applied on each of these nodes, then, the scans become permuted into: И and Z respectively. Given a 2×4 block, up to three distinct rotations can be applied on its $\mathcal{Z}-$tree to maximally permute the scan as follows:

$$\begin{array}{cccc} 0 & 1 & 4 & 5 \\ 2 & 3 & 6 & 7 \end{array} \rightarrow \begin{array}{cccccc} 0 & \swarrow & 1 & 4 & \swarrow & 5 \\ 2 & \nearrow & 3 & 6 & \nearrow & 7 \end{array} \rightarrow \begin{array}{ccc} 0,2 & \swarrow & 4,6 \\ 1,3 & \nearrow & 5,7 \end{array} \rightarrow \begin{array}{cccc} 0 & 2 & 1 & 3 \\ 4 & 6 & 5 & 7 \end{array}$$

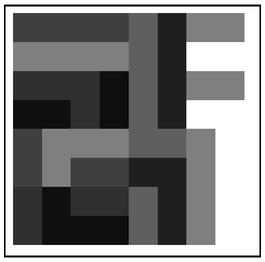
The compound rotation produces two 2×2 squares made of all "even" or "odd" scan indices. Then, we construct the scan corresponding to a 4×4 image, made of two such 2×4 blocks, having been rotated from level 2 and 3. Further rotation from level 4 will swap two 2×2 blocks, as shown below.

$$\begin{array}{ccc} \begin{array}{c} 0,2 \\ 4,6 \end{array} & & \begin{array}{c} 1,3 \\ 5,7 \end{array} \\ & \swarrow\!\!\nearrow & \\ \begin{array}{c} 8,10 \\ 12,14 \end{array} & & \begin{array}{c} 9,11 \\ 13,15 \end{array} \end{array} \rightarrow \begin{array}{cccc} 0 & 2 & 8 & 10 \\ 4 & 6 & 12 & 14 \\ 1 & 3 & 9 & 11 \\ 5 & 7 & 13 & 15 \end{array}$$

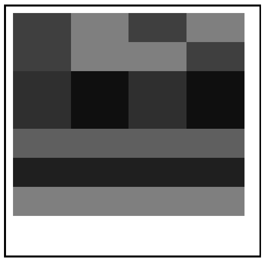
The result of applying rotation at all valid internal nodes can be summarized as follows. It produces two down-sampled version of the input, one made entirely of even columns and the other of odd columns. They also appear transposed in the output. In the case of rectangular images, they manifest as the left-half and the right-half of the output. In contrast, in square-shaped images, they manifest as the top-half and bottom-half of the output.

3.4 A Case for Data Driven Rotations

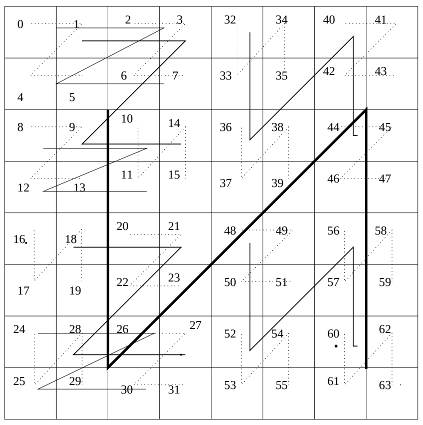
A test image of 8 pixels, its permuted version, and the iconic representation of several local rotations involved in the permutation are shown in Figure 6. The local decision to rotate seeks to align the gradient along the y direction. That is, if the x gradient is high in magnitude in a square shaped sub image, then a rotation is activated at the corresponding node in the $\mathcal{Z}-$tree.



(6a)



(6b)



(6c)

Figure 6. A specially designed 8×8 image shown in (a), and its $\mathcal{Z}-$tree in (b). The tree was rotated by the compound rotation: $R(1, 5, 7, 8, 9, 13, 19, 20, 22, 24, 26, 29, 30, 31)$ followed by an inverse peano transform to produce (c).

3.5 Direct computation of the permuted image

A closed form expression is available to express the scan after rotation. The transformation underlying $\mathsf{Z} \rightarrow \mathsf{N}$ is concisely expressed as $\phi : b \in \{0, 1, 2, 3\} \rightarrow b\prime \in \{0, 2, 1, 3\}$, where, $b\prime = \phi(b) = 2b - 3\lfloor \frac{b}{2} \rfloor$. In particular, when b is represented as a two bit number AB then $b\prime$ can be computed using logical operation ϕ_L, where, $\phi_L : \{A, B\} \rightarrow \{(A \textbf{ xor } B) \textbf{ xor } A, (A \textbf{ xor } B) \textbf{ xor } B\}$.

Consider, $R(t_i), 1 \leq j < \frac{n}{2}$. Let, $h = \log_2 n - \lfloor \log_2 j \rfloor$ be the relative height of the node t_j from the base. Let $\alpha = \lfloor h/2 \rfloor$ and $\beta = \lceil \frac{h}{2} \rceil$. Then, the node t_j

represents a sequence made of $\triangle = 2^h$ pixels, all originating from a $2^\alpha \times 2^\beta$ block. The block will be a square if h is even, or a rectangle otherwise. The scan index of the first pixel in the block is $k_j = j\triangle - n$. One can decompose the block into four quadrants – a square into four squares and a rectangle into four rectangles. The net effect of $R(t_j)$ is that all pixels in the second and the third sub-blocks are swapped intact. Let $q\prime$ be the new scan index of a pixel whose original scan index is q, s.t., $k_j \leq q, q\prime < k_j + \triangle$. Also, let, $q = k_j + b \cdot 4^{-1} \cdot \triangle + o$, where, $b = \lfloor \frac{4(q-k_j)}{\triangle} \rfloor$, and $o = (q - k_j)$ **mod** $(4^{-1}\triangle)$; $0 \leq o < 4^{-1}\triangle$. Then, $q\prime = k_j + 4^{-1}\triangle\ \phi(b) + o$. Thus, $q = k_j + 4^{-1}\triangle b + o \xrightarrow{R(t_j)} q\prime = k_j + 4^{-1}\triangle\ \phi(b) + o$. The rotation $R(t_j)$ causes pixel $(x, y) = P^{-1}(q)$ to be swapped with pixel $(x\prime, y\prime) = P^{-1}(q\prime)$.

```
let $n =$ Height $\times$ Width.{Assume that all are integer powers of 2.}
Construct the Tree.
let $Q_1[q] := q;\ \forall_q,\ 0 \leq q < n.$
for $h := 2$ to $\lfloor \log_2 n \rfloor$ do
begin
  $\triangle = 2^h;\ j_0 = \lfloor \frac{n}{\triangle} \rfloor;$
  for $j := j_0$ to $2j_0 - 1$ do
  begin
    $s := rotation\_state[j];$ {$s = 1$ if true 0 otherwise. }
    for $\delta := 0$ to $\triangle - 1$ do
    begin
      $q = j \cdot \triangle - n + \delta;$
      $q\prime = j \cdot \triangle - n + (4^{-1}\triangle) \cdot \phi(\lfloor 4\triangle^{-1}\delta \rfloor) + \delta$ mod $4^{-1}\triangle;$
      $Q_h[q] := s \cdot Q_{h-1}[q\prime] + (1 - s) \cdot Q_{h-1}[q];$
    end {$\delta \in [0 : \triangle - 1]$}
  end {$j \in [j_0 : 2j_0 - 1]$}
end; {$h \in [2 : \lfloor log_2 n \rfloor]$}
for all $x, y$ do compute
  $\hat{f}(x, y) = f(P^{-1}(Q_h[P(x, y)]));$
end.
```

The above algorithm requires $n \cdot (\log_2 n - 2)$ updates of the array $Q_*[\cdot]$, in the worst case, as it seeks to progressively compute the original scan index of each pixel as a function of its present position. In addition it requires at least two arrays Q_h and Q_{h-1}. The immediate swapping of blocks is essential to the implementation as depicted above. The complexity can be improved by a different implementation.

A deferred swapping mechanism would reduce the time and storage complexity for implementing compound rotations of the form: $R_{(i_1, i_2, i_3, \cdots, i_K)}$. Such a strategy will seek to know all the nodes at which a local rotation has been applied, before initiating the swapping of nodes. Although the data driven rotations are developed in a bottom up fashion in the tree, the tracing of the adapted scan is to be implemented in a top-down fashion. The process can be designed as a labelling process, defined below.

Let t_j be a node in the tree T. Let the label $\ell(j)$ refer to its rank within the breadth first traversed sequence of nodes. In a fully unrotated compact tree, by definition, $\ell(j) = j$. However, for a tree $T\prime$ that has been rotated several times, $\ell\prime(\cdot)$ can be computed as follows. Let $s[0 : n/2 - 1]$ be a state vector, s.t., $s[j] = 1$ if the tree has been rotated at the node t_j and $s[j] = 0$ otherwise. Also, let, $\ell\prime(j) = j; 1 \leq j \leq 3$. Define, $\lambda(j, s) = s\left[\lfloor j4^{-1}\rfloor\right] \cdot ((j \mod 4) - 3\lfloor 2^{-1}(j \mod 4)\rfloor)$. Then, compute $\ell\prime(j), 4 \leq j < 2n$ recursively as follows:

$$\ell\prime(j) := 2\ \ell\prime\left(\left\lfloor \frac{j}{2} \right\rfloor + \lambda(j, s)\right) + (j \mod 2) - \lambda(j, s).$$

Then, for any pixel (x, y) whose original scan index is $q = P(x, y)$, its scan index after the compound rotation is $q\prime(x, y) = \ell\prime(q+n) - n$. Thus, the permuted image can be constructed as: $\hat{f}(P^{-1}(q\prime(x, y))) = f(x, y)$.

The computational complexity of the deferred scan-computation approach is $O(n)$. It must be be emphasized that both the direct and deferred scan-computations involve a bottom up computation with regards to performing the rotation and constructing higher level internal nodes, as they consolidate pixels to blocks. However, in the direct approach, the terminal level descendents are updated each time a rotation is performed. In contrast, the deferred swapping strategy updates only the parent and children pointers at the node $t_j, t_{2j}, t_{2j+1}, t_{4j+\delta}, \delta = 0, 1, 2$ and 3, during the first phase, followed by a top-down labelling procedure in the second phase.

4 APPLICATION

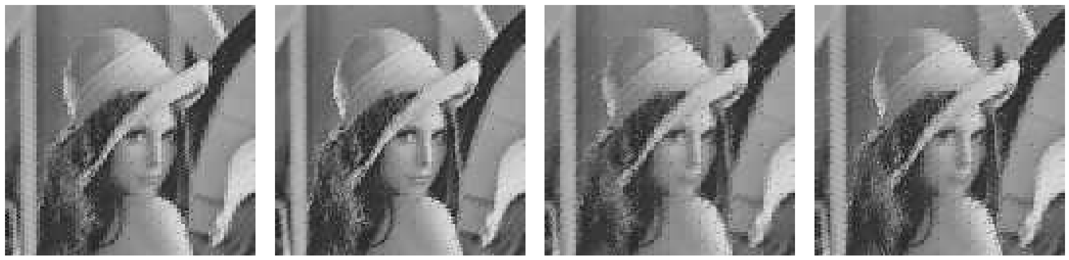
A variety of image processing applications lend themselves to a scan-adaptive image processing technique. This section highlights a few examples to illustrate the practicality and effectiveness of scan-adaptive techniques.

4.1 Image Filtering Applications

This sub-section deals with 1D implementation of image filtering. To maximize the effectiveness, seek to locally adapt the scan to increase (a) smoothness, and (b) contrast, in the context of low pass and high pass filtering applications. The pyramid facilitates a powerful mechanism for processing the 1-D sequence, including the pre and post processing steps such as the scan adaptations.

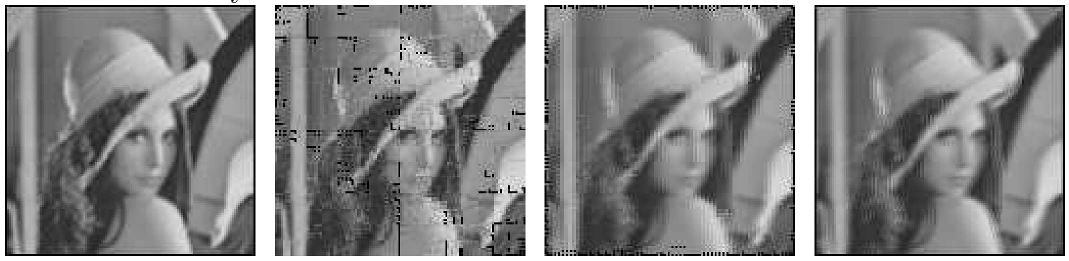
The Peano (Z-) scanned data of an image is subjected to a a set of 1D filters based on a 3x1 and 5x1 neighborhoods. We have considered local adaptations of the scan by applying "rotations" as a pre-processing step. Thus, for each image, we have four cases of pre-processing. 1) Without any rotations, 2) with local rotations seeking maximal local contrast, 3) with local rotations seeking maximal smoothness, and 4) local rotations applied always. This results in a net set of up to 8 experimental evaluation for a given filter. Thus for a smoothing filter, a simple averaging operator $h_3^s = (1/3, 1/3, 1/3)$, and $h_5^s = (1/5, 1/5, 1/5, 1/5, 1/5)$ accordingly.

We now present the results of such evaluations. The interpretation is summarized as follows. A salient feature that may be observed is: The 3x1 operators produce less artifacts than the 5x1s. However, they are less effective in accomplishing the desired filtering effect on non-adapted data. It confirms that rotation preconditions the data to enhance the effectiveness of filtering. For example, R_c enhances the effectiveness of an HPF and R_s that of a 3x1 LPF and so on. The same effect is even more pronounced in the case of filters with non-uniform values (weights) as: $\tilde{h}_3^s = (1/6, 2/3, 1/6)$.



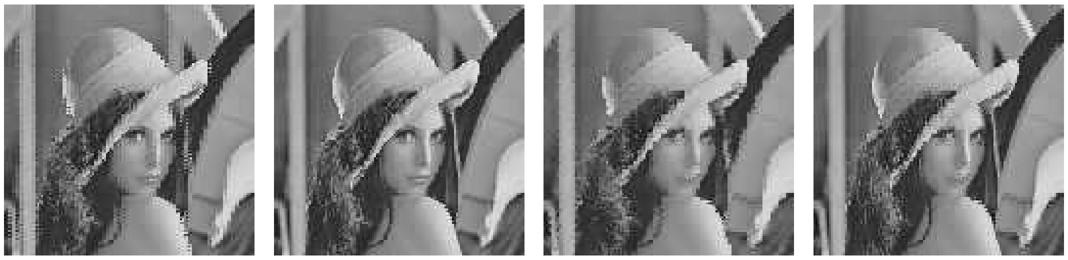
(a) (b) (c) (d)

Figure 7. Filters of the type, $g(x,y) = P^{-1} \cdot T^{-1} R_k^{-1} \cdot \mathcal{O} R_k \cdot T \cdot P \cdot f(x,y)$. The operator $\mathcal{O}$, in this instance is a `3x1` smoothing filter. Various rotations operators are used to locally adjust the tree: left to right – no rotation, rotation seeking contrast, rotation seeking smoothness, and rotation always.



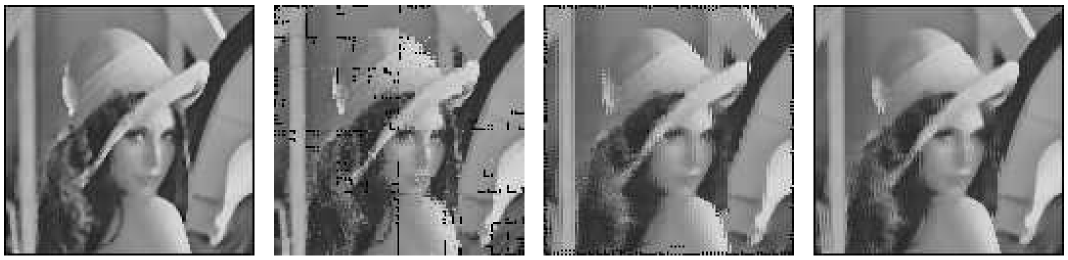
(a) (b) (c) (d)

Figure 8. Filters of the type, $g(x,y) = P^{-1} \cdot T^{-1} R_k^{-1} \cdot T \cdot P \cdot \mathcal{O} P^{-1} \cdot T^{-1} R_k \cdot T \cdot P \cdot f(x,y)$. The operator $\mathcal{O}$, in this instance is a `3x3` smoothing filter. Various rotations operators are used to locally adjust the tree: left to right – no rotation, rotation seeking contrast, rotation seeking smoothness, and rotation always.



(a) (b) (c) (d)

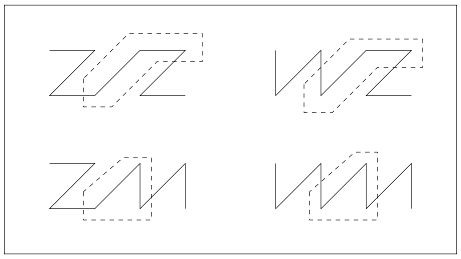
Figure 9. Filters of the type, $g(x,y) = P^{-1} \cdot T^{-1} R_k^{-1} \cdot \mathcal{O} R_k \cdot T \cdot P \cdot f(x,y)$. The operator $\mathcal{O}$, in this instance is a `3x1` median filter. Various rotations operators are used to locally adjust the tree: left to right – no rotation, rotation seeking contrast, rotation seeking smoothness, and rotation always.



(a) (b) (c) (d)

Figure 10. Filters of the type, $g(x,y) = P^{-1} \cdot T^{-1} R_k^{-1} \cdot T \cdot P \cdot \mathcal{O} P^{-1} \cdot T^{-1} R_k \cdot T \cdot P \cdot f(x,y)$. The operator $\mathcal{O}$, in this instance is a `3x3` median filter. Various rotations operators are used to locally adjust the tree: left to right – no rotation, rotation seeking contrast, rotation seeking smoothness, and rotation always.

The source of the artifact effect is made explicit on the canonic patterns: 1) For a 3x1 filter, note that among six results computed in that process, only one may be affected by rotating; and, 2) depending on if the alignment was made, it emphasizes the smoothing effect or contradicts otherwise.



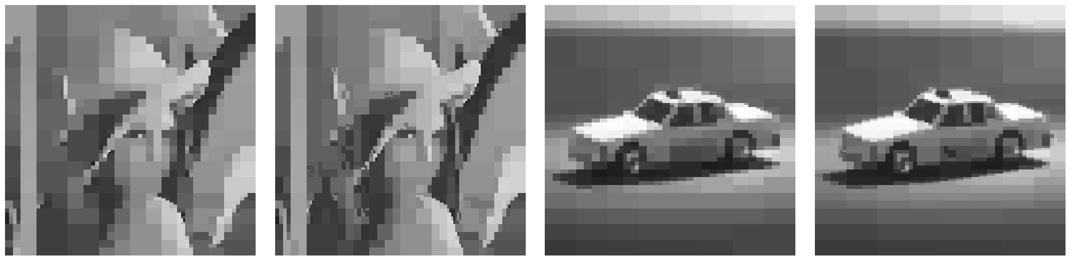
4.2 Run length coding

In this subsection we present an optimization algorithm designed to do run-length coding of a 1-D sequence, as an instance of image segmentation. A run-length coding procedure divides a 1-D sequence into a small number of homogeneous subsequences, and replaces each by a 3-tuple comprised of its location, length and the average pixel value. The global optimization would minimize the overall error introduced by approximating each subsequence by its average under the constraint of producing minimum number of segments. Once again, in a Z-tree, sub-sequences at the base amount to blocks in images.

A dynamic programming approach devised in [19] is used to solve this problem. It restricts itself to the subsequences whose length is an integer power of two, and specifically correspond to the nodes in a binary tree whose base is made of the input sequence. Then, the optimal partitioning of the input sequence into k partitions, $1 \leq k \leq n$ could be reduced to the problem of finding a cut set of k "dominant points" in the tree. Let $C(i, k)$, represent the total error that would result if the subsequence represented by t_i were to be segmented into k optimal segments and replaced by k mean values.

$$\begin{aligned} C(i,k) &= \min_{1 \leq p < k} C(2i, p) + C(2i+1, k-p), \text{ if } k > 1 \\ &= 0 \quad \text{if } k \geq 2^{h_i}, \text{ where } h_i = \log_2 n - \lfloor \log_2 i \rfloor \\ C(i,1) &= \textstyle\sum_q f^2(q) - 2^{-h_i} \left[\sum_q f(q) \right]^2 ; q \in [i2^{h_i} - n : (i+1)2^{h_i} - n - 1]. \end{aligned}$$

The performance depends critically on the match between the spatial orientation bias of the input data and the primary direction of the rasterization. Obviously, a simple image made of homogeneous vertical bars would produce better results on a vertically rasterized image. In any case, a number of consecutive lines of a fixed intensity would be identified as rectangles. For example, for an image of digitized text, it could compress large white spaces separating text lines and paragraphs.



(a) (b) (c) (d)

Figure 12. Filters of the type, $g(x, y) = P^{-1} \cdot T^{-1} \cdot R_k^{-1} \cdot \mathcal{O} \cdot R_k \cdot T \cdot P \cdot f(x, y)$. The operator $\mathcal{O}$, in this instance is a run-length coding process, permitting 512 runs. A rotations operator is used to locally adjust the tree: left to right – no rotation, and rotation seeking smoothness.

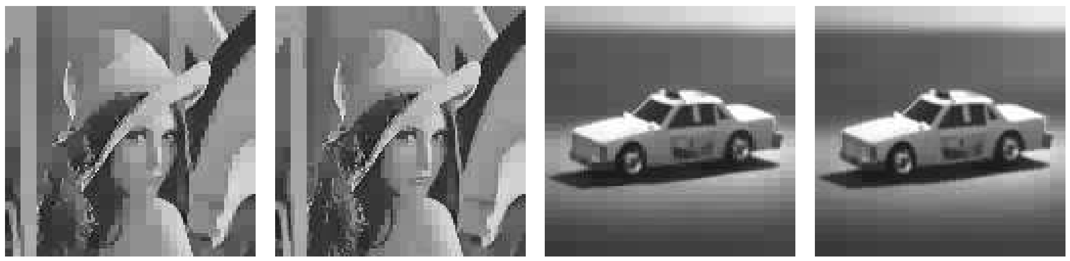

(a) (b) (c) (d)

Figure 13. Filters of the type, $g(x, y) = P^{-1} \cdot T^{-1} \cdot R_k^{-1} \cdot \mathcal{O} \cdot R_k \cdot T \cdot P \cdot f(x, y)$. The operator $\mathcal{O}$, in this instance is a run-length coding process, permitting 1536 runs. A rotations operator is used to locally adjust the tree: left to right – no rotation, and rotation seeking smoothness.

We compare 2 results of compression: 1) on the original Z-tree; and, 2) on the Z-tree where nodes have been rotated to align with the local gradient

4.3 Content Driven Subsampling and Feature Selective Zooming

A $\sqrt{n} \times \sqrt{n}$, image can be decimated into four sub images by applying rotations at the top $\frac{n}{2} - 1$ internal nodes, in two passes. Thus, sub-sampling refers to a sequence of maximal rotations across the entire tree. A set of $2^m \times 2^m$ similar images are generated, by repeating the process over $2m$ passes. The Z tree is now made of 2^{2m} sub trees, each representing a sub image. It is subjected to content driven rotation – seeking to minimize or maximize smoothness across the scan. Such adaptations are restricted to be within each sub image. Thus, each sub image has been transformed toward an unknown but common exemplar. The outcome provides a basis for quantifying the similarities between them. It is then possible to retain only k sub images, $k < 2^{2m}$, and suitably compute/duplicate the rest by a known strategy followed by an inverse of the sub sampling process. The last step is referred to as zooming.

The feature selective zooming effect comes from the selection of certain sub images which were subject to feature selective modification. Several variations have been tried for (1) selecting the k best sub images, and (2) the replacement process. In all cases, the same image-inverse-permutation is used and keeps track of the image structure. For example, (1) the second order statistics of each sub image, (2) root mean squared error with respect to a averaged sub-image, and (3) histogram similarities were used among others. A pixel of same relative coordinates in all sub-images is modelled as a Gaussian and the replacement is assisted with the estimated variance and mean.

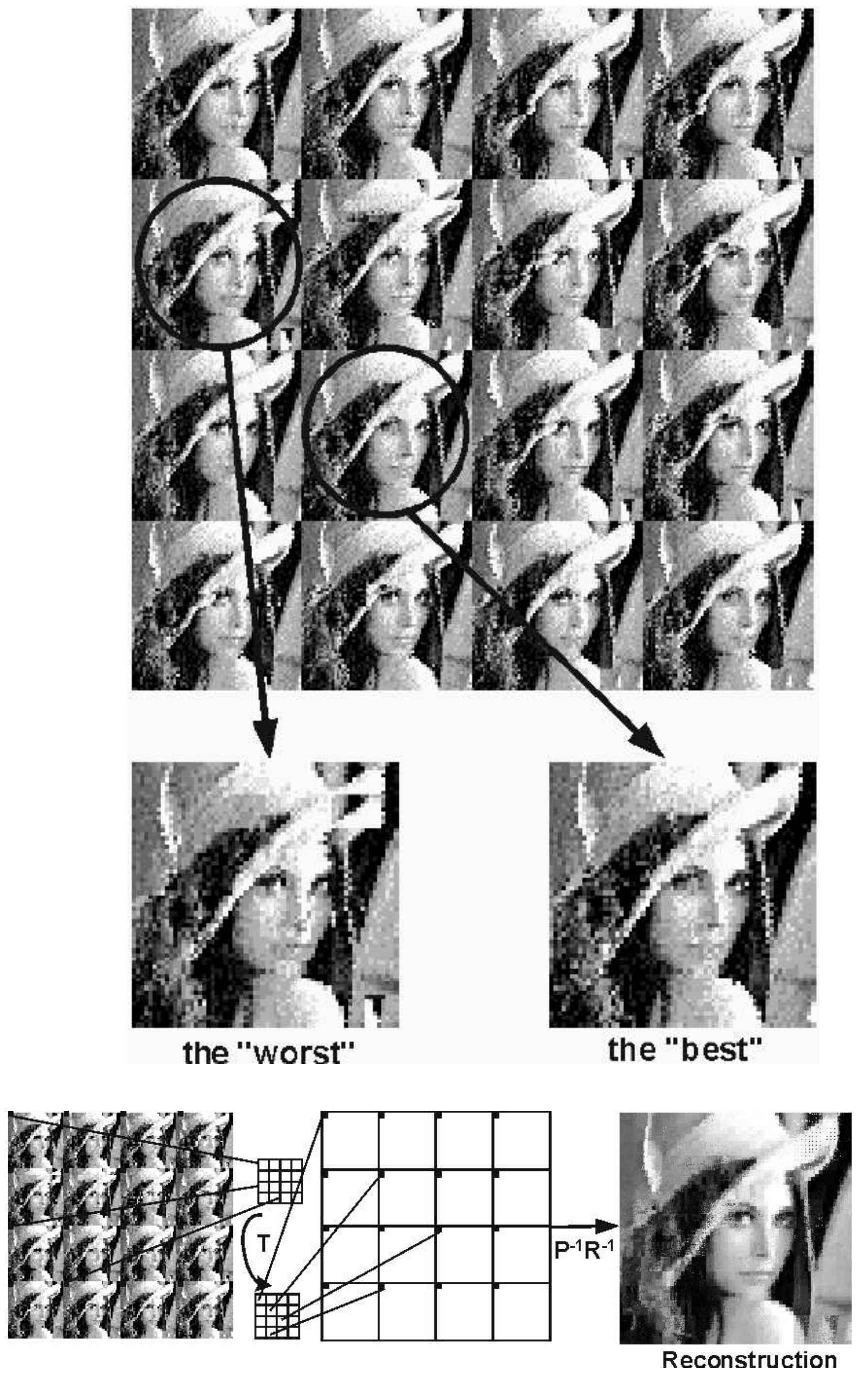


Figure 14. Up to k best subimages are selected. The rest are associated with one of the k candidates, for replacement. Thus, only the rotation vector and the k-best images, and $16-k$ associations must be retained. One approach is to randomly generate the pixel values, for the $16-k$ sub images, using a random-image-field model: $N(\sigma_s(x,y),\mu_s(x,y))$. The field is made of the ensemble of sub-images.

4.4 Concurrent Edge/Region Detection

A systematic process of maximal rotations m times create a group of $2^m \times 2^m$ sub images. It is possible to subject individual sub images to two or more distinct operators, and combine the output. This a content driven way of data fusion. For example, an edge detector (high pass filter) and a region growing operator (low pass filter) are applied on complementing sub sets of I_f. The results are combined in a controlled manner that is closely governed by the data.

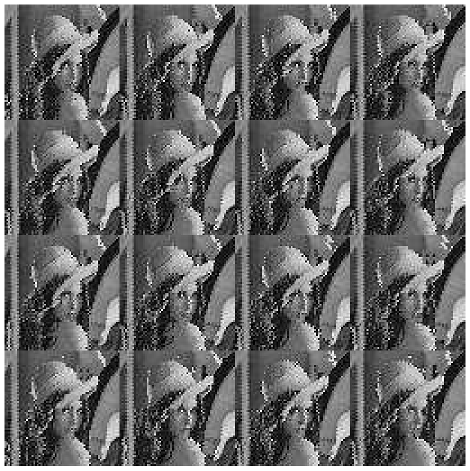
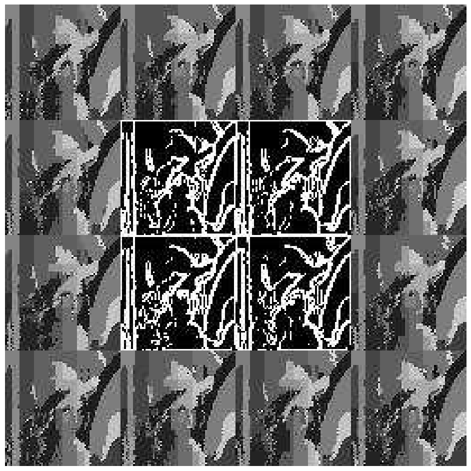
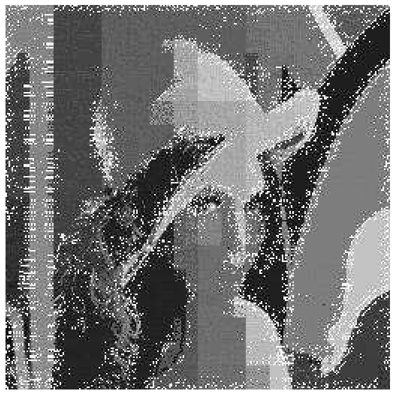

(a)

(b)

(c)

Figure 15. (a) The result of: $P^{-1} \cdot [T^{-1} \cdot R_c \cdot T]^4 \cdot P \cdot f(x, y)$ on `lena`. (b) Four of them subject to `sobel` and the rest to `LPF`.(c) The composite image, after appropriate reconstruction. Both high frquency and low-frequency signals emphasized.

Indeed, the sub images obtained are different from one another, as they depend on the conditional rotation at each node. To expand further, those sub images, which demonstrated good high frequency content were selected for edge detection, and the rest were subjected to region growing. Of course, the number of chosen sub images, and more generally the connectivity these sub images influence the quality of the reconstructed image.

4.5 Content Based Image Retrieval Enhanced by Peano Scans

Let the Z-tree of an image be rotated to locally maximize the smoothness. Then, we notice that the reconstructed image:*e*xhibits smoother variations along X-direction than the input image $f(x, y)$. Let, the input signal be analyzed in the Fourier domain. Then, $f(x, y) := (-1)^x$ would represent the signal of the highest frequency along the X axis. Then, the rotation operator would transform a signal, $f(x, y) := (-1)^x$ into $g(x, y) := (-1)^y$ if the rotation was applied locally at each node t_i, $\frac{n^2}{4} \leq i < n^2/2$, corresponding to each 2×2 block in the image. When the image is comprised of local features described by: $h_x(x, y) = (-1)^x$, and $h_y(x, y) = (-1)^y$, then, at each location the stronger of the two will determine the strength of G_{01}, and the weaker one will assume the position that of G_{10} term, on the 2×2 grid at that location. In effect, the term, $G_{0,\frac{N}{2}}$ will dominate the term $G_{\frac{N}{2},0}$ in the DFT of the final $N \times N$ image $g(x, y)$.

The net consequence of compound rotations is to redistribute the spectral energy in the frequency domain into a narrower band. Then, the $\frac{n^2}{2} - 1$ bit vector can be interpreted as a signature of the image, – a feature indicating the aggregate of local derivatives all across the image, and at various resolutions. And, $G(u, v)$ can be considered as the spectrum fully characterizing an equivalence class among a set of images, in which the rotation state-vector instantiates each member in the class. In effect, when we index an image database based on the Fourier spectrum of the individual images, the *optimal Peano scans* provide a mechanism to enhance the indexing process by giving rise to a notion of equivalence classes.

We have implemented a image retrieval system based on the peano scans using the well known MIT image-database collection. There has been a net increase in the retrieval performance by using the Z-scans over the raster scans. The rotation operation has further increased the number of relevant images selected in fixed-size retrieved set. The results are shown below.

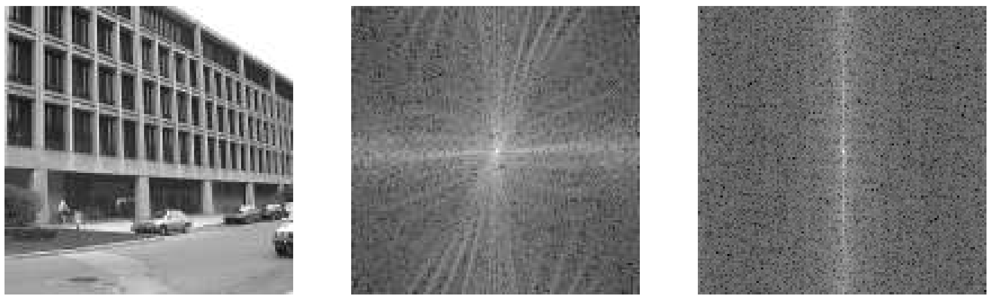
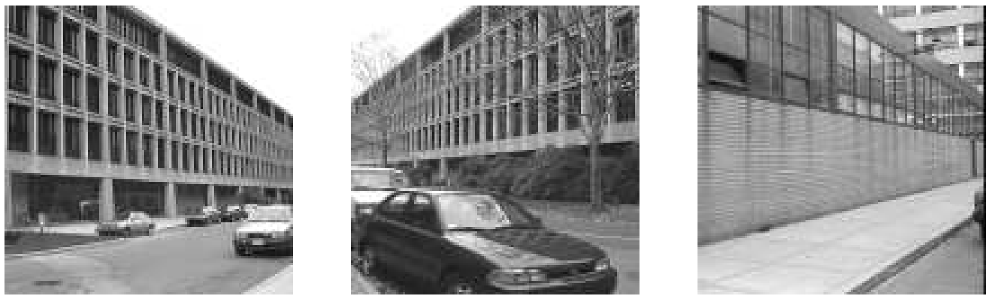
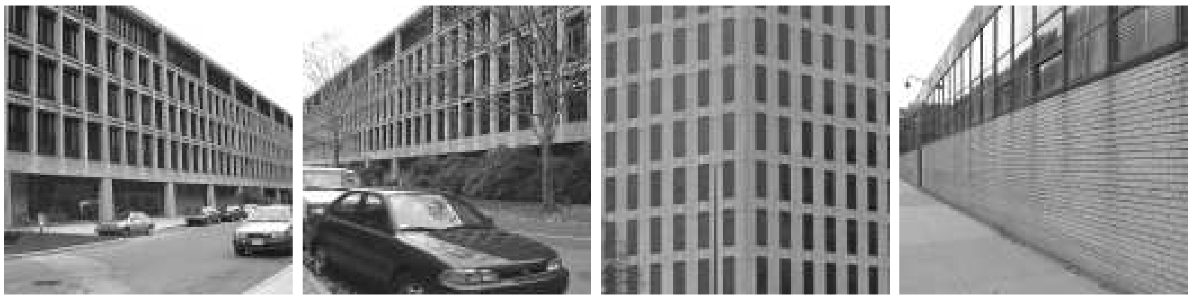

(a) (b) (c)

Figure 16. A sample image, and its 2-D Fourier spectrum, are shown in (a) and (b). The DFT of an image derived from the Z-tree after has been rotated is shown in (c).



(a) (b) (c)

Figure 17. The similar images returned by the image retrieval sub system, where the similarity is computed using the DFT of a simple raster based image representation.



(a) (b) (c) (d)

Figure 18. The similar images returned by the image retreival sub system, where the similarity is computed using the DFT of a, $g(x, y)$ computed after the Z-tree has been adaptively rotated seeking local smoothness as in: $g(x, y) = P^{-1} \cdot T^{-1} R_s^{-1} \cdot \mathcal{O} \cdot R_s \cdot T \cdot P \cdot f(x, y)$.

5 CONCLUSION

The key contribution of this paper is the permutation operation, $R \cdot T \cdot P$, which is used to pre-condition the data, before an image processing operation is applied on the input image. The $\mathcal{Z}-$tree of a digital image brings forth computations of the form:

$$g(x, y) \quad \longleftarrow \quad P^{-1}\, T^{-1}\, R^{-1}\, \mathcal{O}_1(\cdot)\; R\;\; T\;\; P\;\; f(x, y)$$

and,

$$g(x, y) \quad \longleftarrow \quad P^{-1}\, T^{-1}\, R^{-1}\, T\, P\, \mathcal{O}_2(\cdot)\; P^{-1}\, T^{-1}\, R\, T\;\; P\;\; f(x, y)$$

where, $\mathcal{O}$ is a class of image processing operations, the terms R, T, and P are as defined in section 3. Both $\mathcal{O}_1$, and $\mathcal{O}_2$ depict a 1-D or 2-D operator respectively.

The set $\mathcal{O}$ is made of operators that are most effective on images which exhibit some related maximal property, such as homogeneity. In the first form of computation, involving 1-D operators, the operations: RTP would pre permute the image to increase the effectiveness of $\mathcal{O}_1$, and $P^{-1}T^{-1}R^{-1}$ would post-permute the result (*i.e.,* shuffle back the data to its original order). In the second form of computation, involving 2-D operators, $P^{-1}\ T^{-1}\ R\ T\ \ P$ would produce a pre permuted 2-D image that is most suited for $\mathcal{O}_2$, and $P^{-1}\ T^{-1}\ R^{-1}\ T\ P$ would restore the original form from the result.

This forms the basis of a framework to design image content-driven algorithms. Indeed, more generally, one would assume that any image analysis exploits an image scan. This is enough to conjecture that segmentation in general could be formalized through a *universal decomposition*, the $CDEF$ map product:

$$G = C^{-1}D^{-1}E^{-1}\mathcal{F}(.)E\ D\ C$$

with:

G	a se$G$mentation process
C	a Coding scheme
D	a Data-structure
E	an Eulerian transformation (any rotation, permutation...)
$\mathcal{F}$	a Functional image operator

$\mathcal{F}$ is assumed to be most efficient on a class of images defined by the maximum of a given property, -homogeneity, directionality, decorrelation etc. C provides a pixel ordering (image traversal and its associated topology) that *globally* captures a related property,- compacity, isotropy, independence etc. D brings the the computational efficiency together with other computing properties like stability, accuracy etc. E, bound to C, conveys the final optimization for locally more adapted results. Then, after $\mathcal{F}$ transformed optimally prepared images, $(C \circ D \circ E)^{-1}$ reorganizes the data back into its original coherence.

Examples have been cited to illustrate the applicability of the above style of image processing. A tree based implementation of run-length coding and of 1-D filtering, served as a simple example of the first style given above. Where as any conventional segmentation is applicable for the second type of computations, we showed an adapted sub-sampling.

In the first case, for instance assuming (semi)textured images, the pre-permutation of the input image made it possible to: (i) represent homogenous regions in the image compactly in the binary tree; (ii) group pixels over squared and rectangular regions and organize the image as set of permutations; and, (iii) enable run-length coding, possibly perceptual grouping, through a minimal cutset of nodes in the binary tree.

The second example deals with elementary filtering. When filtering is involved in processing, – linear or non linear, it can be performed along the D

structure, which could be a 1-D scan or a 2-D block. The local adaptation of the data structuring part that supports decimation or selection, the data is preprocessed. Additional operational flexibility stems from the (C, E)couple – should the operator $\mathcal{F}$ require it.

In the third example, *zoom*, the $\mathcal{Z}-$tree is transformed through a chain of several $C^{-1}D^{-1}EDC$ to sub sample the image before $\mathcal{F}$ is applied on a several sets of homologous pixels.

In the three examples, coding C is Peano's, the network D is a tree, and the Eulerian transform E is a compound rotation and swap.

The goal is to significantly increase the efficiency of algorithms that would require large, compactly connected regions in the image, – segmentation, run length coding, textured-image clustering etc.

In this paper we propose an image representation called $\mathcal{Z}-$tree, a binary tree built on a Peano traversed image. The proposed representation permits simplification of the design and analysis of divide and conquer algorithms for two and three dimensional images to the same difficulty that of 1-D signals. It brings forth image segmentation operation under the pseudo-inverse form.

From an application stand point, angle, the tree can be constructed in real-time. And, the ability to embed a compact binary tree will thus enhance the simulation of parallel algorithms on some fast single CPU machines. We are currently in the process of expanding the versatility of local adaptations of the Peano traversal over the 2×2 motif. The resulting family of scans is made of 12 distinct motifs, leading to an expanded framework called the $*-$trees. New results on $*-$trees will be published in a companion paper.

6 ACKNOWLEDGEMENT

This work was made possible by a grant from the Centre National Recherche Scientifique. The work was carried in parts while the authors were visiting each other departments on their sabbatical leaves, at IEF, University of Paris XI, and CACS, University of Louisiana at Lafayette. The authors like to acknowledge Guruprasad Shivram, Samia Bouchafa, and Nitin Jhanwar for implementing the algorithms. Also, we acknowledge the technical contribution from Prof. Subhasis Chaudhuri with regards to the content based image retreival.

References

[1] John E. Warnock. A Hidden Surface-Algorithm for Computer Generated Halftone Images. *University of Utah, Computer Science Department*, TR 4-15, 1969.

[2] A. Klinger. Patterns and search statistics. In *Optimizing methods in statistics. (Ed) J. Rustagi et.al. Academic Press, New York.*, pages 303–337, 1971.

[3] A. Klinger and C. Dyer. Experiments in Picture Representation using regular decomposition. *Computer Graphics and Image Processing*, CGIP 5(1), 1976.

[4] S L. Horowitz and Theo Pavlidis. Picture Segmentation by a Tree Traversal Algorithm. *Journal of Assoc. for Computing Machinary*, JACM-23:pp. 368–388, 1976.

[5] H. Samet. Region representation: quadtrees from binary arrays. *Computer Graphics and Image Processing*, CGIP-13(1), 1980.

[6] H. Samet. An algorirthm for converting rasters to quadtrees. *IEEE Tranns. on Pattern Analysis and Machine Intell*, 3(1), 1981.

[7] P. J. Burt and E. H. Adelson. The Laplacian Pyramid as a Compact Image Code. *IEEE Trans. on Communications*, COMM-31(4):532–540, April 1983.

[8] Ph. Clermont, A. Mérigot, J. C. Roussel, and B. Zavidovique. Parallel implementation of a region growing algorithm on a pyramid machine. In Virginio Cantoni et al., editors, *Progress in Image Analysis and Pocessing*, pages 721–729. World Scientific, 1990.

[9] V. Cantoni and S. Leviladi, editors. *Pyramidal Systems for Computer Vision.* Berlin, FRG: Springer Verlag, 1986.

[10] R. Miller and Q. F. Stout. *Parallel Algorithms for Regular Architectures: Meshes and Pyramids,.* MIT Press, Cambridge, MA, 1996.

[11] G. M. Morton. A computer oriented geodetic data base and a new technique in file sequencing. *IBM Technical Memo*, 1966.

[12] G. Peano. Sur une courbe qui remplit toute une aire plaine. *Mathematische Annalen*, 36:157–160, 1890.

[13] G. Seetharaman and B. Zavidovique. Image Processing in a Tree of Peano Coded Images. In *Proceedings of The IEEE Workshop on Computer Architecture for Machine Perception. Cambridge MA*, 1997.

[14] N. Jhanwar, S. Chaudhuri, G. Seetharaman, and B. Zavidovique. Content based image retrieval using optimal peano scans. In *International Conference for Pattern Recognition. Aug 2002*, 2002.

[15] G. Seetharaman and B. Zavidovique. Concurrent Edge/Region Detection Using Peano Scans. In *International Conference on Image Analysis and Processing. IAPR ICIAP-2001. Palermo Italy*, Sep 2001.

[16] S. Bouchafa, G. Seetharaman, and B. Zavidovique. Large Image Decimation and Reconstruction Using Peano Scans. In *IASTED International Conference pn Computer Graphics and Image Processing. Hawaii.*, August 2001.

[17] G. Seetharaman and B. Zavidovique. Z-trees: Adaptive Pyramid Algorithms for Image Segmentation. In *Proc. of the IEEE International Conference on Image Processing. ICIP98, Chicago, IL.*, October 1998.

[18] J. O. Eklundh. A Fast Computer Method for Matrix Transposing. *IEEE Trans. on Computers*, C-21(7):801–803, July 1971.

[19] G. Shivaram and G. Seetharaman. Data Compression of Discrete Sequence: A Tree Based Approach Using Dynamic Programming. *Technical Report.* Center for Advanced Computer Studies, University of Louisiana at Lafayette, TR-02-1997-CACS, 1997.