Representation, Optimization and Compression of High Dimensional Data

Krasimir Kolarov
Interval Research Corp., 1801-C Page Mill Road, Palo Alto, CA 94304
E-mail: kolarov@interval.com, Web: http://www.interval.com/~kolarov

ABSTRACT

The exponential growth of available computing power opens the door for new capabilities in modeling the objects in the high dimensional real world that we live in. Traditionally we have manipulated data defined on simple entities such as the real line (e.g. audio), a rectangle in the plane (e.g. images) or a three-dimensional open-ended box (e.g. video - sequence of planar images). Today graphics cards and accelerators, powerful processors and cheap memory allows us to interact with complex 3D and higher dimensional objects. We need to build mathematical representations, algorithms and software that will allow us to easily represent, compress and manipulate such objects.

In this paper we will summarize some algorithms designed to that goal and used in the areas of robotics, computer graphics, 3D haptics (touch interaction with the environment) and data compression,. Those algorithms use novel methods in evolutionary computation, motion planning and geometric modeling, wavelets and embedded signal coding.

Part I INTRODUCTION

We are surrounded by objects and signals that are represented in high-dimensional spaces. Modeling of natural tasks lead to high-dimensional representations. We would like to use mathematical ideas for dealing with the inherent complexity of high-dimensional data.

There is no one method that can solve all problems, instead we will analyze the problems and try to characterize the potential methods for solution as appropriate for the problems they solve. Different algorithms and approaches work best for different goals. We will introduce and illustrate the working of several such approaches without making any claims for generality. The methods that we are interested in do not even have to work fully automatically. In a lot of cases human interaction leads to better, faster and cheaper solution, in particular in the robotics area.

There are several options for trying to solve problems defined in high-dimensional spaces. We can:
- use analogy from nature (model using Evolutionary Computation);
- use higher mathematical techniques (for example topology in wavelets and motion planning);
- use combination of the approaches above.

In what follows we will describe the application of those approaches in several areas. In particular in Part II we will introduce a simple evolutionary computation model. We will illustrate its performance in simulation and analyze theoretically its features. Part III will illustrate the need of high-dimensional modeling in the area of robotics, and in particular in robotics motion planning and manipulator design. An approach that have been used for high-dimensional planning in robotics involves the application of algebraic topology and geometry. We have also used that approach in building compact representations of signals occurring in the real world. Part IV describes some of our results in that area. It also illustrates the approach in the case of compression of data defined on the surface of 3 dimensional objects. The last part summarizes the presented ideas and points out other possible applications.

The results outlined in this paper have been described in detail in several previous publications and the reader is encouraged to consult the included bibliography if interested.

Part II EVOLUTIONARY MODELING

In this section we will use ideas from biology and population genetics to solve problems in computing. We will introduce simple genetics operations (recombination and mutation), start the system that we are trying to model from a random initial state; and let it evolve using those operations. At every iteration we will check how well the model matches the desired performance (the "fitness function" of the system). We will stop when there is no more change occurring or when the goal has been achieved.

2.1 Problem Formulation.

Evolutionary Algorithm (EA) is a heuristic computation procedure that is inspired by the evolutionary theory. It uses operators like selective reproduction, crossover (recombination) and mutation for search and optimization. The elements of the system, on which these operators are applied, are represented by strings of bits, where each bit can take finite number of values (in our case two - 0 and 1). The set of all elements (individuals) at any given time constitute the current population. Every individual in the population has an associated **fitness** in the environment (corresponding to the objective function we want to optimize) and the problem is to find the individual with the highest fitness.

The stochastic character of the EA operators is one of the reasons why it is difficult to explain that it works well for certain problems, or to predict whether the same approach will work in some other problem. There have been very few attempts to theoretically analyze the dynamics of interaction between the different operators and the performance of EA (e.g. Holland's *building blocks* and *schema* [7]).

For an EA analysis the fitness function is especially important and things are very complicated when it is not an explicit function, it varies with time or it is evaluated at run time only. Thus an interesting question is to analyze the importance of different **fitness regimes** for the viability of a GA system and to include the role of dynamics in this analysis. The rest is mechanics and parameter optimization. When analyzed, it models in high dimensional spaces.

In [9] and [10] we performed experiments with several different fitness landscapes. We ran a series of fixed number of **cases** (100), where the criterion for termination was **fixation** of the population (i.e. the case when all individuals in the population had exactly the same genotype). Note that this criterion is more characteristic for a population genetic analysis. In a typical EA system, the simulation runs until the first individual with an optimal fitness value appears. In nature populations of organisms do not stop their evolution after "the best" individual has appeared. In an effort from each individual to survive and reproduce, the next generation is formed. The best individual that is created at some generation might disappear later as a result of the interaction within the system. Thus we study the dynamics of interaction and look at the evolution process as an **adaptation**, rather than an **optimization**.

2.2 Results for suboptimal fixation of populations

Our goal was to compute the rate of fixation on sub-optimal fitness values and to analyze the mechanism of fixation. As a measure of the fixation level we use the number of cases (out of a 100) that fix on a genotype with a fitness different than the optimal one. This measure is similar in nature to the notion of genetic drift in population genetics [22]. In general in finite population models there is always drift due to the statistical nature of the process of sampling that produces offsprings from the parental types. When genetic drift dominates the selection pressure, the population may fix on a genotype with fitness different from the optimal. In our case fixation may occur not only due to random sampling error but also due to the existence of local optima in the fitness landscape. The initial population is selected at random with equal probability of 0's and 1's. Our model has fixed rates of recombination and mutation. We use single (one-point) recombination with the break point selected at random, uniformly across the chromosome. The offspring are subjected to mutation and selection. New offspring are accumulated until the fixed population size is reached. At that time we have formed the new generation and the current one becomes its parental generation. Thus we do not have explicit elitism - new generation is entirely formed by applying the genetic operators to the previous one.

We would like to point out that this method of constructing the offspring generation where every member of the new population is added as a result of individual "selection" after application of the genetic operators to the previous generation, is a characteristic of population genetic analysis [1]. In a typical EA analysis the individuals in the next population are generated as a result of "group" competition, i.e. their fitness is compared against each other in order to select the offspring. In our case an individual is added to the new generation if it is selected at random and if it is successfully recombined and mutated (based on fixed Recombination and Mutation Rates). In our case, the first generation (after the initial random one) is the one that takes the most amount of computing time. Typically in that generation a significant number of good individuals are created, after which it takes fewer generations overall for the whole population to fixate.

Since every individual in the population is represented by a string of length 20 bits, we can consider the problem as searching and optimization in 20 dimensional space. Analysis becomes even more complicated when we allow for diploid genotypes (individuals with two strings of bits each).

Some representative results from [10] are illustrated in figures below. Figure 1 describes the suboptimal fixation for given type of selection μ (assuming the fitness function for an individual with *i* number of ones is the Gaussian $F(i) = e^{-\frac{(i-\mu)^2}{2\sigma^2}}$). We can see that the increase of the Population size and the increase of the selection strength (smaller σ) lead to a decrease in the rate of "wrong" (suboptimal) fixation. Similarly in [10] we show that the time to fixation increases with the size of the population and with the inverse steepness of the fitness function. Stabilizing selection take longer to fixate than strong directional selection even though it produces less suboptimal fixations.

Figure 2 shows that the number of generations to fixation follows lognormal distribution (similar to the one for the time to extinction in population genetics [22]). Another interesting conclusion in [10] is that diversity (described by the heterozigocity shown in Figure 3) is maintained longer in populations with smaller recombination values.

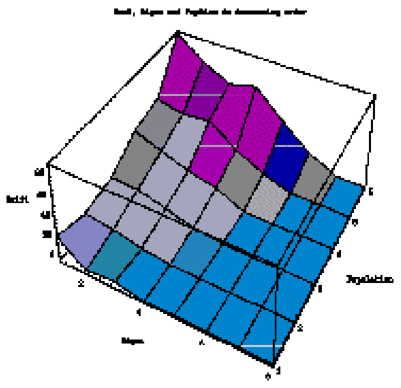
Figure 1. Suboptimal fixation for given selection type μ .

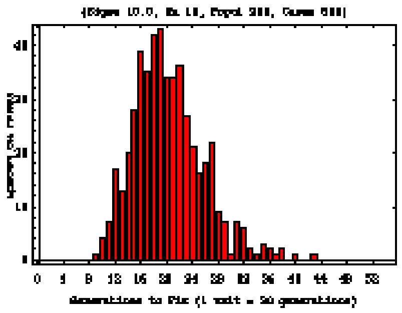

Figure 2. Distribution of the generations to fixation.

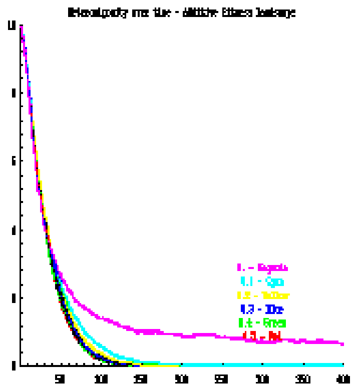
Figure 3. Heterozigocity as a measure for diversity.

2.3 Theoretical analysis for large populations

We can also perform theoretical analysis of the outlined results (see also [9]). Let us denote with *F(s)* the value of the fitness function for an individual with *s* number of 1's. With $\varpi_k(s)$ we denote the expected number of individuals with fitness *F(s)* in the population at generation *k*. *N* denotes the population size and *l* is the length of the genotype (in our examples *N = 100* and *l = 20*). Because we always keep constant population size, we have:

$$\sum_{s=0}^{l} \varpi_k(s) = N \text{ for every } k \qquad (1)$$

The average expected fitness of the population at generation *k*, is:

$$\Phi_k = \frac{1}{N}\sum_{s=0}^{l} F(s)\varpi_k(s) \qquad \text{for every } k \qquad (2)$$

We want to find a recursive relation describing the expected number of individuals with fitness *F(s)* at generation *k+1* in terms of the number of individuals with different fitness at generation *k*.
For simplicity let us first consider the case with no recombination, i.e. the recombination rate $r = 0$. We can derive:

$$\varpi_{k+1}(s) = \frac{F(s)}{\Phi_k}\varpi_k(s) \qquad (3)$$

We can also derive a recursive relationship for the expected average fitness of the new generation in terms of the data from the parental generation:

$$\Phi_{k+1} = \frac{1}{N\Phi_k}\sum_{s=0}^{l} F^2(s)\varpi_k(s) \qquad (4)$$

We will outline the proof of a simple geometrical conjecture (see [9]) for expected direction of fixation. Let us denote with p_I the initial probability of 1's in the first randomly generated population. In our discussion so far we have assumed that every allele (bit) for each individual in the initial population has equal chances (0.5) of being a 1 or a 0. In general however we can choose different distribution of 1's and 0's and we can vary that using p_I. Let us also denote with *X* the largest integer smaller than $p_I.l$ (i.e. $X = \lceil p_I . l \rceil$). Then the areas under the fitness curve scaled by the initial distribution to the left and to the right of point *X* are given by:

$$\Phi_0^+ = \frac{1}{N}\sum_{s=0}^{X} F(s)\varpi_0(s) \qquad \text{and} \qquad \Phi_0^- = \frac{1}{N}\sum_{s=X}^{l} F(s)\varpi_0(s) \qquad (5)$$

Thus the condition for fixating to the left of the mean of the initial distribution can be expressed theoretically as $\Phi_0^+ > \Phi_0^-$. However using relationships (2) and (4) we can show that this condition is equivalent to the condition:

$$\sum_{s=0}^{X}\varpi_1(s) > \sum_{s=X}^{l}\varpi_1(s) \qquad (6)$$

In other words if the scaled area to the left is larger than the one to the right, in the next generation there will be larger expected number of individuals whose fitness is to the left than those to the right. Using (6) by induction for future generations we can prove that the population is expected to fixate on average to an individual with number of 1's to the left of the mean of the initial population *X*.

The same conjecture is true in the general case when the Recombination Rate $r \neq 0$. In that case:

$$\varpi_{k+1}(s) = N[r.\mathrm{P_r}(s) + (1-r)\frac{1}{N}\varpi_k(s)]\frac{F(s)}{\Phi_k} \qquad (7)$$

where

$$P_r(s) = \frac{1}{(l+1)N^2}\sum_{x=0}^{s}\sum_{z=x}^{l}\sum_{p=s-x}^{l}[\sum_{y=0}^{l}\frac{\binom{y}{x}\binom{l-y}{z-x}\binom{l-y}{s-x}\binom{y}{p-s+x}}{\binom{l}{z}\binom{l}{p}}].\omega_k(z)\omega_k(p) \qquad (8)$$

We can add the coefficients for terms that have permuted indices and arrange the resulting coefficients in lower triangular matrices where each entry r_{ij} of the matrix $R_{k+1}(s)$ corresponds to the coefficient in (8) in front of the term $\omega_k(i)\omega_k(j)$.

The next formulae illustrate these matrices for the case of ***l=2.***

$$R_{k+1}(0) = \begin{pmatrix} 3 & & \\ 3 & 1/4 & \\ 2 & 0 & 0 \end{pmatrix}, \quad R_{k+1}(1) = \begin{pmatrix} 0 & & \\ 3 & 5/2 & \\ 2 & 3 & 0 \end{pmatrix}, \quad R_{k+1}(0) = \begin{pmatrix} 0 & & \\ 0 & 1/4 & \\ 2 & 3 & 3 \end{pmatrix} \qquad (9)$$

Those coefficients correspond to the following arrangement:

$$\begin{array}{lll} \omega_k(0)\omega_k(0) & & \\ \omega_k(0)\omega_k(1) & \omega_k(1)\omega_k(1) & \\ \omega_k(0)\omega_k(2) & \omega_k(1)\omega_k(2) & \omega_k(2)\omega_k(2) \end{array} \qquad (10)$$

As we can see from (8) and (9) with the increase of *s* we get less representation in the R-matrices of the $\omega_k(z)$ for $z < s$ and stronger representation of those with $z > s$. At the same time if $\Phi_0^+ > \Phi_0^-$ then higher weight is given to the individuals with fitness less than the mean of the initial population *X*. Because those individuals are more represented in the R-matrices to the left of *X*, that means that in the next generation those individuals will be even more represented and thus by induction the population will be moving toward a fixation on the left of *X*. By analogy if $\Phi_0^+ < \Phi_0^-$ the individuals to the right of *X* get more weight and representation than that is the direction of fixation.

For larger *l* we can clearly see the structure of this argument because for a given *X*, all the R-matrices for $s > X$ will have upper left rectangles of zeros. Accordingly the R-matrices for $s < X$ will have zeroes in the bottom right rows. These properties confirm the validity of our conclusion for the general case.

This conjecture is an attempt of using mathematical theory to analyze complicated high-dimensional systems. In EC such theory was begun by John Holland's schema work [7], and continued by Vose [29] and others.

An alternative way to analyze complex problems is to invoke sophisticated mathematics, for example algebraic and differential topology. In order to better understand that underlying topology, we will first describe two practical applications and approaches.

Part III HIGH-DIMENSIONAL PLANNING IN ROBOTICS

3.1 Motion planning and design in complex environments

The first example has to do with motion and manipulation in the physical world. How do people avoid obstacles and perform complex tasks? It is a sophisticated coordination of brain activity with sensory inputs - vision, sound, touch and smell. However there are a number of activities that are unpleasant, dangerous or too expensive to be performed by humans. In that case we would like to have automation devices (robots) that can perform such tasks effectively. Depending on the goal that a particular robot serves, there can be a number of sensors that match the human ones - vision (cameras, distance sensors), touch (force sensitive devices), sound (sonars). This is a large and active research and development area which we will not cover here.

If we abstract the sensors level out, we are looking at the goal of motion and manipulation of a robot in a complicated environment. The manipulation tasks are usually performed by attaching a gripper ("hand") at the end of the robot structure that allows it to grasp objects , reposition them, do insertions, painting or whatever other tasks are required.

The manipulation can also be done by several cooperating robots, moving robots, etc. Again for the sake of simplicity we will omit the actual manipulation part of the problem. Interested reader can refer to [17, 8] for more details.

At this point of abstraction the general problem we are considering is the following: Given an environment with objects (we call them obstacles) and a robot that is moving in the environment, how can we plan that motion so that robot moves from a given start point A to a given goal point B without collision with the obstacles. When the robot has a given structure, this problem is the so-called "motion planning" problem introduced in [19] and well studied in the robotics literature ([2], [8], [17]).

We have generalized this problem even further in [11] and [12]. In that work we considered the question: given the complex environment above, what is the most appropriate design of a robot that can reach everywhere in such environment in an "optimal" manner without collision with the obstacles. We addressed the issues of: what is the most appropriate type for the structure of the robot; what is the minimum number of structural elements (links) that are needed to cover every point in the free space (not occupied by the obstacles); what is the best placement for the robot in the environment, what are the shortest motion paths in the environment. In answering those questions we introduced "telescoping links" for the structure of the robot (2 degrees of freedom links that rotate and linearly extend from the joint). We developed algorithms for calculating the set of points in the environment satisfying the requirements above. That was done for general shapes of the obstacles (2D, 3D, convex, curvilinear, non-convex). We also proved analytical limits for the optimum number of links given the number and complexity of the obstacles. Further off, we addressed the problem of simultaneous design of the robot and the environment; the case when the robots are mobile in the environment; there are a number of robots; and they can have variable structure.

3.2 High-Dimensional Configuration Space obstacles

How does the problem that we just described relate to a multi-dimensional modeling? We will illustrate that in the following example. Consider the 2D environment in Figure 4 consisting of two polygonal obstacles and the circular robot moving in it.

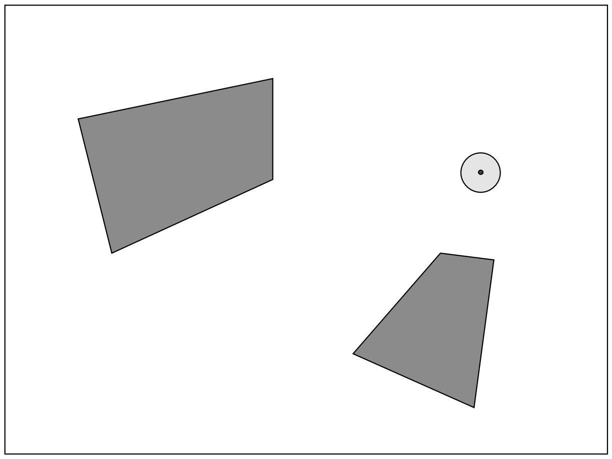
Figure 4. Circular robot in cluttered environment.

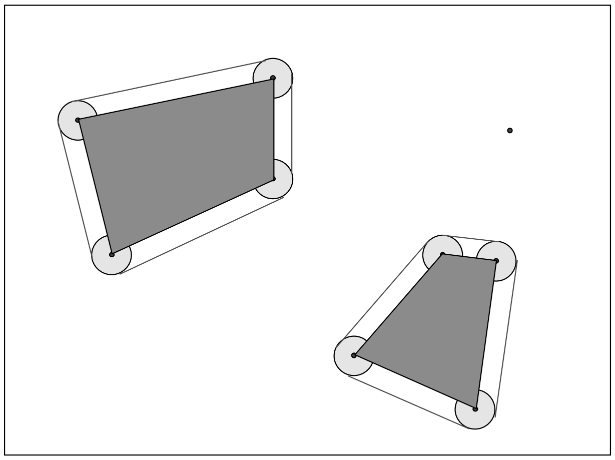
Figure 5. Configuration space obstacles for Figure 4.

Geometrically it is much easier to plan paths for a point in such environment rather than for a circle. Thus we will represent the circular robot by the center point of the circle. Correspondingly we will grow the obstacles (objects) in the environment with the radius of the circle, building the so-called C-obstacles (configuration space obstacles) [19]. The planning is now done in this Configuration space (C-space) in Figure 5. Finding shortest paths in such space between two points is a geometric problem which can be solved in a number of ingenious ways, like: drawing tangents to obstacles, back propagating from the goal to the start, connecting the two points with an elastic band and fitting it around the obstacles (see [17] for examples).

In this simple case of a circular 2D robot the dimensions of the robot space and the C-space are the same (2). Let us now look at the case when the robot is a 2D rectangular one and it can both translate and rotate in the environment. If we consider just translations of the robot first and represent the robot by its center point , the corresponding C-obstacles are shown in Figure 6 below.

If we want to account for the rotation of the robot, we need to move to a higher-dimensional (3D) C-space. We can think of Figure 6 as corresponding to motion of the robot when the angle α between the center horizontal robot axis and the X-axis, is zero. For every value of α , the corresponding C-space can be looked at as a planar horizontal plane in 3D. We can stack continuously (with respect to α) those planes in the vertical direction and the resulting 3D obstacles in Figure 7 represent the C-obstacles for that problem.

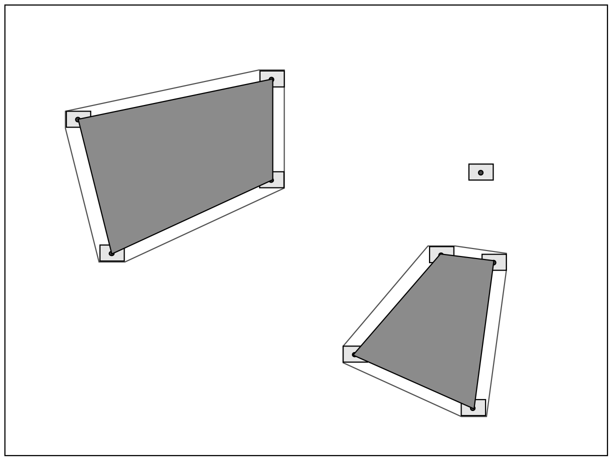
Figure 6. C-space for a rectangular translating robot.

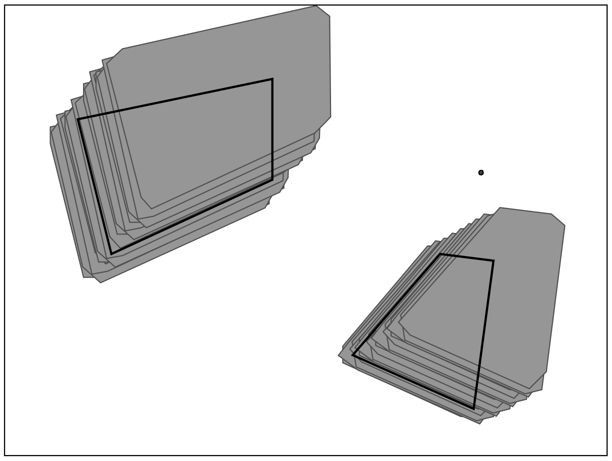
Figure 7. 3D C-space for a rotating and translating robot.

In that case planning a path from one point (given position and orientation of the robot) to another reduces to finding the "best" (shortest, optimal, min. energy, etc.) path in 3D configuration space avoiding the C-obstacles. There are numerous techniques (quadtree decomposition, 3D backprojection, uncertainty cones, gradient descent, flexible bubbles, potential fields, etc.) that have been introduced to solve this problem (see [8, 17]).

3.3 Approaches for solving high-dimensional problems in robotics

So far we have considered a planar robot in a planar environment. In reality the robot is 3-dimensional, it moves in a 3D environment and can have multi-link structure (e.g. a manipulator arm with several parts that can move independently) . In that case we can extend the C-space approach above and the corresponding spaces quickly become very high dimensional . For articulated or high-dimensional robot we generally increase the dimensionality of the space by one for each DOF (degree of freedom) or functionality of the robot.

One way to deal with this dimensionality problem is by projecting in a lower dimensional spaces. Often in an industrial environment objects tend to be vertically homogeneous and can be approximated as $2\frac{1}{2}D$ objects (i.e. correct planar projection that is extended vertically). In that case planning can be done in the projection space and extended vertically (see [11]).

The problem with multi-link robots can be dealt with in some cases by combining local and global motion planning. In that case the main (big) structure of the robot is approximated by simpler shapes. The motion through large, relatively uncluttered space is planned for those shapes. When the robot arrives close to the goal and precise, detailed motions need to be performed, a fine motion planning is done for much smaller spaces (see [12], [17]).

All approaches above have been well developed for particular cases, but still do not allow to attack the general n-dimensional problem. For that we need to abstract the geometry of the situation and move to topological modeling. In topological spaces the obstacles and robots will describe certain subspaces or cellular complexes and appropriate operators are needed to define the transition in the spaces. There has been very limited research in that area ([2, 12]) and further investigations is in order.

The topological approach for working with high-dimensional data that we developed in the area of data compression [20] introduces structures which can prove useful for modeling the general motion planning and robot design problem above. In the next part we will give a brief overview of the data compression problem.

Part IV COMPRESSION OF MULTI-DIMENSIONAL DATA

4.1 Representing data in real environments as functions on manifolds

Topological space modeling can be used in high-dimensional compression for representing, manipulating and compressing realistic 3D data. We will introduce that application by first describing a system we built for compressing functions defined on surfaces of 3D objects.

Many of the current topics of computational interest involve the representation of and the computation on continuous spaces (or approximations of such). Obvious examples are images, video, and audio. In these cases data (e.g. picture luminance) is distributed over and attached to a continuous rather than a discrete space. A photograph is well modeled as a reflectance over a rectangle. The spaces to which the functions of interest are attached are not always exactly the familiar Euclidean 1, 2, or 3 dimensional spaces.

We can also analyze the description of the functions defined on manifolds by considering the related notion of a ***multi-resolution*** description of a function [6]. By going to several levels of sub-division, we can use each level as an approximation to the function being described. If we are careful, we can avoid repeating any information about the coarser levels, so that we get, subdivision level-by-subdivision level, an increasingly detailed description of the function. By choosing an application appropriate way of doing this and taking advantage of natural properties of the function a great deal of function description can be represented in a compact data structure.

Our computer representations are necessarily finitely generated, so we are always talking about representations that ***approximate*** the real world. Of course, we need to measure and control the degree of approximation appropriate to our specific application. A good approximation schemes will not only have appropriate and measurable fidelity, but will also be economical of both computational and memory resources. Such approximations are often called ***compression*** schemes, because they greatly reduce storage requirements, but often also enable a great reduction in the computation required. It is generally under appreciated the extent to which a compressed representation will reduce computation just by the simple fact that the amount of data to be processed is greatly reduced.

Traditionally, data compression methods have been applied to functions defined on simple manifolds such as the real line (e.g., audio), a rectangle (e.g., images), or a three-dimensional open-ended box (e.g., video). However, many conventional data compression technologies, unmodified, are not suitable for compression of data defined on more complex geometries such as spheres, general polytopes, etc. In [13] we introduced a transform compression technique for addressing 2-manifold domains using second generation wavelet transforms and zerotree coding.

4.2 High-dimensional compression using spherical wavelets transform and zerotree coding

The transform compression of a function involves three steps: a transformation of the function, quantization, and entropy encoding. During the first step the function is subjected to a reversible linear transformation in order to concentrate most of its entropy (i.e., information) into a low dimensional subspace, thus simplifying its description. A wide variety of transformation techniques are currently in use, including the discrete cosine transform (DCT) and wavelet transforms.

Wavelets supply a basis for the functions we are representing [3, 4]. They decorrelate the data because in some way they resemble the data we want to represent. More specifically, wavelets have the same correlation structure as the data. They are local in space and frequency. Typically they have compact support (localization in space), are smooth (decay towards high frequencies), and have vanishing moments (decay towards low frequencies). More precisely, the wavelet representation leads to rapidly converging approximations to functions that satisfy mild smoothness assumptions. Finally, the wavelet representation of a data set can be found quickly. More precisely, we can switch between the original representation of the data and its wavelet representation in a time proportional to the size of the data. The fast decorrelation power of wavelets is the key to compression applications.

We can not use easily traditional wavelets for the transformation of data defined on complex manifolds. They are built using translation and dilation of a "mother wavelet" - a process that only makes sense in a Euclidean space. Instead, we will use the "second generation wavelets" introduced in [26] for building wavelets on a sphere.

The lifting scheme [28] and the spherical wavelets are techniques introduced recently for enabling wavelet construction for more general cases than the typical 1D or 2D planar spaces. In that, the wavelet coefficients are generated through a simple linear prediction and update scheme. This is a multi-resolution scheme where in the coarsest level the object is represented by a simple base complex (e.g. an icosahedron). All the cells of this complex are subdivided to generate the next level of approximation and the corresponding wavelet coefficients are computed for the newly generated vertices. This process is continued until appropriate coverage of the data set is achieved. The result is a refined triangular mesh with subdivision connectivity. We have introduced in [13] a tree structure, called a G-tree, in which each node represents a cell of the triangular mesh. Second generation wavelets for the specified function are calculated and scaled, the wavelet coefficients being defined at the vertices in the triangular mesh and at the vertex correspondent nodes of the G-tree.

The zerotree algorithm was introduced for effective and fast embedded (progressive) compression of images [25, 27]. Research into adaptive N-largest non-linear analysis by DeVore offers a possible theoretical foundation of the method. In our context that algorithm processes the wavelet coefficients generated from the transform analysis part based on significance with respect to given threshold. The coefficients are arranged in a tree structure (see [13]) whereby the main premise of the method is that if a certain coefficient is insignificant with respect to the threshold, all coefficients below that one in the tree are also insignificant. That allows for the tree to be pruned whenever the premise is true and hence a smaller set of coefficients are written out representing the data. Using a modified zerotree encoding scheme, the G-tree is processed threshold by descending threshold, outputting bits indicative of significant G-tree nodes and the corresponding coefficient bits. This results in a bit plane by bit plane embedded encoding.

The decoding algorithm inputs bits according to the modified zerotree scheme into the G-tree structure, refining the wavelet coefficients. De-scaling and an inverse second generation wavelet transform completes the synthesis of the original function. The canonical ordering of the bits is similarly generated by both the encoder and the decoder.

In our system the user can interactively select the type of wavelets to be used, the base complex , the number of levels of subdivision, the function to be compressed, and the desired compression. The program starts with a simple base representation as a triangular mesh for the object that we are modeling. For example when we model an image the base complex is a rectangle subdivided in two triangles. We achieve more detail in the representation by subdividing all the triangles in the current model based on some subdivision rule. In computer graphics the predominant method is mid-point subdivision. The newly created vertices are projected up on the surface of the model using different projection methods. For complicated 3D surfaces we often use the "Dyn" (or "butterfly") [5] projection, where a stencil of 8 neighboring points are used to compute the position of the new vertex. The coefficients in the newly created vertices are computed using the wavelet analysis. The number of subdivision levels is determined by the limitation of the hardware and the complexity of the model.

4.3 Simulation results from modeling data defined on surfaces of 3D objects

The system that we implemented based on the description above achieves significant compression results. In the examples below the function that we are compressing is defined by the elevation data on a grid for the surface of the Earth (see [13]). We mapped this function to several different 3D objects. The canonical example maps the function on the surface of a sphere, thus representing the actual Earth shape and data set. We also experimented with using the surface of a multiresolution triangular mesh representation of a cat, teapot, flat rectangle (representing an image in [13]) and other shapes. While the cat surface described in [16] is still homeomorphic to a sphere, the teapot surface described in this paper is significantly more complicated.

The types of results we obtained are the first result of a kind, in that there are no other compression data (in terms of PSNR or bits per pixel results) in 3D to compare our performance with. For comparison with the existing 2D compression algorithms, we applied in [13] our method to a flat manifold (an image). The results compared very favorably with the state of the art in image compression. We obtained even better results in [14] and [15] using different wavelet predictors; scaling of the coefficients based on the capabilities of the Human Visual System (HVS) [4]; optimization of the tree structure; and using arithmetic coding [30] rather than straight binary entropy coding.

The simulations were coded in C++ on a UNIX platform (SGI Indigo 2 Impact 10000). For interactive visualization the implementation requires that the entire data structure be kept in RAM. For the Teapot base complex the 256 MB RAM of the SGI limits the subdivision level to three. The Earth example in [15] allowed for 8 levels, while a typical gray-scale image can be completely modeled by 9 levels of subdivision. In this paper we used only 7 levels for the Earth so that the number of coefficients are comparably to the Teapot example.

For the Teapot example we compressed the topographic function that is the elevation (with respect to sea level) of the Earth. This function is initially approximated by the ETOPO5 data set (satellite data available from the National Geophysical Data Center) which samples the Earth every 5 arc minutes (the entire data file is 6 million points). This data set was 4:1 sub-sampled to the ETOPO10 data set which samples the Earth every 10 arc minutes (1.5 million points approximately 11 miles apart). The resulting data set is mapped on the surface of the teapot by a radial projection. The center of the sphere of the projection is located at the mass center of the teapot and the data value for each point on the surface of the teapot is calculated by intersecting the ray from the center to the point with the sphere wrapped around. . The topographic elevation at each vertex was then determined by interpolation of the ETOPO10 data set. The result is color coded based on that elevation value.

	Teapot(3 levels,197826 coeff.)			Earth (7 levels, 163842 coeff.)		
bitplanes	PSNR	b/vertex	Compress	PSNR	b/vertex	Compress
10	64.72	6.06	1.3:1	44.01	1.7	5:1
9	56.91	4.67	1.7:1	38.25	0.73	11:1
8	49.99	3.42	2.3:1	33.92	0.3	26:1
7	44.35	2.35	3.4:1	30.82	0.13	63:1
6	39.65	1.55	5.2:1	28.27	0.055	145:1
5	35.59	1.02	7.9:1	25.83	0.023	348:1
4	32.27	0.71	11.2:1	23.74	0.011	753:1
3	29.68	0.56	14.2:1	21.78	0.0055	1447:1
2	28.04	0.51	15.6:1	19.61	0.0032	2497:1
1	27.16	0.5	16.1:1	18.41	0.0027	2919:1

Figure 8. PSNR data for different compression ratios.

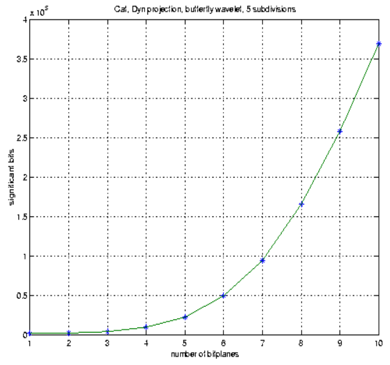

Figure 9. Significance Bits Allocation.

When new points are generated via subdivision their function values are calculated with the same procedure. Their geometrical location is computed using the butterfly scheme over the spatial (x, y, z) coordinates of the coarser level vertices. The data is wavelet transformed using butterfly lifting and compressed at various ratios. The base complex of the teapot is a triangular mesh with 3072 vertices and 6182 triangles. After 3 levels of subdivision we generate 197,826 vertices (wavelet coefficients) that cover about one eight of the available data points in ETOPO10 (we also have 395,648 triangular faces).

For the Earth example we use the same function mapped on the surface of a sphere approximating the Earth. The base manifold is an icosahedron (12 vertices, 20 triangular faces and 30 edges) and it is subdivided using midpoint subdivision. Geometrically the newly generated points are projected up on a sphere using geodetic projection. We use the butterfly scheme as a prediction operator. The data is subdivided 7 times which results in 163,842 vertices, and 327,680 triangular faces (covering about a ninth of the ETOPO10 data points).

Figure 8 summarizes the results for the peak signal-to-noise ratio (PSNR) for the two examples above for several different compression ratios. Scaling was chosen appropriate to the L_2 norm. The table reports the results relative to the interpolated vertex data. PSNR is calculated as in [13] (the range over the mean square error).

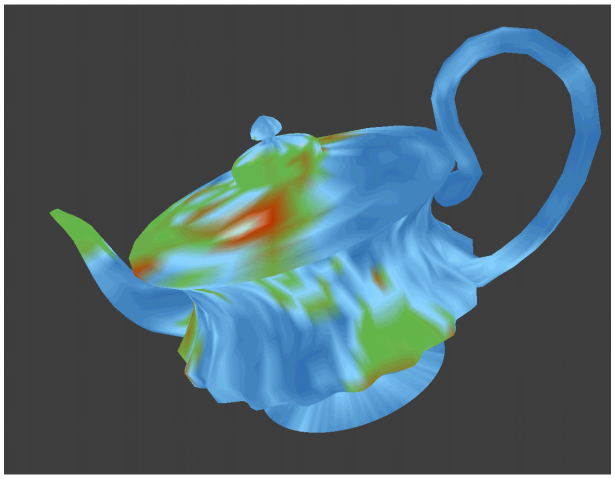
Figure 10. Teapot manifold (ETOPO10 data mapped).

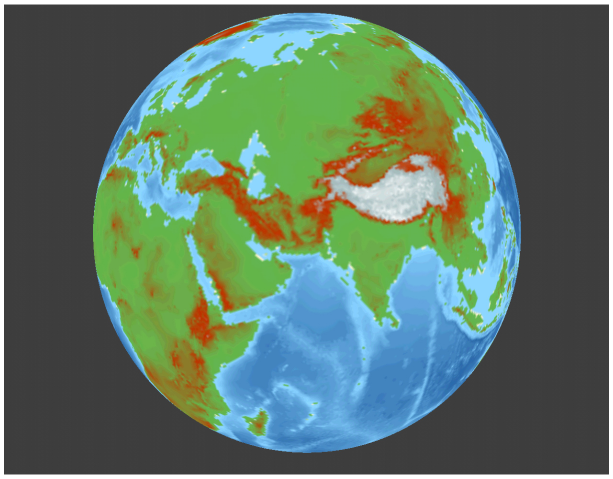
Figure 11. Earth Manifold (ETOPO10 data mapped).

Each row in the table corresponds to the number of bitplanes read during the decompression. Every coefficient is represented with 10 bits. In all cases the compression subroutine writes out the significant bits associated with all 10 bits for all the coefficients bitplane by bitplane. The first row in the table corresponds to the case when during decompression all 10 bitplanes of significant coefficients are read in. In the next row we read only 9 bitplanes (the most significant ones) for all coefficients, etc. As we can see, even when we read in all the bitplanes that we wrote, there is still a 5:1 compression (for the Earth example) due to the zerotree compression.

The number of significant bits increase exponentially with the number of bitplanes retained in the representation of the coefficients. For the Earth example, the most significant bitplane (bitplane 10) have only 54 bits that are significant for all 163,842 coefficients. In contrast, the least significant bitplane (bitplane 1) have 142,507 significant bits. For the Teapot example, since we start with quite a large number of coefficients in the base triangular mesh, the number of bits is relatively similar throughout the bitplanes (about 16,000). Figure 9 illustrates the relationship between number of significant bits and number of bitplanes used for coefficient representation for the Earth example. That curve shows that if we select the number of bitplanes for decompression such that we guarantee adequate PSNR, we can achieve a significant amount of compression. In fact with bitplane reduction we can achieve close to 100:1 compression with the PSNR being in the virtually lossless range. Visually Figures 10 and 11 illustrate the quality of the compression. The results look exactly as what we would expect for good compression given the resolution that we are achieving.

In practice modeling data or objects in the real world can bring to a need for higher-dimensional representation. For example if we want to build a complete 3D model of a complex object, we can use a method generalizing the "light fields" method (see [18]). In that case two spheres are wrapped around the 3D object and the model is built by looking at intersection of radial rays with the surface of the object. The resulting model is a cross-product of the two families of spheres, which is a 4-dimensional object.

Similarly in considering state space problems, a number of high-dimensional spaces arise. In order to efficiently compress such models, we need to be able to manipulate m-dimensional functions on n-dimensional manifolds. Similarly to what we described earlier, one way to do this is by abstracting the geometry of the problem and working in topological spaces. We have developed one such possible approach in [20].

4.4 Modeling and compressing n-dimensional data

In [20] we generalized the approximation domain for scalar functions described in [15]. In particular we described the transition from 2-manifolds to n-manifolds (2-simplices to n-cells), from mid-point subdivision to dual-intersection subdivision. We did that while retaining finite stencils of support for wavelet multi-resolution analysis and preserving the compression techniques.

From a mathematical point of view, we introduced the foundations for a computational approximation theory for CW complexes. We addressed the question: To what extent can approximations of real valued functions defined on curves, surfaces, manifolds, and other Euclidean type spaces, be effectively and computationally carried out by purely topological means?

This question is approached by considering domains that are homeomorphic to finite CW complexes (see [21] for introduction to algebraic topology) , which tessellate a locally Euclidean space into a finite number of cells of various dimensions. Associated with these cell complexes are the chain complex and boundary homeomorphism that describe how the cells are linked together. We can build *regular* CW complexes where the boundary operator can be easily represented and computed.

A simplistic explanation of CW complexes follows: Consider a 3D triangular mesh representing an object in space. That mesh has vertices, edges and triangles on the surface. If we also consider the volume of the object, it can be subdivided into pyramids. In defining the cellular structure, the vertices in the model denote cells of dimension 0. The edges are cells of dimension 1, the triangles are cells of dimension 2 and the pyramids are cells of dimension 3. Similarly for an object in n - dimensional space, we can define cells of dimensions 4, 5, …, n. To build the cellular complex we need to describe the relationships between the different cells. In particular we need to specify which cells constitute the **boundary** of a given cell and which are the neighbors (or the **star**) for a given cell. With those two operations correctly defined, we can describe any m-dimensional object in an n-dimensional space [21].

We used cells rather simplices (which have been used previously in multiresolution approaches) because cell constructions typically require fewer cells than the equivalent number of simplices. That leads to smaller data structures and quicker computations. We can also build better subdivision methods and there is no need for complicated fix-ups upon partial or adaptive subdivision.

Approximation is accomplished by the iterated application of a subdivision operator which can eventually separate any two points into the interiors of separate cells. Algebraically, a subdivision operator is a 1:1 chain homomorphism from the CW complex to another CW complex generated by the result of the cell partitioning. The regularity of the CW complex also enables the calculation of the boundary operator in the subdivided complex. We use a finite sequence of finite regular CW complexes to approximate continuous functions, to build and compress wavelet expansions of such functions.

We have developed the Spherical Lifting Wavelet Library SLW (described in [20]) as a set of C++ routines intended to represent topological objects. That library can be used for a variety of applications like:

- Multidimensional signal compression. When the underlying signal has some notion of geometry, CW complexes can be used to approximate both the domain and range space of the signal. We would like to be able to build approximations to continuous functions via the cellular approximation theorem. A systematic approach to refining the approximation gives rise to a multiresolution scheme and the possibility of efficiently representing the signal, i.e., wavelets. The approach should preserve the compression techniques described in [15].

- Efficient representation of texture maps for computer graphics applications. In some computer graphics problems, thousands of simplices need to be texture mapped to properly display a scene. Efficient storage and rapid usability of texture maps can be studied using the library.

Part V SUMMARY AND CONCLUSIONS

There are a number of possible approaches for dealing with high-dimensional data obtained in modeling of the real world. The approaches that we described in this paper try to deal with the data in the space where it lives, rather than projecting it or reducing it to lower dimensional subspaces. As a result we obtain simpler, faster algorithms that are easy to implement in practice.

We have also described several application domains for such approaches. In particular we considered: modeling of population genetics systems using evolutionary algorithms; solving motion planning problems from robotics using high-dimensional configuration spaces; and compressing data defined in complicated real world environments by using geometrical and topological modeling.

The applications that we described so far allow us to interact visually and analytically with high-dimensional data. Another powerful sensorium that we as humans actively use, is the sense of touch. Using state of the art force-feedback "haptic" devices we can try to model in the computer complicated 3D objects and explore them virtually using our hands and our sense of touch. In [23] and [24] we have developed and implemented a system that allow us to quickly (in a manner of minutes) take complex 3D objects or scenes (defined by tens of thousands to hundreds of thousands of triangles) and build haptic virtual models. Users can explore and manipulate those models adding a powerful dimension of interaction created by the force feedback experienced. Using compact representations (built using the high-dimensional approach that we described) of the environments allows for enhanced experience with realistic environments.

There is an underlying connection and relationship between the approaches to solving the problems that we described. Clearly different methods are the most appropriate ones for solving each separate problem. Thus instead of developing a general "universal" approach, for any given problem we have concentrated our efforts on parametric and theoretical understanding of the problem, classifying it in a general category , classifying the potential solution models with respect to their strengths and characteristics; and finding and applying the most appropriate method for the given problem.

REFERENCES

[1] A.Bergman and M.Feldman, Recombination Dynamics and the Fitness Landscape, *Physica D 56*, p. 57-67, 1992

[2] J. Canny. *The Complexity of Robot Motion Planning*, MIT Press, Cambridge, MA, 1988.

[3] I. Daubechies. *Ten Lectures on Wavelets*, CBMS-NSF Regional Conf. Series in Appl. Math. Vol. 61. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1992.

[4] R. A. DeVore, B. Jawerth, and B. J. Lucier, "Image compression through wavelet transform coding", *IEEE Trans. Inform. Theory*, vol. 38, pp. 719-746, March 1992

[5] N. Dyn, D. Levin, and J. Gregory. A butterfly subdivision scheme for surface interpolation with tension control, *Transactions on Graphics*, 9(2):160--169, 1990.

[6] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes, *Computer Graphics Proceedings, (SIGGRAPH 95)*, pages 173--182,1995.

[7] J.Holland. *Adaptation in Natural and Artificial System*, MIT Press, 1992.

[8] O. Khatib. Real-Time Obstacle Avoidance for Manipulators and Mobile Robots, *International Journal of Robotics Research*, 5(1), pages 90-98, 1986.

[9] K. Kolarov. Landscape Ruggedness in Evolutionary Computation, *Proc. of IEEE International Conference on Evolutionary Computing*, Indianapolis, 1997.

[10] K. Kolarov. The Role of Selection in Evolutionary Algorithms, *Proc. of IEEE International Conference on Evolutionary Computing*, Perth, Australia, pages 86-91, 1995.

[11] Kolarov, K. *Optimal geometric Design of Robots for Environments with Obstacles*, PhD Thesis, Stanford University, 1992.

[12] K. Kolarov. Algorithms for Optimal Design of Robots in Complex Environments, *Algorithmic Foundations of Robotics*, A.K.Peters, Ltd., 1995.

[13] K. Kolarov and W. Lynch. Compression of functions defined on surfaces of 3D objects, *Proceedings of the Data Compression Conference*, Snowbird, Utah, pages 281--291, March 1997.

[14] K. Kolarov and W. Lynch. Optimization of the SW algorithm for high-dimensional compression, *Proceedings of SEQUENCES'97* , Positano, Italy, June 1997.

[15] K. Kolarov and W. Lynch. Wavelet Compression for 3D and Higher-Dimensional Objects, *Proc. of SPIE Conference on Applications of Digital Image Processing XX* , Volume 3164, San Diego, California, July 1997.

[16] K. Kolarov and W. Lynch. Compact Representation of Functions on Multidimensional Manifolds, talk at *Curves and Surfaces*, Lillehammer, Norway, 1997.

[17] Jean-Claude Latombe. *Robot Motion Planning*, Kluwer Academic Publishers, 1991.

[18] M. Levoy and P. Hanrahan. Light Field Rendering, *Computer Graphics Proceedings, Annual Conference Series*, *SIGGRAPH'96*, New Orleans, Louisiana, August 1996.

[19] T. Lozano-Perez and M. Wesley. An Algorithm for Planning Collision-Free Paths Among Polyhedral Obstacles, *Communications of the ACM*, 22(10), pages 560-570, 1979.

[20] W. Lynch and K. Kolarov. Topological Approximation and Compression of Functions and Their Wavelet Expansions, *Proc. of the International Wavelets Conference "Wavelets and Multiscale Methods"*, INRIA, Tangier, Morocco, April 1998.

[21] W. Massey. *A Basic Course in Algebraic Topology*, Springer-Verlag 1991

[22] J.Roughgarden, *Theory of Population Genetics and Evolutionary Ecology: An Introduction*, Macmillan Publishing Co., 1979.

[23] D. Ruspini, K. Kolarov and O. Khatib. The Haptic Display of Complex Graphical Environments, *Computer Graphics Proceedings, Annual Conference Series*, *SIGGRAPH'97*, Los Angeles, California, August 1997.

[24] D. Ruspini, K. Kolarov and O. Khatib. Haptic Interaction in Virtual Environments, *Proc. of the IEEE International Conference on Intelligent Robots and Systems IROS'97*, Grenoble, France, September 1997.

[25] A. Said and W.A. Pearlman. A new fast and efficient image codec based on set partitioning in hierarchical trees, Submitted to the IEEE Transactions on Circuits and Systems for Video Technology.

[26] P. Schröder and W. Sweldens. Spherical wavelets: Efficiently representing functions on the sphere, *Computer Graphics Proceedings, (SIGGRAPH 95),* pages 161--172, 1995.

[27] J.M. Shapiro. Embedded image coding using zerotrees of wavelet coefficients, 41(12):3445--3462, 1993, IEEE Trans. Signal Process.

[28] W. Sweldens. The lifting scheme: A custom-design construction of biorthogonal wavelets, Technical Paper 1994:7, Industrial Mathematics Initiative, Department of Mathematics, University of South Carolina, 1994.

[29] M. Vose and C. Liepens, Punctuated Equilibria in Genetic Search, *Complex Systems 5*, 1991, p. 31-44

[30] I. Witten, R. Neal and J. Cleary. Arithmetic coding for data compression, *Communications of the ACM*, v.30, n. 6, pages 520-540, June 1987.