Synthesis of a Multizone Survey Visualization Palette on the Learning Basis[1]

L. M. Mestetskii and K. V. Rudakov

Computer Center, Russian Academy of Sciences, ul. Vavilova 40, Moscow, 117967 Russia
e-mail: mest@kemar.msk.ru

Abstract—A multizone survey visualization method is outlined, which is based upon a multizone frame palette clusterization, expert choice of the colors of some clusters and automatic formation of remaining clusters colors.

1. INTRODUCTION

Present-day approaches to terrestrial surface on-line monitoring are essentially based on multizone surveys carried out onboard satellites or aircrafts. Availability of receiving complexes that are relatively inexpensive, compact and easy to maintain serves to increase the number of the users of natural resources satellite information. Availability of this information in real time and a sizable amount of corresponding data pose the problem of multizone video information on-line visualization in order to both analyze it effectively and exploit it by the users.

A conventional solution to the visualization problem consists of the following. One chooses three channels from the set of all spectral channels, and assigns one of the three basic colors (red, green or blue) to each of them. The picture obtained from the three monochrome pictures superposition, after some additional refinement (for example, by the histogram expansion method) is a visualization result. The deficiencies of such an approach are evident. For one thing, only part of the available information (only 3 spectral channels of 5–10) is used. On the other hand, the resulting colors often do not adequately represent habitual terrestrial surface coloration.

The method of attack presented here enables the user to choose coloration for a set of a single or several multizone picture fragments, thus training the computer. Thereafter the entire survey, i.e., all the imagery generated under similar conditions, is automatically visualized on the basis of the learning in the color palette constructed in such a way. In this case, information from all the multizone survey spectral channels is used.

2. FORMULATION OF THE PROBLEM

Let us assume that the multizone survey is carried out in such a way that all individual pictures in every range (single zone pictures) are executed simultaneously from the same point, that they represent the same area, and have the same resolution. Moreover, let us assume that all the single zone pictures are matched with respect to all the picture elements (pixels).

Let X be a set of the picture elements which are arranged as a rectangular matrix. Each ith single zone picture is a monochrome (gray) one and is specified by function $F_i(x)$, $x \in X$, which maps the set of pixels X into the set $I = \{0, \ldots, 255\}$, which represents 256 levels of gray color intensity.

The multizone picture consists of m single zone ones and is depicted by vector function $F(x) = (F_1(x), \ldots, F_m(x))$ that maps X into integer-valued m-dimensional cube I^m:

$$F: X \longrightarrow I^m. \quad (1)$$

Let $S \in I^m$ be an image of a map F. Let us call S multizone palette of picture (1). The visualization process of multizone picture $F(x)$ is reduced to a generation of the so called visual picture. The visual picture (*RGB* image) is a vector function $H(x)$, $x \in X$ that associates every pixel x with vector $(R(x), G(x), B(x))$ of red, green and blue color intensities respectively for representation on the color display.

A peculiarity of the mapping devices most extensively employed at the moment are limitations on the total number of *RGB*-colors that simultaneously exist on the screen. This implies that there can be no more than M colors in the palette ($M = 256$ is common). Each color in the *RGB* palette can be interpreted as a point in three-dimensional integer-valued cube I^3.

Thus, the visual picture is a function $H(x)$ that maps a set X of the pixels into some color palette P,

$$H: X \longrightarrow P, \quad (2)$$

where P is a finite set of M points in the integer-valued cube I^3.

In accordance with the notations introduced, the multizone picture visualization problem is reduced to the construction of both color palette P and mapping T

[1] This work was supported by the Russian Foundation for Basic Research, project nos. 96-01-00552 and 96-01-00553.

Received May 26, 1998

of integer-valued cube I^m together with multizone palette S contained in it, into a RGB-palette P, i.e.,

$$T: I^m \longrightarrow P. \quad (3)$$

If the visualization problem is solved and mapping T is found, direct transformation of an initial multizone picture to an RGB-image consists of a calculation of function $H(x) = T(F(x))$ for all the pixels $x \in X$.

Setting up the problem has the following peculiarities in different applications:

—a great number of the multizone survey spectral channels (up to m=10);

—a limited number of colors in palette P (M = 100–120) which owes to the necessity of the part of the color palette redundancy for different computer system needs;

—a necessity of immediate visualization in real time in order to carry out on-line check and control at the multizone survey operator working place.

3. METHOD OF SOLUTION

The suggested approach consists of constructing a color RGB-palette P and mapping T on the basis of processing a few multizone learning pictures with the participation of an operator-teacher (at the preliminary learning stage).

Following direct visualization of the arbitrary multizone pictures of the same class as learning ones is realized then, using on-line palette P and mapping T. However, in the case of the appearance of multizone pictures of different classes (change of seasons, area types, etc.), P and T correction (i.e., additional learning) is possible.

In order to construct a palette P and mapping T, a method is proposed which incorporates the following stages:

(1) Carry out the multizone palette S clusterization by its partition to M clusters using the learning pictures.

(2) Choose RGB-colors for each of the M clusters. Here it is possible to choose colors for some clusters manually by the teacher-operator. Since the number of remaining clusters is sufficiently large, the colors for them may be chosen in an automatic mode.

Thus, two algorithms: one of multizone palette clusterization and the other of cluster coloration are the most essential in this approach.

4. MULTIZONE PALETTE CLUSTERIZATION

Clusterization is based on the construction of the Voronoi diagram for the integer-valued cube I^m. Elements of this diagram (clusters) are loci $W_1, W_2, \ldots, W_M$ that are defined in the following way.

Let R = [0.255] be a segment of the number axis, R^m—m-dimensional cube with side R, and $p_1, \ldots, p_M$—points inside R^m. Let us associate with every point p_i called a pole, a Voronoi polyhedra (locus) which is defined as

$$W_i = \{z \in R^m: \rho(z, p_i) \le (z, p_j),\ z \in R^m,$$
$$j = 1, \ldots, M\}, \quad i = 1, \ldots, M,$$

where $\rho(z, p)$ designates Euclidean distance between points z and p.

Every set W_i consists of all such points of m-dimensional cube R^m, the distance from which to the pole p_i does not exceed distances to other poles. The totality of loci $W_1, \ldots, W_M$ is called Voronoi diagram and is a covering of the cube R^m and, correspondingly, of the integer-valued cube I^m. Loci are convex polyhedra and intersect each other only at the boundary points. The problem of localization of the arbitrary sampling point in Voronoi diagram is reduced to determining the pole nearest to it from the set of poles $p_1, \ldots, p_M$.

A peculiarity of the multizone palette S is the fact that in spite of an immense number of possible vector-colors (up to 256^m) that could be contained in it, only a small part of them actually occur in multizone pictures. Besides, corresponding points are nonuniformly distributed in I^m, i.e., they form clusters. Therefore the problem of multizone palette clusterization is reduced to the choice of poles taking into account these clusters.

Thus, multizone palette $S = \{s_1, \ldots, s_n; s_j \in I^m; j = 1, \ldots, n\}$ is a set of all the vectors-colors that occur in the set of the learning pictures. Let us associate with every point s_j a weight $D(s_j)$ that is equal to the number of pixels in the learning pictures which have color vector s_j (i.e., the number of pixels $x \in X$ for which $F(x) = s_j$).

Since the distances from the points s_j to the poles p_i of the clusters determine the degree of their proximity to clusters, it is natural to take the sum of weighted square deviations of the points of S from the corresponding poles $p_1, \ldots, p_M$ as a criterion of the pole choice:

$$\Delta(p_1, \ldots, p_M) = \sum_{i=1}^{M} \sum_{s \in S \cap W_i} \rho^2(s, p_i) D(s).$$

Now the problem of the cluster poles choice reduces to determination of a set of M points $\{p_1, \ldots, p_M\}$ that minimize $\Delta(p_1, \ldots, p_M)$:

$$\min_{p_1, \ldots, p_M \in R^m} \Delta(p_1, \ldots, p_M). \quad (4)$$

Since a search of the global minimum in (4) is a NP-complete problem, let us use an heuristic algorithm in order to find an admissible local minimum, similar to [1].

For each locus W_i let us consider the point c_i that is its center of gravity,

$$c_i = C(W_i) = \sum_{s \in S \cap W_i} \frac{D(s)}{D(W_i)} s.$$

It is shown in [1] that if a partition of S to clusters were not dependent upon the poles $p_1, \ldots, p_M$ location, using loci c_i centers of gravity as poles would give minimum total square deviation in (4). However, if we allocate poles at the cluster centers of gravity, this immediately changes Voronoi polyhedra and clusters themselves. The concept of the algorithm proposed is to construct an iterative convergent process of the successive correction of the poles and cluster centers of gravity.

The process consists of two phases: successive generation of the cluster poles and following iterative cluster correction. In what follows, both those phases are implemented in a single algorithm.

Multizone Palette Clusterization Algorithm

1. Generation of the first pole p_1 as a center of gravity of all the points of S: $= W_1 := S$, $c_1 := C(S)$, $p_1 =: c_1$.

2. Let k be the number of generated poles $p_1, \ldots, p_k$. Generation of the pole p_{k+1}:

2.1. Let us find a locus W_r with the maximum total square deviation $\Delta_i = \sum_{s_j \in W_i \cap S} \rho^2(s_j, p_i) D(s_j)$:

$$r = \arg \max_{i = 1, \ldots, k} \Delta_i;$$

2.2. Let us decompose locus W_r into two subsets $W_r = W_r' \cup W_r''$ dividing it by a $(m-1)$-dimensional midpoint hyperplane that is perpendicular to one of the coordinate axes. We select the coordinate axis with maximum dispersion of points from $S \cap W_r$ along it. The hyperplane goes through the points projection median;

2.3. Let us calculate the centers of gravity of the sets obtained, W_r' and W_r''–points $c_r := C(S \cap W_r')$ and $c_{k+1} := C(S \cap W_r'')$, and let us assume that $p_r := c_r$, $p_{k+1} := c_{k+1}$.

3. Let us correct the location of all the $k+1$ poles taking into account the pole p_{k+1} generated. In order to do that:

3.1. For each point $s \in S$ let us find the nearest pole among $p_1, \ldots, p_{k+1}$. Thus the set S is decomposed to $k+1$ subsets $S = S_1 \cup \ldots \cup S_{k+1}$;

3.2. In every subset S_i let us calculate the new center of gravity $c_i = C(S_i)$, and let us determine the new location of the pole $p_i := c_i$.

4. If $k+1 < M$, assign $k := k+1$ and pass to the point 2. If $k+1 = M$ and if $\max \rho(c_i, p_i) < \varepsilon$, the process is finished, otherwise go to the point 3.

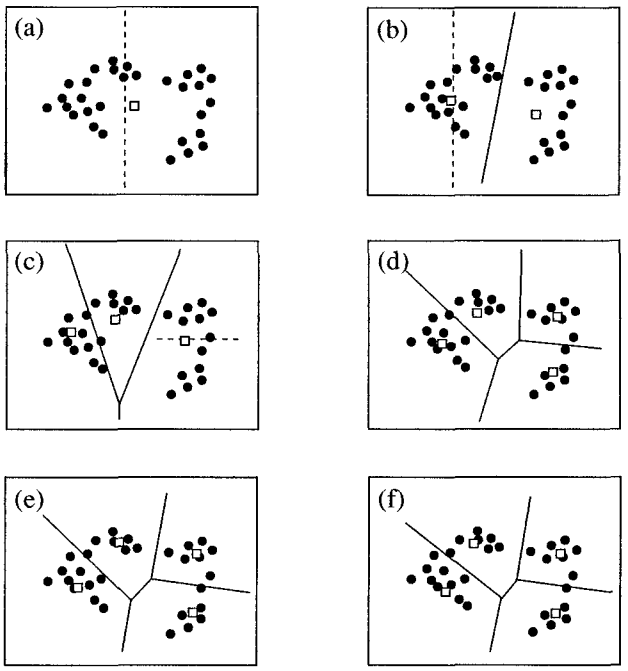

Fig. 1. Multizone palette clusterization

Figure 1 illustrates the algorithm operation when the number of spectral channels is $m = 2$ and the number of clusters is $M = 4$. Initial points in the multizone palette are represented by black circles, and cluster centers, by white squares. Redistribution of the points with respect to clusters after the poles have been changed is shown by solid lines. The first four drawings illustrate successive generation of the cluster poles. The last two drawings present iterations of the correction of cluster centers' locations.

It is known from [1] that this iteration process converges sufficiently well to some local minimum for problem (4), which is quite acceptable for applied problems.

5. CHOICE OF COLORS FOR A CLUSTERED MULTIZONE PALETTE

Loci $W_1, \ldots, W_M$ with poles $p_1, \ldots, p_M$ are constructed according to the results of the multizone palette I^m clusterization. Now every locus should be assigned some color in the visualization palette P. The process of the choice of these colors which we call cluster coloring is realized with the help of an operator-teacher. When analyzing the collection of the single-zone learning picture fragments at the screen of display, he can point out any pixel and choose the color from the visualization palette which should be assigned to it. When choosing the color, the operator is guided by his own preferences. One of the coloring versions corresponds to the geographical map coloring: the water surface is blue, mountains are brown with white glaciers, the woods are green, etc. Since the pixel chosen is

linked with some vector-color in a multicolor palette and this vector-color lies in one of the clusters, the same color chosen by the operator in the visualization palette can be associated with all the points of the multizone palette of this cluster. Thus, colors for all M clusters can be chosen manually, and, by doing so, the cluster coloring problem will be completely solved. However, since number M is too great for manual sorting, we propose to supplement manual visualization color matching by automatically coloring the remaining clusters. Its essence is, on the basis of the operator chosen colors for clusters $W_1, \ldots, W_k$, to assign colors to the remaining clusters $W_{k+1}, \ldots, W_M$. Naturally, the operator has the possibility of changing the color of any cluster colored both manually or automatically.

The essence of the automatic choice of colors for the clusters remaining after manual coloring is linear interpolation of the function $T(p)$ given at the nodes $p_1, \ldots, p_k$. Values of this function $T(p_1), \ldots, T(p_k)$ are colors prescribed by the operator for clusters $W_1, \ldots, W_k$ in the process of manual coloring, i.e., points of integer-valued cube I^3. Performing linear interpolation of function $T(p)$, we can ascribe to poles $p_{k+1}, \ldots, p_M$ the colors that correspond to the rounded off values $T(p_{k+1}), \ldots, T(p_M)$. Thus the problem is reduced to a multidimensional linear interpolation of the function given on irregular node mesh.

It is known that the best linear interpolation of functions of the two or three variables on an irregular finite node mesh is accomplished when we use Delaunay triangulation. Triangulation on a plane is a plane graph, all faces of which are triangles. Delaunay triangulation [2] is a triangulation where all triangular faces have empty circumscribed circles, i.e., such that they do not contain any graph vertices inside. It is known that for a given set of general position points (such that no four of them belong to a single circumference) Delaunay triangulation exists and is unique. Effective algorithms of Delaunay triangulation construction for a plane [2] and three-dimensional space [3] are also known.

Employing the Delaunay triangulation for solving the problem of two or three variable function interpolation is as follows. At first the problem of localization is solved for an arbitrary test point, i.e., the problem of determining a simplex (triangular face in the two-dimensional case or a tetrahedron in the three-dimensional case) containing this point. Then, both baricentric point coordinates relative to the simplex and function value at the test point, according to the function values at its vertices, are calculated. A single constraint is that the test point should be inside the interpolation nodes convex shell, because otherwise the test point localization problem does not have a solution.

The basic idea of function interpolation by Delaunay triangulation can easily be generalized to a multidimensional case. Here, m-dimensional triangulation of a general form will be understood as a graph, all hyperfaces of which are m-dimensional simplexes that do not have mutual inner points, and Delaunay triangulation will be a triangulation, all hyperfaces (simplexes) of which have empty circumscribed hyperspheres. Then the interpolation problem can also be solved, in theory, by test point localization and calculating its baricentric coordinates in the simplex.

However, implementing this idea for the case of space dimension $m > 3$ runs into considerable problems. Algorithms of construction of all the Delaunay triangulation edges in multi-dimensional cases are known [4]. But this is not enough to solve the interpolation problem: one must have the triangulation hyperfaces (simplexes with vertices in nodal points). However, the number of hyperfaces in triangulation grows exponentially with space dimensionality m, and its value is of the order of $O(k^{m/2})$, where k is the number of triangulation vertices [4]. Thus the construction of all such triangulation hyperfaces can take an unacceptable amount of memory and time. Hence, even for m = 5–6 it is impossible, in practice, to use a traditional line of attack to construct Delaunay triangulation and then consequently solve the localization and interpolation problem for all the test points.

The suggested approach consists of abandoning the calculation of all the Delaunay triangulation at first, and constructing, immediately, only its individual simplexes which are necessary in order to solve the test point localization problem. Naturally such an approach can essentially take greater time for localization as compared with the prior triangulation construction at the prepossessing stage. But since, in our particular case, the number of points $p_{k+1}, \ldots, p_M$ for which it is necessary to calculate $T(p)$ is relatively small, these time expenditures appear to be quite acceptable.

The realization of the suggested approach consists of two stages. The first stage of initial search is Delaunay simplex construction, the hypersphere of which contains the test point. For this stage a "hypersphere inflation" algorithm is proposed. The second stage incorporates an accurate search of the enveloping Delaunay simplex that contains the test point. In order to carry out an accurate search, a simplex tracing algorithm is proposed.

5.1 Construction of Enveloping Hypersphere

Let us introduce the following designations:

P_k is a set of interpolation nodes $P_k = \{p_1, \ldots, p_k\}$ in m-dimensional space ($k > m$);

V_N is an N-dimensional simplex in m-dimensional space ($0 \le N \le m$) that is determined by $N + 1$ vertices from set P_k;

s is a test point due to localization, for which it is necessary to find a set V_m that forms Delaunay simplex with circumscribed sphere $\Omega(V_m)$ such that $s \in \Omega(V_m)$;

L_N is linear manifold spanned on the simplex V_N;

u_N is the test point s projection on the manifold L_N;

q_N is the center of a m-dimensional hypersphere that passes through all the points of V_N and contains test point s;

Z_N is a perpendicular dropped from the point q_N to L_N, i.e.,

$$Z_N = \{q_N + t(q_N - u_N), t \in (-\infty, \infty)\}.$$

A physical model of the algorithm consists in the process of inflation of a m-dimensional sphere. For the case $m = 2$, illustration of this process is presented on Fig. 2.

The process is initialized from the zero-radius sphere with a center in test point s. Then, it is "inflated" (its radius increases, and the center stays put) till its surface touches one of the points of set P_k (Fig. 2a). This point v_0 is the first vertex of the simplex required, and it generates set $V_0 = \{v_0\}$. Here, the center of the sphere is $q_0 = s$. Following inflation of the sphere consists of increasing its radius and shifting its center along the direction $\overrightarrow{v_0 q_0}$ so that point v_0 remains on the sphere surface. This process continues until the sphere touches one more point v_1 from V (Fig. 2b). Point v_1 is the second point of the simplex required, therefore $V_1 = V_0 \cup \{v_1\}$. The new center of the sphere is point q_1. Further sphere inflation consists of increasing the radius and shifting its center along the perpendicular to the line $\overrightarrow{v_0 v_1}$. This perpendicular passes through the point q_1 and its projection u_1 on $\overrightarrow{v_0 v_1}$. Inflation is continued up to the moment of touching the third point v_2 of V (Fig. 2c). This process which is illustrated for a two-dimensional case, is also easily generalized to $m > 2$.

Formally, the algorithm can be represented in the following way.

Hypersphere Inflation Algorithm

1) Initial step–construction of a 0-dimensional simplex ($N = 0$):

Find

$v_0 := \{v: \rho(v, s) \le \rho(V, s)\}$—the nearest to s point of V;

$V_0 := \{v_0\}$—the set of the simplex vertices;

$q_0 := s$—center of the $\Omega(V_0)$ hypersphere which touches v_0 and contains test point s in its interior;

L_0—0-dimensional manifold spanned on V_0 (it is just the point v_0);

$u_0 := v_0$—the projection of the point q_0 to L_0;

$Z_0 = \{q_0 + t(q_0 - u_0), t \in (-\infty, \infty)\}$—the perpendicular to L_0 that passes through the point q_0.

2) The sphere inflation step and construction of the next simplex of a greater dimensionality:

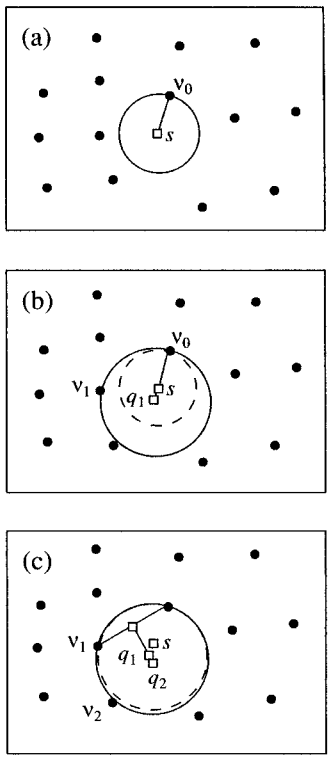

Fig. 2. Sphere inflation algorithm

Let a N-dimensional simplex V_N be constructed, as well as points q_N and u_N and manifolds L_N and Z_N associated with it.

Designate as

$$t(v) = \frac{[(v, v) - (v_0, v_0) - (q_N, v - v_0)]}{(q_N - u_N, v - v_0)}$$ the value

of the parameter that determines the location of the sphere center on the straight line Z_N that passes through all the points of V_N and some point v.

Then $\tau := \min\{t(v) | t(v) \ge 1, v \in V \backslash V_N\}$ is the parameter that determines the center of the sphere of minimum radius which passes through all the points of V_N and one more point of $V \backslash V_N$.

Find

$q_{N+1} := q_N + \tau(q_N - u_N)$—the proper center of this sphere;

v_{N+1}—point of $V \backslash V_N$ where the minimum of $t(v)$ is achieved, i.e., the next point which was touched by the inflating sphere;

$V_{N+1} := V_N \cup \{v_{N+1}\}$—vertices of new $(N + 1)$-dimensional simplex.

If $N + 1 = m$, the job is completed, else go to 3.

3) Constructing the projection of the sphere center on the simplex linear span:

Let u_{N+1} be the projection of q_{N+1} on the linear manifold L_{N+1} spanned on the simplex V_{N+1}. Let us

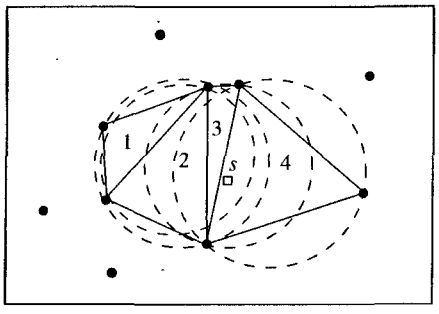

Fig. 3. Simplex tracing algorithm

express u_{N+1} as a linear combination of vertices of simplex V_{N+1}:

$$u_{N+1} := v_0 + \sum_{j=1}^{N+1} (v_j - v_0)\alpha_j,$$

where α_j is a solution to the system of linear equations

$$\sum_{j=1}^{N} (v_j - v_0, v_i - v_0)\alpha_j = (q_{N+1} - v_0, v_i - v_0),$$

$$i = 1, \ldots, N.$$

Let $N := N + 1$ and go to 2.

As a result of the realization of the algorithm, we get simplex V_m that has a circumscribed hypersphere $\Omega(V_m)$ containing a test point s and no points of V. The computational complexity of the algorithm is determined by the fact that on each of m steps, sorting of all the points of V is carried out, and we must solve a system of linear equations (point 3 of the algorithm) of dimensionality $N = 1, \ldots, m$. This yields an asymptotic estimate $O(mN) + O(m^4)$. Since in all practical problems m is of the order of ~10, the real time of the algorithm execution is quite acceptable.

5.2 Search of the Enveloping Simplex

In the general case, the simplex V_m constructed does not necessarily contain test point s in its interior. But since its circumscribed sphere contains this point, it can be supposed that the required simplex containing the test point is somewhere near V_m. The search of this simplex is carried out by a directed sorting of Delaunay simplexes by the proposed simplex tracing algorithm. The name of the algorithm can be justified because the sorting of simplexes is carried out along the direction from simplex V_m to point s. The idea of the algorithm is to construct, consequently, several adjacent Delaunay simplexes until we find one of them which envelops test point s. An illustration of this idea is presented in Fig. 3 for the case $m = 2$.

Let $(\lambda_0, \ldots, \lambda_m)$ be baricentric coordinates of the test point s in simplex V_m. It means that

$$s = \lambda_0 v_0 + \ldots + \lambda_m v_m,$$

$$\lambda_0 + \ldots + \lambda_m = 1.$$

This means that if all $\lambda_i \geq 0$, the point s lies inside simplex V_m. If this condition is not satisfied, we should pass from simplex V_m to some adjacent simplex and check the location of s relative to it. Passing to an adjacent simplex consists of the substitution of one of the vertices of simplex V_m to the other point from $V \backslash V_m$. In order to make this process purposeful, it is natural to substitute the vertex with baricentric coordinate λ_i that is negative with the maximum absolute value. This rule reminds one very much of the corresponding part of the linear programming simplex-method. A peculiarity of the simplex tracing is that the vertex introduced into the simplex (into the basis in terms of the linear programming simplex-method) is chosen in such a way that the new simplex generated would be a Delaunay simplex.

In the example in Fig. 3, simplex 1 does not contain test point s, but its circumscribed circumference contains it in its interior. Constructing, subsequently, simplexes 2, 3, and 4, we arrive at the simplex enveloping the test point (simplex 4).

Formally, the algorithm is outlined as follows.

Simplex Tracing Algorithm

Let point q be the center of an empty sphere of the simplex V_m.

1) Calculate baricentric coordinates $(\lambda_0, \ldots, \lambda_m)$ of test point s in simplex V_m by solving (2).

2) Find among the baricentric coordinates the minimum one, $\lambda^* = \min\{\lambda_i, i = 1, \ldots, m\}$. If $\lambda^* \geq 0$, the job is completed.

3) Let j^* be the index of the minimum baricentric coordinate. Eliminate vertex v_{j^*} from simplex V_m.

4) Look for a new vertex in order to include it in the simplex. In order to do that find the projection u of point q which is the center of sphere $\Omega(V_m)$, on the set $V_m \backslash \{v_{j^*}\}$ linear span. This is accomplished similarly to Step 3 of the sphere inflation algorithm.

The search of the introduced vertex is accomplished by sorting the vertices from set $V \backslash V_m$ that lie at the opposite side of the simplex V_M hyperface, which is generated by its vertices $V_m \backslash \{v_{j^*}\}$ relative to the point v_{j^*}.

Direct search of the introduced vertex is carried out similarly to Step 2 of the sphere inflation algorithm. The vertex found together with the points from $V_m \backslash \{v_{j^*}\}$ forms the new simplex V_m. When it is constructed, we should assume $r := r + 1$ and go to Step 1.

As a result of executing the algorithm a Delaunay simplex that contains the test point in its interior is

Number of iterations of the simplex tracing algorithm

Space dimensionality	Number of interpolation nodes			
	$k = 50$	$k = 100$	$k = 150$	$k = 200$
$m = 2$	0.303	0.398	0.303	0.300
$m = 3$	0.979	0.927	0.840	0.987
$m = 4$	1.315	1.525	1.506	1.664
$m = 5$	1.592	2.164	2.418	2.564
$m = 6$	–	2.964	3.412	3.190
$m = 7$	–	–	4.230	4.280

found, and baricentric coordinates of this point in the simplex are calculated.

The calculation complexity of the suggested algorithm is determined by the fact that the number of sorted adjoint simplexes, in the worst case, can be commensurable with their total number which, as it was mentioned, has an order of $O\left(n^{\frac{m}{2}}\right)$, where n is the number of points in set V. This algorithm has, in practice, quite acceptable consumption time, as it is demonstrated by the results of the computational experiment presented in Table 1. It presents an average number of iterations (for 1000 experiments) of the simplex tracing algorithm for space dimensionality $m = 2$–7 and interpolation nodes number $k = 50$–200.

5.3 Linear Interpolation in the Enveloping Simplex

As mentioned above, if the test point should get into one of the Delaunay triangulation simplexes with certainty, it necessarily should lie inside the set of the vertices convex hull. Since any point of m-dimensional integer-valued cube I^m can be a test point, it is necessary to include all 2^m vertices of cube R^m in the set of interpolation nodes V, in addition to poles $p_1, \ldots, p_k$, for which the operator-teacher has specified *RGB*-colors.

Since values of function T should be defined at the interpolation nodes, it is necessary to specify some *RGB*-colors to the R^m cube vertices. The calculation experiments that were executed showed that it is quite acceptable to assign the black *RGB*-color (0, 0, 0) to these vertices.

Let us assume that a Delaunay simplex, which contains test point s, is found and that its baricentric coordinates $(\lambda_0, \ldots, \lambda_m)$ in this simplex are determined. Then, we determine the color of point s as

$$T(s) = \lambda_0 T(v_0) + \ldots + \lambda_m T(v_m),$$

where $v_0, \ldots, v_m$ are simplex vertices.

Thus, the suggested algorithm allows one to select colors for clusters $W_{k+1}, \ldots, W_M$ using multidimensional linear interpolation, the colors of clusters $W_1, \ldots, W_k$ specified by the operator-teacher being given.

6. CONCLUSION

The method suggested was implemented and tested when developing the model-prototype of the automated working place (AWP) of the multizone survey operator. The design of the AWP was implemented for the survey automatization including stages of learning and operation.

On the learning stage, this process includes:

—the viewing of educational gray single-zone pictures;

—automatic clusterization of the multizone palette;

—partial manual coloring of the clustered palette;

—automatic total coloring of the clustered palette;

—the viewing of visual color images of educational multizone pictures.

On the operating stage viewing of visual images of current multizone pictures in a real time scale of survey is implemented.

REFERENCES

1. Schreiber, T., Clustering for Data Reduction and Approximation, Proc. Conf. on Computer Graphics and Visualization "Graphicon 93", vols. 1, 2, St. Petersburg, 1993.
2. Preparata, F.P. and Shamos, M.I., *Computational Geometry*, Springer, 1985.
3. Kanaganathan, S. and Goldstein, N.B., Comparison of Four-Point Adding Algorithms for Delaunay-Type Three Dimensional Mesh Generators, *IEEE Trans. of Magnetics*, 1991, vol. 27, no. 3.
4. Avis, D., and Bhattacharya, B.K., Algorithms for Computing d-Dimensional Voronoi Diagrams and Their Duals, *Advances in Computing Research*, 1983, vol. 1, pp. 159–180.

Leonid M. Mestetskii. Born 1949. Graduated from Moscow State University in 1971. Received degree of Doctor of Technical Sciences in 1992. Professor at Tver University. Scientific interests: operations research, simulation, computational geometry and image analysis. Author of 42 papers.

Konstantin V. Rudakov. Born 1954. Graduated from the Moscow Institute of Physics and Technology in 1978. Received degree of Doctor of Physics and Mathematics in 1992. Chief of the Department of Computer Center of the Russian Academy of Sciences. Author of 32 papers. Corresponding Member of the Russian Academy of Sciences.