A Volume Shader for Quantum Voronoi Diagrams inside the 3D Bloch Ball

By Frank NIELSEN / Frank.Nielsen@acm.org
Ecole Polytechnique (France)
Sony Computer Science Laboratories, Inc. (Japan)
Last revised, August 6th 2008.

Introduction and preliminaries

In quantum information theory [NielsenChuang'00], particle state distributions are analyzed probabilistically by means of *density matrices* X.
A d-level system is characterized by a $d \times d$ square matrix with complex coefficients that satisfies the following three core properties:

1. X is equal to its conjugate transpose: $X = \left(X^*\right)^T$;That is, density matrix X is *Hermitian*. (The conjugate of a matrix is the matrix with all conjugate elements.)

2. X has *unit trace*, where the trace of a matrix is defined as the sum of its diagonal coefficients: $\mathrm{trace}(X) = \sum_i X_{i,i} = 1$. This means that the trace of X has *no* imaginary part.
3. X is *semi-positive definite*. That is, we have for all d-dimensional vector x, the following property: $x^T X x \geq 0$.

When studying systems representating statistically one quantum bit (or *1-qubit* for short), the above three conditions yield the following parameterized set of 2x2 density matrices:

$X = \left\{ \frac{1}{2} \begin{bmatrix} 1+z & x-iy \\ x+iy & 1-z \end{bmatrix}, x^2+y^2+z^2 = 1 \right\}$, where i denote the imaginary complex number ($i^2 = -1$). Note that unfortunately, there are no such simple equivalent representations of matrix densities for higher level systems since the semi-positive definite property is much more delicate to handle in higher dimensions.
It follows that states of 1-qubits can be represented equivalently by a triplet of *real* numbers (x,y,z): A 3D point inside a unit ball centered at the origin.
In quantum theory, this unit ball bears the name of the *Bloch ball* [NielsenChuang' 00,

Bloch], where a special parameterization proposed by physicist Felix Bloch is used to describe the quantum states [NielsenChuang' 00]. Note that in classical information theory, the state of a bit either *true* or *false*. For a quantum bit, its "state" is rather a superposition of states with respective probability given by the Hermitian density matrix.

Pure states X have degenerated matrices of *rank 1* (that is, matrix X is not invertible). Pure states correspond to the density matrices parameterized by triplets lying on the surface of the Bloch ball: The *Bloch sphere*.

Mixed states have full rank 2: They correspond to triplets (x, y, z) falling strictly inside the Bloch sphere.

Von Neumann quantum entropy and its relative entropy divergence

In classical information theory [NielsenChuang' 00], one uses the Shannon entropy on probability distributions p to define the following quantity that measures the degree of uncertainty of a distribution: $H(p) = \sum_i p_i \log \frac{1}{p_i} = -\sum_i p_i \log p_i$. The distance between two statistical distributions p and q is then defined using the asymmetric Kullback-Leibler divergence: $KL(p \| q) = \sum_i p_i \log \frac{p_i}{q_i} \geq 0$.

Similarly, Von Neumann generalized this entropy measure and relative entropy distortion measure ("distance") to get respectively the quantum entropy and quantum divergence as:

$H(X) = -Tr(X \log X)$ and $I(P \| Q) = Tr(P(\log P - \log Q)) \geq 0$, where the logarithm of an Hermitian matrix is defined from the matrix *spectral decomposition*.

Let the spectral decomposition of X be $X = V \times \mathrm{Diag}(\lambda) \times V^*$, where

$$\mathrm{Diag}(\lambda) = \begin{bmatrix} \lambda_1 & 0 & \cdots & \cdots & 0 \\ 0 & \lambda_2 & 0 & \cdots & \vdots \\ \vdots & 0 & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & 0 & \lambda_d \end{bmatrix}$$ is a diagonal matrix of d eigenvalues.

Then the logarithm matrix of X is defined as $X = V \times \mathrm{Diag}(\log \lambda) \times V^*$.

A remarkable property between classic and quantum theory is given by the following inequality:

$I(X \| X') \geq KL(\lambda, \lambda') \geq 0$, where λ and λ' are the respective eigenvalues of matrix X and X'.

These information-theoretic divergences are not symmetric, nor do they satisfy the triangular inequality of metrics. Furthermore, the quantum divergence I(.||.) is not well-defined when right-side matrix X' is not full rank since the logarithm of the matrix cannot be properly computed (one of the eigenvalues is equals to zero).

Quantum Voronoi Diagrams

Given a set of n density matrices $\{X_1,...,X_n\}$, called *sites* or *generators*, the *Voronoi diagram* of these quantum states defines a *partition* of the space into elementary *Voronoi cells*. The Voronoi cell of a given site is defined by the set of density matrices closer to that site than to any of the other sites. Since the divergence I(P||Q) is asymmetric, we consider in the following three kinds of *Voronoi cells* as follows:

1. The right-side Voronoi cell: $Vor_R(X_i) = \{X \mid I(X_i \| X) \leq I(X_j \| X)\}$,
2. The left-side Voronoi cell: $Vor_L(X_i) = \{X \mid I(X \| X_i) \leq I(X \| X_j)\}$, and
3. The symmetrized Voronoi cell obtained by symmetrizing the divergence: $Vor_S(X_i) = \{X \mid I(X \| X_i) + I(X_i \| X) \leq I(X \| X_j) + I(X_j \| X)\}$

We are interested in computing and visualizing interactively these quantum Voronoi diagrams using GPU pixel shaders. Exploring interactively these structures proved to be very helpful for researchers to gain intuition. Indeed, interactive visualization helps in revealing structural properties, and may thus be useful in a number of setting including, for example, to get a visual feeling of the additivity property of Holevo's channel capacity [Holevo'04] that is conjectured to hold by not yet proved.

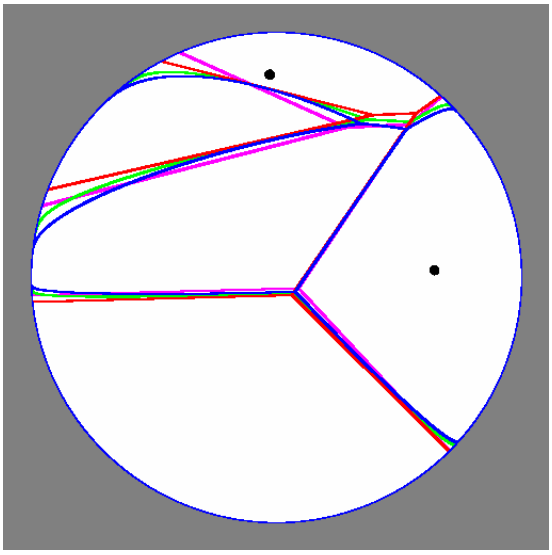
In the remainder, we distinguish two fundamental cases:

1. The quantum Voronoi diagrams of mixed density matrices, and
2. The quantum Voronoi diagram of pure density matrices.

Figure 1 illustrates a cross section of the Bloch ball showing the quantum Voronoi diagram of mixed density matrices on that cutting plane. The color figure is available

from the web page indicated in the conclusion. The red lines depict the Voronoi affine separators (right-type), the blue lines show the dual curved Voronoi diagram, the green lines show the symmetrized curved Voronoi diagram [NielsenBoissonnatNock' 07]. For comparison, we also added the regular affine Euclidean Voronoi diagram in purple color too. A video showing an animation of the cross-section of the Bloch ball is available at the URL mentioned in the conclusion. The video allows one to observe the relationships of the internal quantum and Euclidean Voronoi diagrams.

*** INSERT FIGURE 1 HERE ***



*** Caption 1

A cross section of the Bloch ball showing the various quantum and Euclidean Voronoi diagrams. The Voronoi generators are rasterized using small radii balls. Here, we see two of these "site balls" intersecting the cross-section plane.

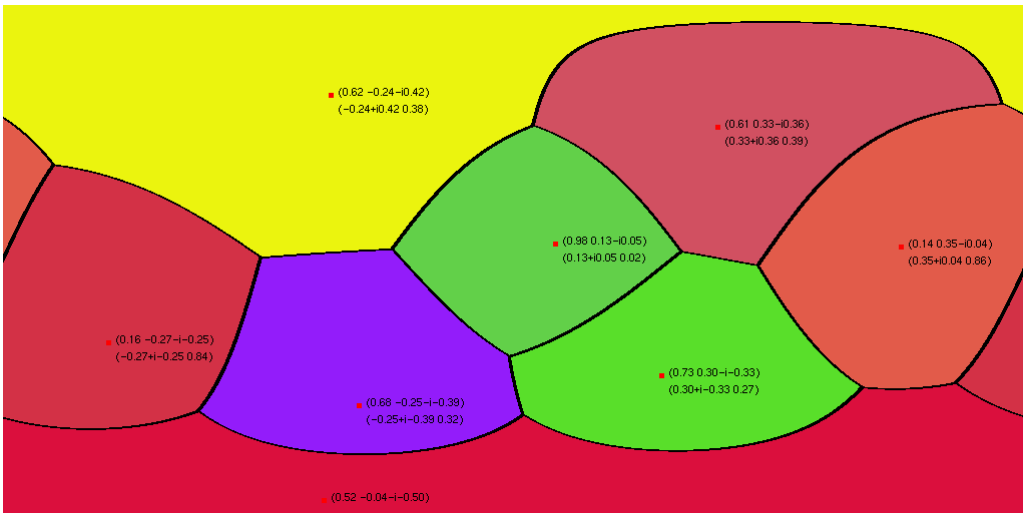
***

Nielsen and Nock [NielsenNock08] showed that the quantum Voronoi diagram is a kind of generic Bregman Voronoi diagram for the appropriate Bregman generator acting on Hermitian matrices. It follows that the right-side Voronoi diagram has affine bisectors and can be computed as a power diagram in disguise [NielsenBoissonnatNock07]. The left-side Voronoi diagram can be computed from a right-side Voronoi diagram using a dual divergence defined by the Legendre convex conjugate [NielsenBoissonnatNock07].

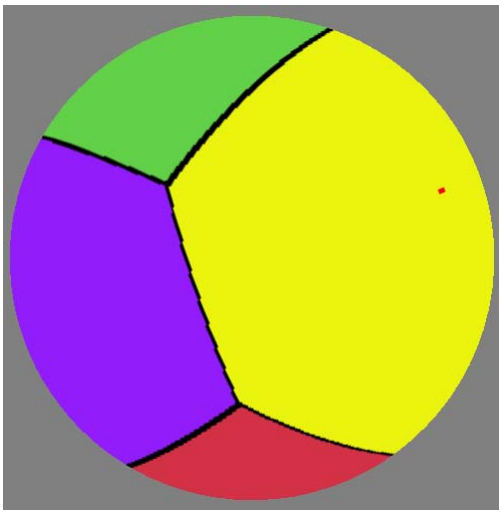
Quantum Voronoi Diagrams for Pure States

Kato et al. [Katolmai' 07] and Nielsen and Nock [NielsenNock' 08] proved that the

Voronoi diagram of pure density matrices lying on the Bloch sphere is equivalent to a spherical Voronoi diagram with respect to the geodesic distance (proportional to the angle defined by the pure states). Alternatively, this spherical Voronoi diagram can also be computed by intersecting the 3D Euclidean Voronoi diagram with the sphere.



(a)


(b)

Insert Figure 2a and Figure 2b here

CAPTION 2: Visualizing using the GPU the quantum Voronoi diagram of 1-qubit pure states. Voronoi cells are (a) annotated with their corresponding density matrices on the latitude-longitude map, and (b) can be visualized on the Bloch sphere.

*** END OF FIGURE 2***

We compute this quantum Voronoi diagram on the Bloch sphere by choosing a

latitude-longitude representation of the pure states (also called theta-phi representations using the 3D point spherical coordinates).
The GPU code for computing the quantum Voronoi diagram of pure states on the Bloch sphere consists in tracing the border of the Voronoi cells on this latitude-longitude environment map, and the use texture mapping to wrap that environment texture onto the Bloch sphere.
The Cg per-pixel shader that allows the GPU to rasterize the Voronoi diagram onto the latitude-longitude map is given below. The shader proceeds as follows: It basically converts the 2D texture position pos passed as an argument into an equivalent 3D point lying on the sphere using a spherical to Cartesian conversion procedure. Namely, the shader calculates the distance between a given site and a point on the latitude-longitude texture map by first converting them into equivalent 3D points on the sphere using function Spherical2Cartesian, and then calculating their distance on the sphere using function DistanceSphere. For all pixels pos, the index of the closest Voronoi generator is found using function Winner and the color of the pixel is assigned according to the mapping function WinnerColor.

```
// Quantum Voronoi diagram on the Bloch sphere
// (c) 2007 2008 Frank Nielsen
// Density matrices: pure states
// (Quantum pure state Voronoi =Spherical Voronoi)

// Number & coordinates of Voronoi sites (pure 1-qubit systems)
#define MAXN 8
float2 position[MAXN];

// Bounding box of domain X
float minx,maxx,miny,maxy;

// Border thickness of Voronoi cells
float s=1.0/300.0;

// Max textures [0,1] coordinate to domain X: LERP rescaling
float2 ToDomain(float2 p)
{return float2(minx+(maxx-minx)*p[0],miny+(maxy-miny)*p[1]);}
```

```
// Convert latitude longitude to 3D xyz Cartesian coordinate
float3 Spherical2Cartesian(float2 tp)
{float3 xyz;
xyz[0]=cos(tp[1])*sin(tp[0]);
xyz[1]=sin(tp[1]);
xyz[2]=cos(tp[1])*cos(tp[0]);
return xyz;
}

float DistanceSphere(float2 tp, float2 tq)
{float3 P, Q;
float angle;
P=Spherical2Cartesian(tp);
Q=Spherical2Cartesian(tq);
angle=acos(P[0]*Q[0]+P[1]*Q[1]+P[2]*Q[2]);
return abs(angle);
}

// Reports the index of the closest point
int Winner(float2 p)
{
int i,winner;
float dist,mindist=1.0e5;

dist=DistanceSphere(p,position[0]);
mindist=dist;
for(i=1;i<MAXN;i++)
{
       dist=DistanceSphere(p,position[i]);
       if (dist<mindist)
              {mindist=dist;winner=i;}
}
return winner;
}

// A few colors
```

```
float3 c1=float3(0.575426,0.111484,0.979553);
float3 c2=float3(0.348125,0.875674,0.16422);
float3 c3=float3(0.88638,0.357219,0.288858);
float3 c4=float3(0.819391,0.31315,0.378887);
float3 c5=float3(0.859276,0.0589618,0.240242);
float3 c6=float3(0.382763,0.815882,0.287912);
float3 c7=float3(0.920255,0.955535,0.0535295);
float3 c8=float3(0.827662,0.193457,0.273507);

float3 WinnerColor(int p)
{float3 c;
if (p==0) c=c1; if (p==1) c=c2; if (p==2) c=c3;
if (p==3) c=c4; if (p==4) c=c5; if (p==5) c=c6;
if (p==6) c=c7; if (p==7) c=c8;
return c;
}

// Voronoi sphere on latitude longitude map
float3  VoronoiSphere(float2 pos: TEXCOORD0) : COLOR0
{
int index, indexx, indexy;
float2 posx,posy; // position of the shader pixel
float3 color;

// pos is in [0,1]
posx=ToDomain(pos+float2(s,0));
posy=ToDomain(pos+float2(0,s));
pos=ToDomain(pos);

index=Winner(pos); indexx=Winner(posx); indexy=Winner(posy);

if ((index!=indexx)||(index!=indexy))
       color=float3(0,0,0); // black
              else color=WinnerColor(index);
return color;
}
```

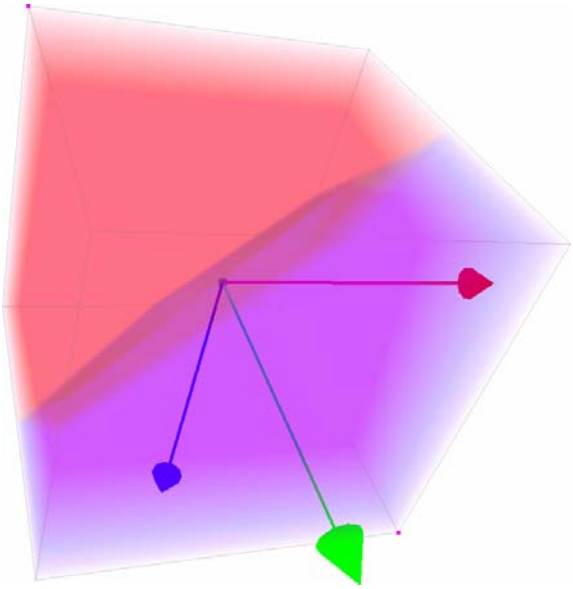
The C++ code and accompanying Visual C++ project is available in the program directory GPUQuantumVoronoiPure. A trackball user interface is provided for exploring interactively the Voronoi structures of these pure state density matrices.

Quantum Voronoi Diagrams for Mixed States

The quantum Voronoi diagram of density matrices encoding mixed states is far more difficult to design since we need to visualize inside the Bloch ball structures using transparency. We design a volume shader that rasterizes the Bloch ball using RGB color and alpha channel attributes. This allows one to visualize not only the mixed state sites by transparency but more importantly allows us to display the various quantum right/left/symmetrized Voronoi diagrams since they are all distinct for the case of mixed state generators. To explain the basic programming principle, let us for now consider the regular Euclidean Voronoi diagram of 3D points. We design a pixel shader that given a quad to texture will rasterize the pixel by choosing color according to the color assigned to its nearest generator. We can also clip this Euclidean Voronoi diagram to the unit sphere, and further choose to draw Voronoi bisectors (the borders of Voronoi cells) using simple tests to check whether neighborhood pixels have the same closest Voronoi generator or not. If not, the pixel belongs to the boundary of the Voronoi diagram: It is part of a Voronoi bisector. Figure 3 illustrates these software volume shaders using the traditional Euclidean distance to start with.

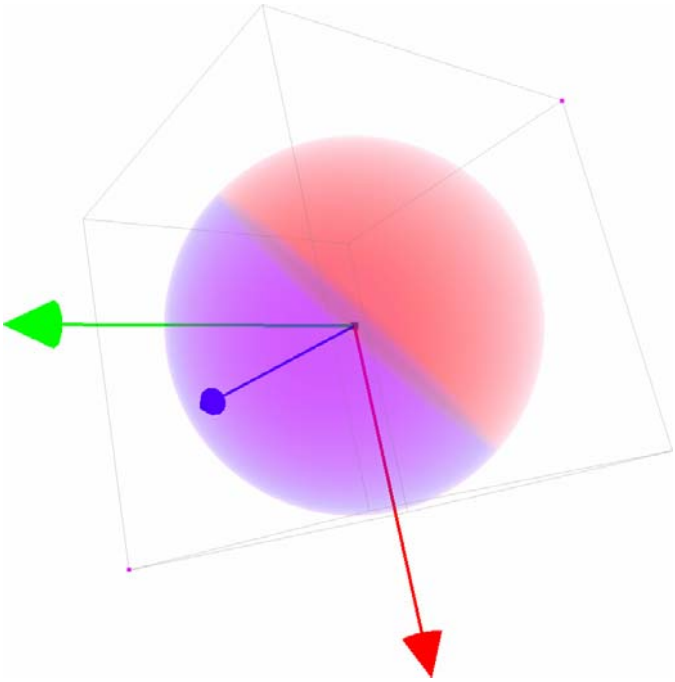
*** Insert Figure 3 here**

*** CAPTION 3: The left figure (a) visualizes the Euclidean Voronoi bisector of two antipodal generators. The right figure (b) similarly renders volumetrically these Voronoi cells delimited by the Euclidean bisector. These cells are all clipped with a unit sphere.



(a)

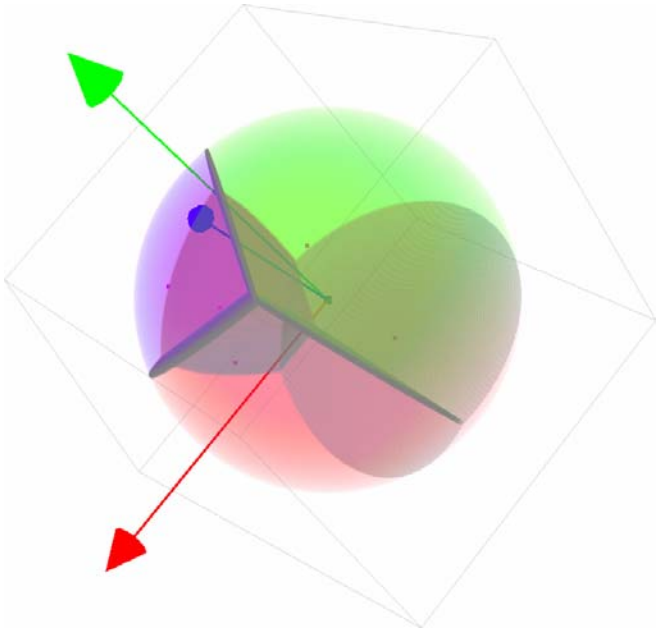
(b)



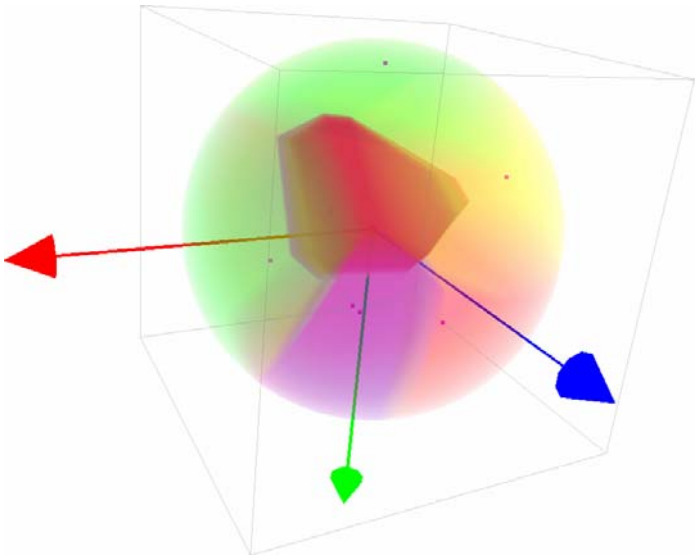
*** end of figure 2*

We can furthermore choose to color each cell of the Voronoi diagram with an attribute color and display the bisector “walls” using another specially chosen color. Moreover, a target cell may be displayed with a different transparency attribute to emphasize its location and boundaries with respect to the other cells. Figure 4 illustrates the interactive visualization that modern GPUs offer.

*** INSERT FIGURE 4 HERE ***



(a)



(b)

CAPTION: Using a pixel shader to render a volumetric Voronoi diagram with transparency attribute (a). The color of the Voronoi boundaries can be freely chosen to emphasize the boundaries of Voronoi cells. (Right) A Euclidean 3D Voronoi diagram using a different transparency mode for the color of the target cell reaveals the convex polyhedron structure of the internal Voronoi region.

*** END OF FIGURE 4***

The pixel shader fragment for performing these tasks is given below:

```
// (c) August 2008, Frank Nielsen
//
float radius=1.0;
float centerx=0.5, centery=0.5,  centerz=0.5;
float zzz; // slice level

float4 colorBloch=float4(0.0,0.0,0.0,0.00); // ambiance color

float4 colorRight=float4(0.5,0.0,0.0,0.01);
float4 colorLeft=float4(0.5,0.0,0.0,0.01);
float4 colorSymmetrized=float4(0.5,0.0,0.0,0.01);
float4 colorEuclidean=float4(1,0.0,0,1);
float4 colorBorder=float4(0.5, 0.5 ,0.5 ,1);

float4 colorWhite=float4(0,0.0,0,0);

#define  tt 0.01

float4 c0=float4(0.827662,0.193457,0.27350,tt);
float4 c1=float4(0.575426,0.111484,0.979553,tt);
float4 c2=float4(0.348125,0.875674,0.16422,tt);
float4 c3=float4(0.88638,0.357219,0.288858,tt);
float4 c4=float4(0.819391,0.31315,0.378887,tt);
float4 c5=float4(0.859276,0.0589618,0.240242,tt);
float4 c6=float4(0.382763,0.815882,0.287912,tt);
float4 c7=float4(0.920255,0.955535,0.0535295,tt);

float4 WinnerColor(int p)
{
float4 c;
if (p==0) c=c0; if (p==1) c=c1; if (p==2) c=c2;
if (p==3) c=c3; if (p==4) c=c4; if (p==5) c=c5;
if (p==6) c=c6; if (p==7) c=c7;
return c;
}
```

```
float s=0.01;
#define MAXN 8
float3 position[MAXN];

float SqrL2(float x1, float y1, float z1, float x2,float y2,float z2)
{return (x2-x1)*(x2-x1)+(y2-y1)*(y2-y1)+(z2-z1)*(z2-z1);}

double Distance(float3 p, float3 q)
{
double x1=p[0],y1=p[1],z1=p[2];
double x2=q[0],y2=q[1],z2=q[2];
return (x2-x1)*(x2-x1)+(y2-y1)*(y2-y1)+(z2-z1)*(z2-z1);
}

int Winner(float3 p)
{
int i,winner;
float dist,mindist=1.0e5;

dist=Distance(p,position[0]);
mindist=dist;
for(i=1;i<MAXN;i++)
{
       dist=Distance(p,position[i]);
       if (dist<mindist)
              {mindist=dist;winner=i;}
}
return winner;
}



float4  QuantumVoronoi(float2 pos: TEXCOORD0) : COLOR0
{
float rr=SqrL2(centerx,centery,centerz, pos.x,pos.y,zzz);
int index, indexx, indexy;
```

```
float3 npos, nposx,nposy; // position of the shader pixel

npos=float3(pos[0],pos[1],zzz);
index=Winner(npos);

if (rr<0.5*0.5)
{
       return WinnerColor(index);
}
else
       return colorWhite;
}
```

This shader is called whenever texturing 2D quads slicing the unit cube.
We render 200 z-slices for the above figures using the following OpenGL/Cg code:

```
// 200 slices
for(float l=1.0;l>=0.0;l-=0.005)
{
glEnable(GL_TEXTURE_2D);
cgGLEnableProfile(FProfile); CheckCgError();
cgGLBindProgram(FProgram); CheckCgError();
cgGLSetParameterArray3f(CGposition,0,MAXN,position);
cgGLSetParameter1f(CGzzz,l);
QuadZ(l);
cgGLDisableProfile(FProfile);
glDisable(GL_TEXTURE_2D);
}
```

For quantum Voronoi diagrams with mixed state generators, we need to define explicitly in the shader the Von Neumann relative entropy between Hermitian matrix P and Hermitian matrix Q (encoded respectively by 3D points p and q strictly falling inside the Bloch ball). The formula is quite intricate but can be simplified by using the radii of p and q and their dot product $< p,q >= p_x q_x + p_y q_y + p_z q_z$. We calculate the divergence [Katolmai' 07] as:

$$I(P \| Q) = \frac{1+r_p}{2}\log\frac{1+r_p}{2} + \frac{1-r_p}{2}\log\frac{1-r_p}{2} - \frac{1}{2}\log\frac{1-r_q^{\,2}}{4} - \frac{< p,q >}{2r_q}\log\frac{1+r_q}{1-r_q}.$$

Since the GPU shaders allow one to render at interactive frame rate, we may also animate the generators by assigning them a velocity and observe that the quantum Voronoi diagrams of mixed states tend in the limit case of pure states to the regular Euclidean Voronoi diagram. The program QuantumVoronoiDiagramMixedStates.cpp provides a user interface for choosing and animating Voronoi generators.

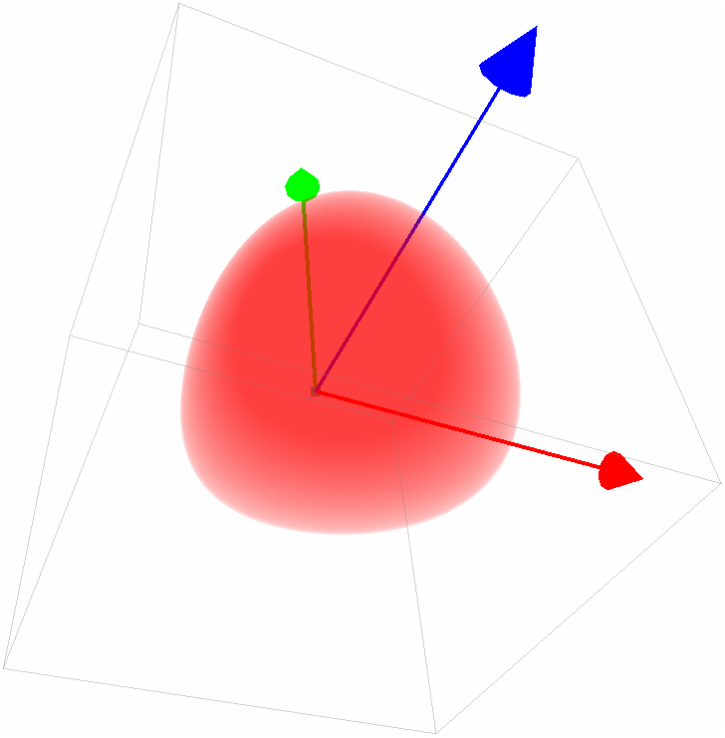
Quantum channel and Holevo's capacity

A quantum channel is modeled mathematically as a *linear transform* T.
That is, a quantum channel is described by an affine map T that maps quantum states to other quantum states. The geometric effect of a channel is to map the Bloch ball to a deformed 3D ellipsoid by T, that is always strictly contained inside the Bloch ball. The Holevo capacity C(T) of this channel T is defined as the smallest radius of ball enclosing the ellipsoid [NielsenNock' 08] as follows:

$C(T) = \inf_P \sup_Q I(T(Q) \| T(P))$, where I(.||.) denotes the quantum relative entropy.

Therefore, the Holevo capacity of a channel amounts to geometrically computes the smallest enclosing ball with respect to I(.||.) of the deformed Bloch ball. We provide several approximation algorithms in [NielsenNock' 08].
Figure 5 shows such a relative entropy ball visualized using the same volume technique: Slicing the unit bounding box and rasterizing using a pixel fragment shader.

*** INSERT FIGURE 5 here ***



*** Caption: Visualizing the smallest ball enclosing the deformed Bloch ball using a software volume rendering based on a GPU pixel fragment.
*** End of FIGURE 5 ***

Concluding Remarks

We have presented several pixel shaders for rendering the quantum Voronoi diagrams of 1-qubits represented by their density Hermitian matrices or equivalement 3D points in the Bloch ball. The quantum Voronoi diagram of pure state sites amounts to compute a Voronoi diagram on the surface defined with respect to the geodesic arcs. This can be rasterized interactively on the GPU using a pixel shader that prepares a latitude-longitude environment map that is then textured on the Bloch sphere. The quantum Voronoi diagram of mixed state sites requires to visualize the inner structure using transparency. This is done by implementing a software volume renderer using Cg: We call a parameterized Cg pixel shader for every X, Y, Z quad plane that is rendered. This provides a nice volumetric visualization of the various quantum Voronoi diagrams that can then be interactively explored.

A home page gathering all materials including Cg/C++source codes and a video of a quantum Voronoi diagram sliced are available at the following web page:

http://www.lix.polytechnique.fr/Labo/Frank.Nielsen/QVD/

References

[Bloch] Bloch Sphere, Wikipedia, retrieved in August 2008/08/06 http://en.wikipedia.org/wiki/Bloch_sphere

[Holevo'04] Additivity of classical capacity and related problems, 2004. http://www.imaph.tu-bs.de/qi/problems/10.html

[KatoImai'07] Kimikazu Kato, Hiroshi Imai, and Keiko Imai, Error Analysis of a Numerical Calculation about One-Qubit Quantum Channel Capacity, 4th International Symposium on Voronoi Diagrams, pp. 265-269, 2007.

[NielsenBoissonnatNock' 07] Frank Nielsen, Jean-Daniel Boissonnat and Richard Nock, On Bregman Voronoi diagrams, ACM-SIAM Symposium on Discrete Algorithms, SODA, 2007.

[NielsenChuang'00] Michael A. Nielsen and Isaac L. Chuang, Quantum Computation and Quantum Information, Cambridge University Press, ISBN 978-0521635035, 2000.

[NielsenNock'08] Frank Nielsen and Richard Nock, Quantum Voronoi Diagrams and Holevo Channel Capacity for 1-Qubit Quantum States, IEEE International Symposium on Information Theory (ISIT), 2008.