COMPSCI 716 S2 T - Assignment 4
Solution Hints

Computer Science

This assignment is worth 10% of your final mark for COMPSCI 716 S2 T.

1. *Data Transformation and Reconstruction* *(14 Marks)*

Download the zip-file `Ass4Q1.zip` from the assignment webpage. The enclosed program `InterpolatedSurface.cpp` draws the function

$$f(x,y) = -0.5xy + 0.2e^{x} - 0.1\sin(2\pi y)$$

within the interval [0,1] ×[0,1] using a Gouraud shaded surface. You can toggle the visibility of the surface by pressing the '1' key.

Assume the function *f(x,y)* is sampled such that $f_{ij} = f(0.25i, 0.25j)$.

(a) [3 marks] Compute the bilinear interpolant of this surface within the interval [0,1] ×[0,1] and draw it as a wire frame mesh. Please represent each bilinear surface patch with 3x3 polygons, i.e. the bilinearly interpolated surface is represented by 12 x 12 polygons. Modify the program such that the visibility of the bilinearly interpolated surface is toggled with the '2' key.

Solution hints: The bilinear interpolant at a point (x,y) for a patch (i,j) can be computed as follows:

```
// sample points
const int NUM_SAMPLES=5;
float p[NUM_SAMPLES][NUM_SAMPLES];

// computes p(x,y) for the bilinear patch
// with the vertices (i,j), (i+1,j), (i,j+1), (i+1,j+1)
inline float patch(int i, int j, float x, float y)
{
        x=(NUM_SAMPLES-1)*x-i;          // scale x to [0,1] within patch
        y=(NUM_SAMPLES-1)*y-j;          // scale y to [0,1] within patch
        return p[i][j]*(1-x)*(1-y)+p[i+1][j]*x*(1-y)
                +p[i][j+1]*(1-x)*y+p[i+1][j+1]*x*y;
}
```

(b) [3 marks] Compute the surface gradient of *f(x,y)* at each sample points analytically and visualize it using thin blue cylinders. Modify the program such that the visibility of these cylinders is toggled with the '3' key. What is the value for the gradient at the point (0.5, 0.5)? Write your answer into the document `Ass4Answers.doc`.

Solution hints: The gradients are obtained by differentiating the function *f(x,y)*.

```
inline float dfdx(float x, float y){ return -0.5*y+0.2*exp(x); }
inline float dfdy(float x, float y){ return -0.5*x-0.1*cos(2*Pi*y)*2*Pi;}
```

(c) [4 marks] Approximate the surface gradient of the bilinear interpolant of *f(x,y)* at each sample points by computing the surface gradients of all bilinear patches sharing that point and averaging these value. Visualize these gradients using thin green cylinders. Modify the program such that the visibility of these cylinders is toggled with the '4' key. What is the value for the gradient computed with this method at the point (0.5, 0.5)? Write your answer into the document `Ass4Answers.doc`.

Solution hints: The gradients are obtained by differentiating the bilinear interpolant. Since we scale the parameters to the interval [0,1] we have the multiply the results with the inverse of the grid size.

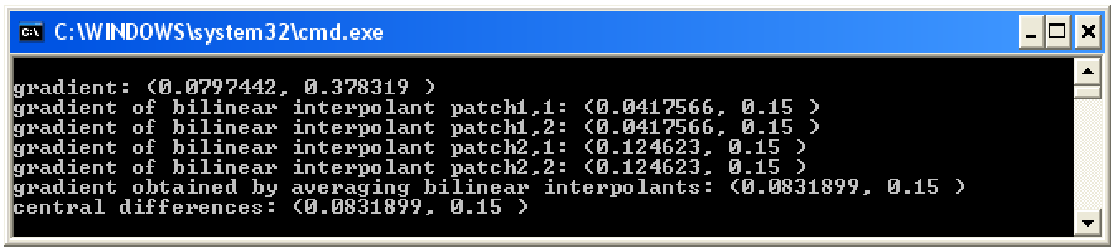
```
// computes derivative in x-direction for the bilinear patch with the vertices
// (i,j),(i+1,j), (i,j+1), (i+1,j+1)
inline float dpdx(int i, int j, float x, float y)
{
 x=(NUM_SAMPLES-1)*x-i;                // scale x to [0,1] within patch
 y=(NUM_SAMPLES-1)*y-j;                // scale y to [0,1] within patch
 return (NUM_SAMPLES-1)*(p[i][j]*(-1)*(1-y)+p[i+1][j]*(1-y)+
                          p[i][j+1]*(-1)*y+p[i+1][j+1]*y);
}

// computes derivative in y-direction for the bilinear patch with the vertices
// (i,j),(i+1,j), (i,j+1), (i+1,j+1)
inline float dpdy(int i, int j, float x, float y)
{
 x=(NUM_SAMPLES-1)*x-i;                // scale x to [0,1] within patch
 y=(NUM_SAMPLES-1)*y-j;                // scale y to [0,1] within patch
 return (NUM_SAMPLES-1)*(p[i][j]*(1-x)*(-1)+p[i+1][j]*x*(-1)+
                         p[i][j+1]*(1-x)+p[i+1][j+1]*x);
}
```

(d) [4 marks] Approximate the surface gradient of the bilinear interpolant of *f(x,y)* at the sample points using central differences and visualize it using thin green cylinders. Modify the program such that the visibility of these cylinders is toggled with the '5' key. What is the value for the gradient computed with this method at the point (0.5, 0.5)? Write your answer into the document `Ass4Answers.doc`.

Solution hints: The value for the gradient at the point (0.5, 0.5) computed with the different methods is:

```
// Part (b): compute analytic gradient
cout << "\ngradient: (" << dfdx(0.5,0.5) << ", " << dfdy(0.5,0.5) << " )";
// compute analytic gradient of bilinear interpolant
cout << "\ngradient of bilinear interpolant patch1,1: ("
     << dpdx(1,1,0.5,0.5) << ", " << dpdy(1,1,0.5,0.5) << " )";
cout << "\ngradient of bilinear interpolant patch1,2: ("
     << dpdx(1,2,0.5,0.5) << ", " << dpdy(1,2,0.5,0.5) << " )";
cout << "\ngradient of bilinear interpolant patch2,1: ("
     << dpdx(2,1,0.5,0.5) << ", " << dpdy(2,1,0.5,0.5) << " )";
cout << "\ngradient of bilinear interpolant patch2,2: ("
     << dpdx(2,2,0.5,0.5) << ", " << dpdy(2,2,0.5,0.5) << " )";
// Part (c): compute average of analytic gradients of bilinear interpolant
cout << "\ngradient obtained by averaging bilinear interpolants: (";
cout << (dpdx(1,1,0.5,0.5)+dpdx(1,2,0.5,0.5)
        +dpdx(2,1,0.5,0.5)+dpdx(2,2,0.5,0.5))/4 << ", ";
cout << (dpdy(1,1,0.5,0.5)+dpdy(1,2,0.5,0.5)
        +dpdy(2,1,0.5,0.5)+dpdy(2,2,0.5,0.5))/4 << " )";
// Part (d): compute surface gradient using central differences
cout << "\ncentral differences: ("
     << (p[3][2]-p[1][2])/0.5 << ", " << (p[2][3]-p[2][1])/0.5 << " )";
```

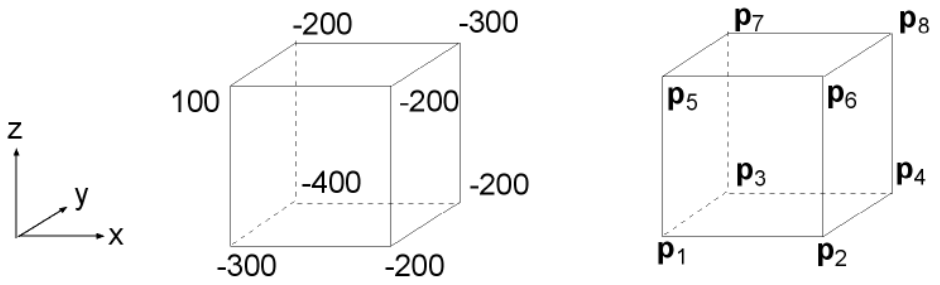

C:\WINDOWS\system32\cmd.exe

```
gradient: (0.0797442, 0.378319 )
gradient of bilinear interpolant patch1,1: (0.0417566, 0.15 )
gradient of bilinear interpolant patch1,2: (0.0417566, 0.15 )
gradient of bilinear interpolant patch2,1: (0.124623, 0.15 )
gradient of bilinear interpolant patch2,2: (0.124623, 0.15 )
gradient obtained by averaging bilinear interpolants: (0.0831899, 0.15 )
central differences: (0.0831899, 0.15 )
```

Note: It's easy to prove that when using a bilinear interpolant the methods in (c) and (d) always give the same results.

2. The "Marching Cubes" Algorithm (9 Marks)

(a) [2 Marks] A Marching Cubes algorithm is being used to output the 0-isosurface of a scalar field. The figure below shows on the left one cube configuration. The numbers written beside the vertices are the field values at those vertices. The figure below on the right shows the coordinates of the cube on the left.

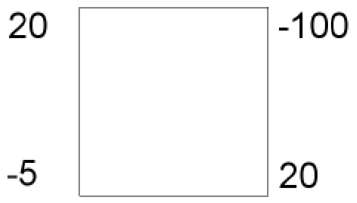


What are the vertices of the polygon(s) generated for the cube configuration on the left? Express the polygon vertices in terms of the cube vertices $\mathbf{p}_1,\ldots,\mathbf{p}_8$.

Solution: Only the vertex $\mathbf{p}_5$ is above the isosurface. We therefore have only one polygon with the vertices:

$$\left\{\frac{3}{4}\mathbf{p}_5+\frac{1}{4}\mathbf{p}_1,\ \frac{2}{3}\mathbf{p}_5+\frac{1}{3}\mathbf{p}_6,\ \frac{2}{3}\mathbf{p}_5+\frac{1}{3}\mathbf{p}_7\right\}$$

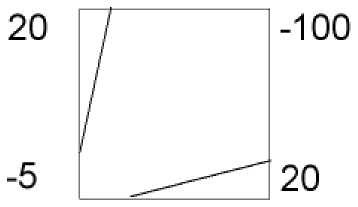
(b) [4 Marks] The marching cubes algorithm is used to output the 0-isosurface in a scalar field. The figure below shows a face and the field values at the vertices of the face.



As explained in the lectures there are two possible topologies for the 0-isocontour of within this face. **Compute** which topology is that of the bilinear interpolant.

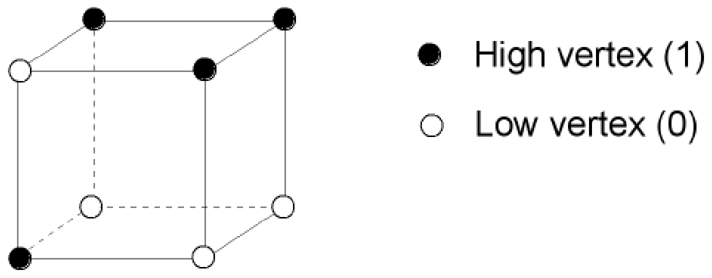
Solution: We have to compute the four intersection points and then sort them along one coordinate direction. Say the vertices are (0,0), (0,1), (1,0) and (1,1) then the four intersection points are: (0, 0.2), (0.2, 0), (0.1667, 1), (1, 0.1667).

Hence the topology is the one shown in the image on the right:

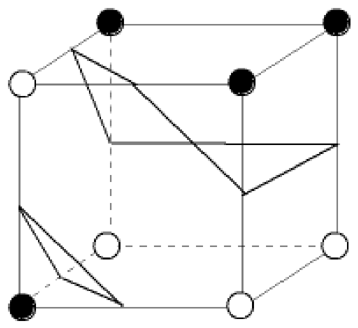


(c) [3 Marks] The lecture notes introduced an algorithm to build the look-up table for the Marching Cubes algorithm.

(i) Which polygon(s) would the algorithm produce for the configuration below if you use the cube table from slide 17 in handout 5? Please draw your answer (using a drawing program or by scanning a hand drawn image) into the document `Ass4Answers.doc`.



Solution: The algorithm in lecture will start with an arbitrary edge intersection, and will find the next edge intersection in clockwise direction for the face on the right of the current edge (with respect to 0->1 direction). This means that the resulting polygons will always separate "1" (high) points for ambiguous faces. The solution for the given configuration is shown on the right.



(ii) How many different ways are there to form topological polygons from the edge intersections in the above configuration?

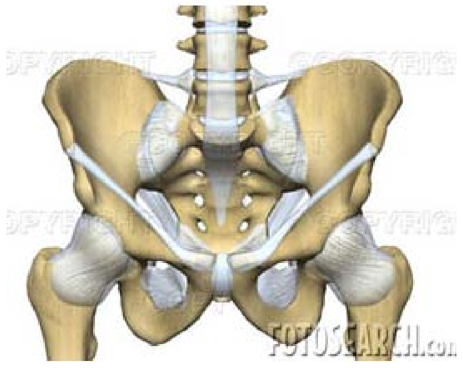
Solution: The above configuration has 2 ambiguous faces, so there are 2^2=4 different ways to form topological polygons.

3. *Volume Rendering* *(7 Marks)*

Download the zip-file `Ass4Q3.zip` from the assignment webpage. The enclosed program `Pelvis.cpp` reads a Computed Tomography data set of a pelvis. All values within the volume are between 0 and 255 and are of type unsigned char.

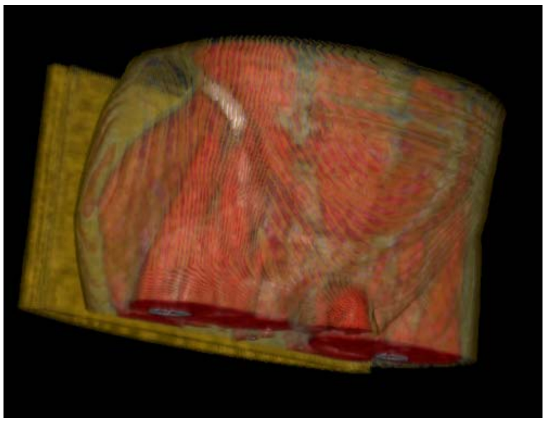
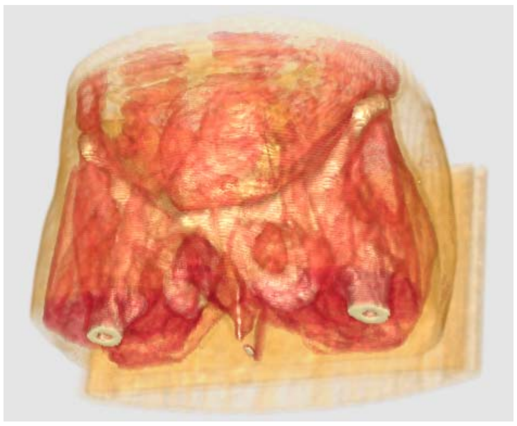
(a) [1Marks] Find a nice picture of a pelvis bone (use a drawing or an image) and add it together with a reference to your answer sheet.

Solution: URL: http://www.fotosearch.com/LIF122/3d205007/



(b) [6 Marks] Complete the file `Pelvis.cpp` so that it renders the pelvis data set using direct volume rendering. Define opacity and colour transfer functions so that the skin, muscle, and bone layer are all visible using anatomically realistic colours.

Solution: Below are two nice images generated by students from this course :-)

4. *Vector Field Visualization* *(8 Marks)*

Given is a vector field

$$\mathbf{v}(x) = \begin{pmatrix} 2y \\ -2x \\ 0.1 \end{pmatrix}$$

A streamline $\mathbf{s}(t)$ is defined by

$$\frac{d\mathbf{s}}{dt} = \mathbf{v}(\mathbf{s}(t)), \qquad \mathbf{s}(0) = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$$

(a) [2 marks] Proof that

$$\mathbf{s}(\mathrm{t}) = \begin{pmatrix} \sin 2t \\ \cos 2t \\ 0.1t \end{pmatrix}$$

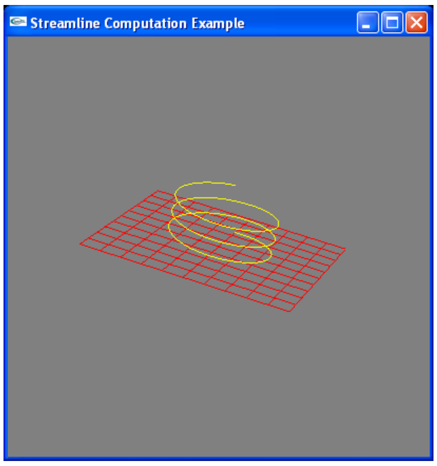
is the closed form of the streamline defined above.

Solution:

$$\mathbf{s}(0) = \begin{pmatrix} \sin 0 \\ \cos 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \qquad \text{OK!}$$

$$\frac{d\mathbf{s}}{dt} = \begin{pmatrix} 2\cos 2t \\ -2\sin 2t \\ 0.1 \end{pmatrix} = \begin{pmatrix} 2s_y(t) \\ -2s_x(t) \\ 0.1 \end{pmatrix} = \mathbf{v}(\mathbf{s}(t)) \qquad \text{OK!}$$

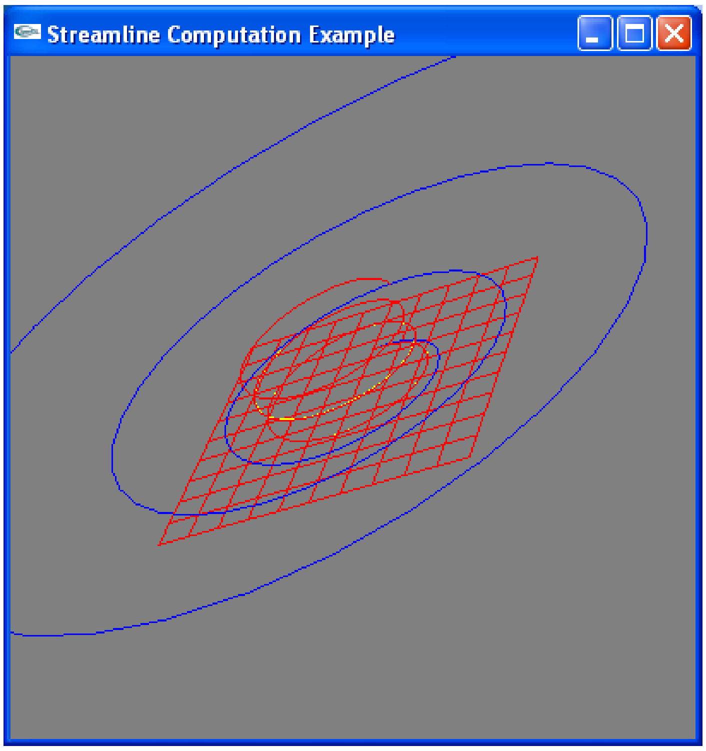
(b) [6 marks] Download the zip-file `Ass4Q4.zip` from the assignment webpage. The enclosed OpenGL program `Streamlines.cpp` draws the closed form of the streamline above in yellow. Complete the program so that it draws additionally two approximations of the above streamline computed by numerically solving the ordinary differential equations describing the streamline. The first approximation should be computed using 100 Euler steps of size $\Delta t = 0.1$ and be drawn in blue. The second approximation should be computed using 100 steps of size $\Delta t = 0.1$ with the Mid-point method and be drawn in red.



Solution:

```
// compute vertices with the Euler method
eulerVertices[0].setVector(0, 1, 0);
for(int i=1;i<=numVertices;i++)
{
  eulerVertices[i]=eulerVertices[i-1]+deltaT*vectorField(eulerVertices[i-1]);
}

// compute vertices with the Mid-point method
midpointVertices[0].setVector(0, 1, 0);
for(int i=1;i<=numVertices;i++)
{
   CVec3df midpoint=midpointVertices[i1]
                    +0.5*deltaT*vectorField(midpointVertices[i-1]);
   midpointVertices[i]=midpointVertices[i-1]+deltaT*vectorField(midpoint);
}
```



5. *Tensor Field Visualization* *(7 Marks)*

(a) [4 Marks] What are the eigenvalues and eigenvectors of the 3D symmetric second-order tensor

$$\mathbf{T} = \begin{pmatrix} 2 & 0 & 3 \\ 0 & 1 & 0 \\ 3 & 0 & 2 \end{pmatrix}$$

Solution:

$$|\mathbf{T} - \lambda\mathbf{I}| = \begin{vmatrix} 2-\lambda & 0 & 3 \\ 0 & 1-\lambda & 0 \\ 3 & 0 & 2-\lambda \end{vmatrix} = \left((2-\lambda)^2 - 9\right)(1-\lambda)$$

$$= (1+\lambda)(-5+\lambda)(1-\lambda)$$

$$\Rightarrow \lambda_1 = -1, \quad \lambda_2 = 5, \quad \lambda_3 = 1$$

$$(\mathbf{T} - \lambda_1\mathbf{I})\mathbf{v}_1 = \begin{pmatrix} 3 & 0 & 3 \\ 0 & 2 & 0 \\ 3 & 0 & 3 \end{pmatrix}\mathbf{v}_1 \doteq 0 \Rightarrow \mathbf{v}_1 = \begin{pmatrix} 1/\sqrt{2} \\ 0 \\ -1/\sqrt{2} \end{pmatrix}$$

$$(\mathbf{T} - \lambda_2\mathbf{I})\mathbf{v}_2 = \begin{pmatrix} -3 & 0 & 3 \\ 0 & -4 & 0 \\ 3 & 0 & -3 \end{pmatrix}\mathbf{v}_2 \doteq 0 \Rightarrow \mathbf{v}_2 = \begin{pmatrix} 1/\sqrt{2} \\ 0 \\ 1/\sqrt{2} \end{pmatrix}$$

$$(\mathbf{T} - \lambda_3\mathbf{I})\mathbf{v}_3 = \begin{pmatrix} 1 & 0 & 3 \\ 0 & 0 & 0 \\ 3 & 0 & 1 \end{pmatrix}\mathbf{v}_3 \doteq 0 \Rightarrow \mathbf{v}_3 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$$

(b) [3 Marks] Assumed you are given a unit sphere and the eigenvalues $\lambda_1, \lambda_2,$ and λ_3 and the corresponding eigenvectors $\mathbf{v}_1, \mathbf{v}_2,$ and $\mathbf{v}_3$ of a 3D symmetric second-order tensor. Give the transformation matrix which scales and rotates the unit sphere into a tensor ellipsoid representing the given tensor.

Solution: The unit sphere can be converted into a tensor ellipsoid by first scaling it with the eigenvalues and then aligning the principal axis of the ellipsoid with the eigenvectors by using a coordinate transformation matrix as rotation matrix:

$$\mathbf{T} = \mathbf{R} \cdot \mathbf{S} = \begin{pmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3 \end{pmatrix} \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{pmatrix} = \begin{pmatrix} \lambda_1\mathbf{v}_1 & \lambda_2\mathbf{v}_2 & \lambda_3\mathbf{v}_3 \end{pmatrix}$$