Visualization of high order numerical results with ViZiR

Visualisation de simulations numériques d'ordre élevé avec ViZiR

Matthieu Maunoury, Rémi Feuillet, Adrien Loseille

English Abstract—We present ViZiR: a light, simple and interactive high-order meshes and solutions visualization software using OpenGL 4 graphic pipeline. The use of OpenGL Shading Language (GLSL) allows to perform pixel exact rendering of high order solutions on flat elements and almost pixel exact rendering on curved elements. Post-processing tools, such as picking, isolines, clipping, capping are provided to interact on the fly with the results displayed.

1 Introduction

High order (HO) methods (for instance high-order finite elements or Discontinuous Galerkin methods) become very popular as they allow to perform complex computations efficiently and get good convergence rates with almost no numerical dissipation. Then, many fields, such as Computational Fluid Dynamic, acoustics, electromagnetism or medical modeling, use this kind of methods. However, the post-processing of high order meshes and solutions is still a current and complex challenge. Indeed, most of the standard visualization softwares (e.g. ParaView [1], Visit [2], Tecplot [11], Gmsh [3]) are based on linear primitives as imposed by the baseline graphic pipeline commonly-used. As a consequence, these tools are not able to visually inspect and validate meshes and solutions employed in high order methods.

To bypass these limitations, two main strategies exist: low-order remeshing and pixel-exact rendering. In the low-order remeshing strategy, the idea is to transform the data given by the high-order scheme in a combination of affine functions so that a standard visualization software is used to post-process it. The principle is therefore to define a mesh and affine representations which approximate the solution. A visualization error, corresponding to the gap between the numerical solution and its representation, is therefore introduced and controlled. The main difference between the works is how the mesh is created or reffined and the way the visualization error is defined and controlled [3], [5], [6], [10], [12]. The second strategy is dedicated to high-order solutions. Some approaches are based on raycasting [7]–[9]. The idea is that for each pixel, rays are cast to determine the color for this pixel. However, two non-linear problems (root finding problem and inversion of the geometrical mapping) have to be solved and are very costly. As a consequence, this solution has limited interactive capabilities [7]. Our method [4] intends to be a compromise between these two strategies. The goal is to keep interactivity and to guarantee a pixel-exact rendering on linear elements or almost pixel-exact rendering on curved elements without the need of extra subdivisions or raycasting.

We present ViZiR[1], an interactive and reliable high order meshes and solutions visualization platform, based on OpenGL Shading Language (GLSL).

- *Matthieu Maunoury: GAMMA3 team, INRIA Saclay, France Matthieu.Maunoury@inria.fr.*
- *Rémi Feuillet: GAMMA3 team, INRIA Saclay, France E-mail: Remi.Feuillet@inria.fr.*
- *Adrien Loseille: GAMMA3 team, INRIA Saclay, France E-mail: Adrien.Loseille@inria.fr.*

2 Main characteristics of ViZiR

ViZiR is very easy to use. As input, a mesh file including information on the elements and the connectivity as well as a solution file containing the degrees of freedom computed by a numerical code are given. These files follow the GMF format provided by the libMeshb[2] library.

Thanks to the pixel rendering, there is no visualization error (gap between the numerical solution and its representation) to be controlled and thus no parameter introduced. The only case a parameter is needed is related to the approximation of curved elements. However, in this case, it is possible to refine or unrefine this approximation on the fly and interactively in the software.

Figure 2 compares pixel exact rendering obtained with ViZiR to affine representations for a polynomial function of degree 3 on a simple element. Even with an adaptive subdivision, which allows a better approximation than with an uniform one, there is still a visualization error which is particularly visible when isolines are displayed (see figure 2 right and middle).

1. http://vizir.inria.fr
2. https://github.com/LoicMarechal/libMeshb

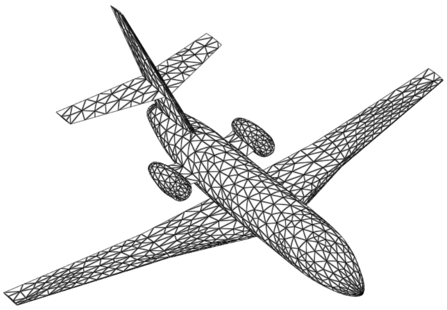


Fig. 1. Example of a curved mesh of 24560 P^2-triangles.

2.1 Pixel exact rendering on flat elements

OpenGL 4 graphic pipeline flexibility allows to compute on the fly the solution. It leads to a pixel exact rendering when flat elements (of degree one) are considered regardless of the degree of the solution. This recent language (GLSL) enables ViZiR to certify a faithful and interactive depiction. High order solutions are natively handled by ViZiR on surface and volume (tetrahedra, pyramids, prisms, meshes, hexahedra) meshes which can naturally be hybrid.

An example of pixel exact rendering on flat elements is presented on figure 3. The function is a mode defined as

$$f(x, y, z) = \sin(100\,\pi\,x)\sin(100\,\pi\,y)\sin(100\,\pi\,z), \quad (1)$$

where the length of the cube is 2. As the wavelength is 0.02, the solution is very oscillating. Nevertheless, all this richness of the function is perfectly reproduced.

2.2 Almost pixel exact rendering on curved elements

When more complex geometries are considered, curved elements perform a better approximation of the geometry. In this case, tesselation shaders occur in OpenGL pipeline and ensure a nearly pixel exact rendering (see [4] for more details on the shaders pipeline). Curved elements are subdivided and an exact rendering is done on these sub-elements. This subdivision is done internally, that is in the graphic pipeline. So, there is no increase of the ram memory used. An example of curved mesh is shown on figure 1.

2.3 Post-processing tools and interactivity

In addition to the solution representation, all the post-processing tools have to be included in the visualization software in order to make the analyses of the results possible.

In ViZiR, it is possible to pick elements to have information on the selected cell or on the mesh. This picking tool can be used to hide or display elements whose reference is the same. It is particularly useful for instance when volume meshes are treated. Furthermore, isolines can be displayed in ViZiR as shown on figure 2. Finally, clipping or capping can be done. This last point allows to investigate volume domains and therefore is crucial (see figure 4). Also, when dealing with HO solutions, the range of their values is estimated in a preprocessing step. Unlike standard methods, no sampling is needed to compute the solution. A proper approximation of extrema still remains mandatory to define the palette of the colormap.

3 Conclusion

The development of ViZiR using OpenGL 4 is still in progress but many tools are already available (pixel rendering, isolines, picking, clipping, capping for flat elements). You can visit http://vizir.inria.fr to download the demo and try ViZiR.

References

[1] U. Ayachit. *The paraview guide: a parallel visualization application*. Kitware, Inc., 2015.

[2] H. Childs. Visit: An end-user tool for visualizing and analyzing very large data. 2012.

[3] C. Geuzaine and J.-F. Remacle. Gmsh: A 3-d finite element mesh generator with built-in pre-and post-processing facilities. *International journal for numerical methods in engineering*, 79(11):1309–1331, 2009.

[4] A. Loseille and R. Feuillet. Vizir: High-order mesh and solution visualization using opengl 4.0 graphic pipeline. *56th AIAA Aerospace Sciences Meeting, AIAA Scitech*, 2018.

[5] M. Maunoury. *Méthode de visualisation adaptée aux simulations d'ordre élevé. Application à la compression-reconstruction de champs rayonnés pour des ondes harmoniques.* PhD thesis, 2019.

[6] M. Maunoury, C. Besse, V. Mouysset, S. Pernet, and P.-A. Haas. Well-suited and adaptive post-processing for the visualization of hp simulation results. *Journal of Computational Physics*, 375:1179–1204, 2018.

[7] B. Nelson, E. Liu, R. M. Kirby, and R. Haimes. Elvis: A system for the accurate and interactive visualization of high-order finite element solutions. *IEEE transactions on visualization and computer graphics*, 18(12):2325–2334, 2012.

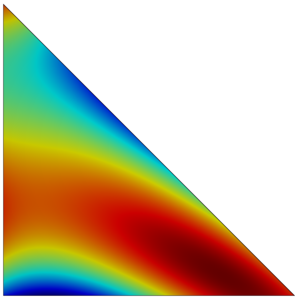
[8] B. W. Nelson. *Accurate and interactive visualization of high-order finite element fields.* PhD thesis, 2012.

[9] J. Peiro, D. Moxey, B. Jordi, S. Sherwin, B. Nelson, R. Kirby, and R. Haimes. High-order visualization with elvis. In *IDIHOM: Industrialization of High-Order Methods-A Top-Down Approach*, pages 521–534. Springer, 2015.

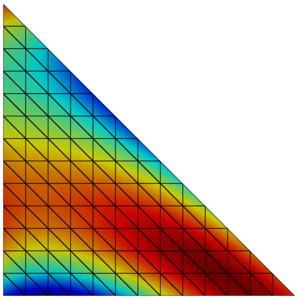
[10] W. J. Schroeder, F. Bertel, M. Malaterre, D. Thompson, P. P. Pebay, R. O'Bara, and S. Tendulkar. Methods and framework for visualizing higher-order finite elements. *IEEE Transactions on Visualization and Computer Graphics*, 12(4):446–460, 2006.

[11] TecPlot Inc. TecPlot. https://www.tecplot.com/.

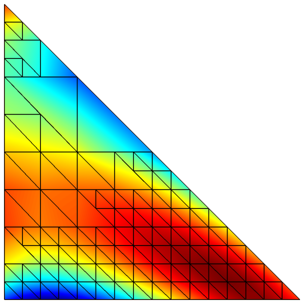
[12] L. Xu, X. Ren, X. Xu, H. Li, Y. Tang, and Y. Feng. An adaptive visualization tool for high order discontinuous galerkin method with quadratic elements. In *2017 IEEE International Conference on Computer and Information Technology (CIT)*, pages 176–183. IEEE, 2017.



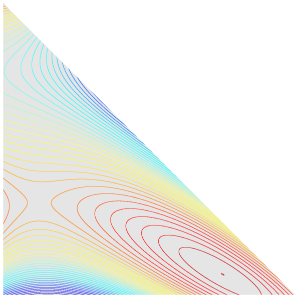
(a) Rendering with ViZiR



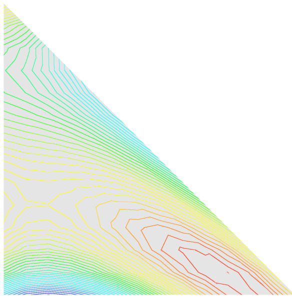
(b) Rendering with uniform subdivision



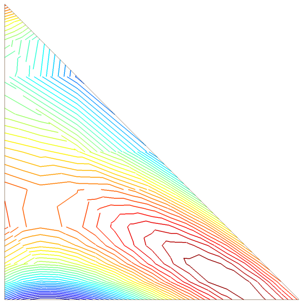
(c) Rendering with adaptive subdivision



(d) Isolines with ViZiR

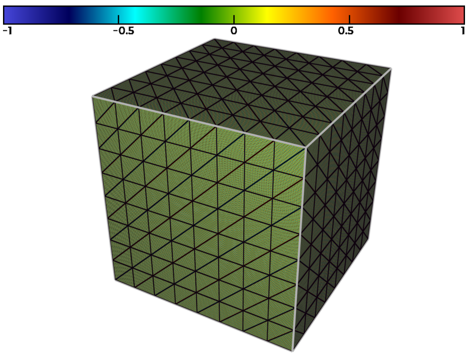


(e) Isolines with uniform subdivision

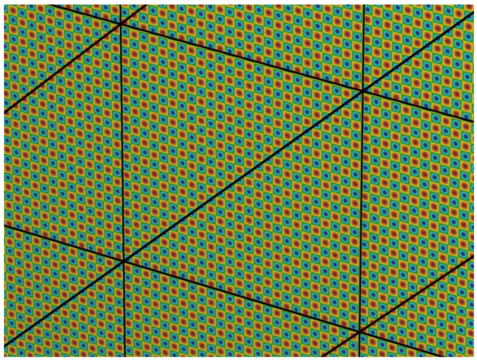


(f) Isolines with adaptive subdivision

Fig. 2. Rendering (top) and isolines (bottom) of a P^3-solution. Left: pixel exact rendering with ViZiR. Middle: uniform subdivision of 169 triangles. Right: adaptive subdivision of 169 triangles.

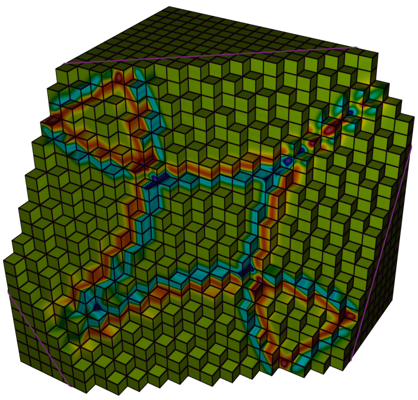


(a) Far view representing a continuous function

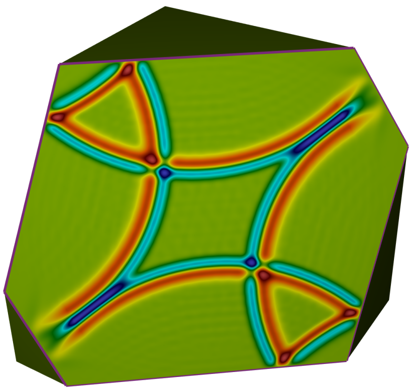


(b) Zoom on the rendering

Fig. 3. Pixel exact rendering of a mode in a mesh of flat elements (3832 tetrahedra).



(a) Clipping (the mesh is displayed)



(b) Capping (intersecting the cut plane)

Fig. 4. Examples of clipping (a) and capping (b) for a Q^6 solution on a mesh of 8000 hexahedra.