IMAGE SCIENCE USES OF GPU TEXTURE MEMORY

Introduction

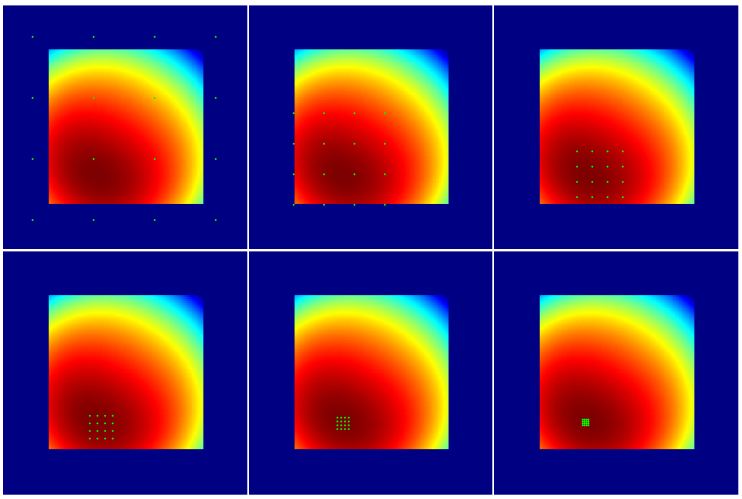
Graphics processing units (GPUs) have recently emerged as a flexible and powerful hardware solution for many scientific applications including computational chemistry, seismic processing, fluid dynamics, computational finance, and medical imaging, just to cite a few. The large number of cores that GPU devices feature provide a simple way to speed up many number-crunching applications. Through the CUDA™ parallel programming model, other features of GPUs—such as texture and surface memory—are available to the programmer. This document discusses how texture memory can be used to speed up a relevant estimation problems in medical imaging.

Maximum Likelihood Estimation of Position of Interaction

In maximum likelihood (ML) estimation of position of interaction we are interested in using noisy photomultiplier (PMT) outputs $\boldsymbol{g} = \{g_1, \dots, g_K\}$ to estimate the 2D position of interaction $\widehat{\boldsymbol{R}}$ of a gamma-ray photon with the camera's crystal. The estimation algorithm uses mean detector response data (which can be obtained through simulation or by performing an experiment) and calculates $\widehat{\boldsymbol{R}}$ by maximizing the probability $\Pr(\boldsymbol{g}|\boldsymbol{R})$. The quantity $\Pr(\boldsymbol{g}|\boldsymbol{R})$ models the detector outputs and it is characterized by the mean detector response $\bar{\boldsymbol{g}}(\boldsymbol{R}) = \{\bar{g}_1(\boldsymbol{R}), \dots, \bar{g}_K(\boldsymbol{R})\}$. If we assume Poisson statistics,

$$\Pr(\boldsymbol{g}|\boldsymbol{R}) = \prod_{k=1}^{K} \frac{[\bar{g}_k(\boldsymbol{R})]^{g_k}}{g_k!} e^{-\bar{g}_k(\boldsymbol{R})}.$$

Our implementation of the estimation algorithm uses the same algorithm of [1], in which $\Pr(\boldsymbol{g}|\boldsymbol{R})$ is evaluated for points $\boldsymbol{R}$ on a regular grid. The point of the grid that attains the largest value of $\Pr(\boldsymbol{g}|\boldsymbol{R})$ is retained and used as the center of another grid, finer than the previous one. As shown in the figure to the right, this process is repeated until a fixed number of iterations are performed. Texture fetching was used to speed up our implementation: we represented the mean detector response $\bar{g}_k(\boldsymbol{R})$ as a 2D layered texture in which the layer index is associated to the k^{th} PMT and $\boldsymbol{R}$ is the 2D texture coordinate. The GPU cubic B-spline interpolation library [2] was also used. Texture zero-padding—automatically performed by the hardware—provided an extra bonus: we no longer had to explicitly deal with boundary conditions at the detector edges and that resulted in a more elegant and much faster code.



Conclusions

We have shown how texture memory can be used to simplify and speed up computation in image science by discussing one application. Many other applications could benefit from texture memory. As an example, the sensitivity $s(\boldsymbol{r})$ of an imaging system can be calculated or measured over a discrete set of points and then stored as a texture to allow fast evaluation of $s(\boldsymbol{r})$ for any point $\boldsymbol{r}$ in the field of view.

References

[1] J. Y. Hesterman, L. Caucci, M. A. Kupinski, H. H. Barrett, and L. R. Furenlid, "Maximum-likelihood estimation with a contracting-grid search algorithm," *IEEE Trans. Nucl. Sci.*, vol. 57, no. 3, pp. 1077–1084, Jun. 2010.

[2] D. Ruijters and P. Thévenaz, "GPU prefilter for accurate cubic B-spline interpolation," *Comput. J.*, vol. 55, no. 1, pp. 15–20, Jan. 2012.