SUNVIZ: A Real-Time Visualization Environment for Space Physics Applications

S. Eliuk, P. Boulanger, and K. Kabin

University of Alberta Computing Science
Advanced Man Machine Interface Laboratory
University of Alberta Physics

Abstract. Real-time physically accurate simulations are difficult to create because of limited computational power available on a CPU. General purpose computing on the graphics processing unit (GPU) can provide a significant increase in performance. We are able to investigate the flow characteristics of a cloud of charged particles, which is one of the first steps in our goal of generating a real-time Coronal Mass Ejection (CME) simulator. Preliminary results show a sustained 60 Hz visual simulation with approximately four million particles and a non-visual simulation of 16 million particles at 30 Hz. The simulator provides a novel way to investigate a CME in real-time, and it has the potential to predict when a particular CME is geoeffective, i.e. an event that could damage electrical infrastructure such as satellites, space stations, power grids, etc...

1 Introduction

Real-time simulations have been a major focus in Computer science, physics, and many other fields, ever since the introduction of computers in the sixties. The attraction stems from the difficulty involved in modeling and predicting events in the real world. Numerical models can represent real-world phenomenon more or less accurately depending on how well the phenomenon is known, how well one can describe it mathematically, and how much computational power is available. In general, the more physically accurate the simulation is, the more complex the model tends to be.

One of the original goals of computational science was to be able to achieve real-time performance that would allow one to perform virtual experiments as if one were dealing with the real-world. For instance, a simple simulation of an n-body problem with approximately 16K bodies requires on the order of 300 gigaflops in order to achieve an update rate of 30 Hz [1], which is at the limit of real-time interaction. Until recently, this was totally out of the reach given the available computational power. In many real-world simulations, one needs processing power in the teraflops range to achieve real-time performance. Even with Moore's law and the increase in processing power of the CPU in the recent years, current CPUs do not provide the computational power necessary to obtain performances that will allow for real-time physical modeling. Current commercial CPUs provide on average 10 gigaflops of peak performance, at the cost of high-power consumption. One would require several hundred of these CPUs to achieve performance in the teraflops range. High Performance Computers (HPC) such as the Cray XT3, the IBM blue gene machine, or the SGI ALTIX machine can achieve this

G. Bebis et al. (Eds.): ISVC 2008, Part II, LNCS 5359, pp. 1–11, 2008.

performance as they have solved many of the engineering problems associated in connecting such a large number of CPUs. Problems such high-speed interconnect, power management, and failure management make the cost associated with these machines to be in the order of millions of dollars.

The ratio between flops/dollar has recently changed with the introduction of Graphic Processing Units (GPU), which are capable of processing information in the order of hundreds of gigaflops using a single instruction, multiple data (SIMD) architecture. These GPUs were originally designed to accelerate computer graphics rendering pipeline, but have recently been used to perform more general computations such as physical modeling. Clusters such as the NVIDIA PLEX or TESLA systems can harness the processing power of 4-6 GPUs, capable of achieving a computational power in the order of 2-3 teraflops. This is truly a revolution for computational sciences as NVIDIA systems have a price tag in the low tens of thousands instead of millions. For a slight increase in cost, one can build a cluster of 32 GPUs with a theoretical speed of 16.2 teraflops and an actual performance of 7.1 teraflops on a n-body simulation with one million bodies [2]. This is the computational power one needs to start to perform physically accurate simulations in real-time, opening the door to the dream of virtual experimentation.

To illustrate the power of the GPU, this paper presents an implementation of a physically accurate and real-time solar particle emission simulation to investigate particle interaction with the magnetosphere of the Sun and the Earth, mainly Coronal Mass Ejection (CME) seen in Figure 1. Researchers are interested to discover the location and energy required for a CME to have maximal geoeffective interaction with the Earths magnetosphere. This could eventually lead to Space weather predictions. At the heart of such system is the ability to analyze, in real-time, propagation of various disturbances from the Sun to the Earth, and their effects on the terrestrial magnetosphere.

Many of these CME's are not geoeffective, i.e. they have no effect on the Earth. When a CME is geoeffective it can result in global changes to the Earth's magnetic field. Take for instance the medium size CME of March 13, 1989 that caused severe transformer failure in Hydro-Quebec's power grid, resulting in power loss to a significant portion of the North American East Coast. A large CME event towards the earth could destroy satellites, power grids, and electrical infrastructure with costs in the trillions of dollars.

The importance of understanding these events is crucial for an advanced technological society like ours. However, with current computer technology, accurate simulations are not real-time, taking several days and sometimes weeks to complete [3]. Depending on the velocity of CME particles, some large event could reach Earth in just a few days. Therefore, simulations that takes days to weeks to be complete do not provide early enough warnings to be useful to protect the electronic infrastructures.

In Section 2, we describe the difficulties involved with CME simulations. Section 3 gives a review the state-of-the-art in the field. Section 4 defines a new technique for physically accurate simulation of CME's on the GPU. Section 5 describes experimental results, and Section 6 discusses some issues associated with the use of GPUs for general purpose computing. Finally, we review some future work.

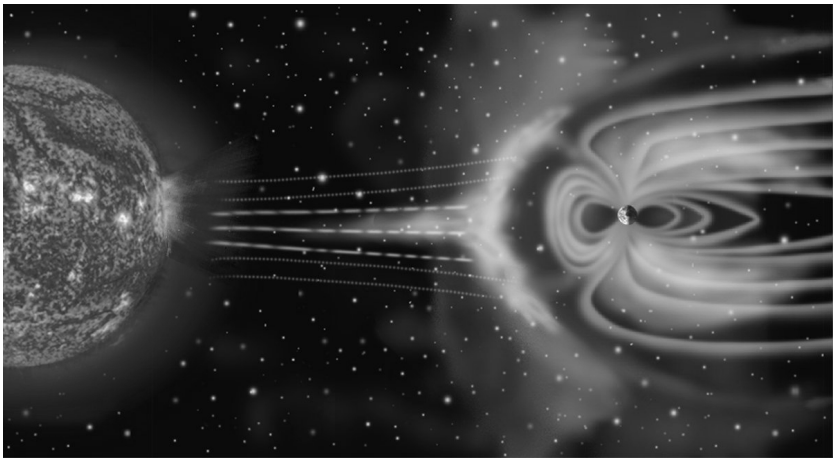

Fig. 1. NASA simulated CME magnetosphere interaction (picture from NASA website)

2 Problem Statement

Corona mass ejection simulations are extremely difficult to model in real-time. The computational complexity involved with a large particle count, greater than a few thousand, is very high. The problem can be broken down into four main computational requirements:

- Computation of the accurate localization of the Earth relative to the Sun using a n-body simulator;
- Computation of the Sun and the Earth magnetic fields configuration for each location of the Earth;
- Computation of the particle trajectories produced by the CME as it interacts with the Sun's magnetosphere;
- Computation of the particles trajectories received on Earth as it interacts with its magnetosphere;

The n-body component gives accurate positioning of inter-planetary bodies (Earth, Moon, etc) at any given time-step as they rotate around the Sun.

The exact location of inter-planetary bodies is essential as the global magnetic field (from Earth and the Sun) change as the Earth moves around the Sun. The planetary bodies included could be asteroids, space stations, satellites, moons, or many others.

The second component involves the understanding of plasma physics and particle interaction with magnetic fields. Fortunately the magnetic field of the Sun is relatively stable with periodic polar shifts occurring only every 11 years. This property allows the use of a simple analytical model that will be described shortly. More complex models can be used, but for Sun weather prediction these models are too complex to be useful and not needed because of the stability in polar shifts.

Last, the particle interaction with the Earth's magnetic field is needed. This is very similar to particle interaction with the Sun's magnetic field. The problem formulation is identical to the Sun, and it is not included.

3 Related Work

There are many computational techniques concerning n-body interaction, and CME magnetosphere interaction. We review a small subset of the most recent research in these areas.

3.1 Real-Time n-body Computation for Inter-planetary Body Localization

The n-body problem is an easy to understand example that illustrates the correlation between model complexity and computational power. The n-body problem is to compute the position of each inter-planetary body rotating around the Sun at a specific time step, given an initial position, mass, and velocities of all n-bodies. The computations are based on classical Newtonian mechanics. Compution of trajectories relies on pairwise comparisons to compute the trajectory of each body, leading to a $O(n^2)$ computational complexity; making the computation very expensive as n grows. Real-time simulation of physically accurate n-body interaction, for large value of n, is near impossible on generic CPU's.

There are simplifications to the physically accurate solution to n-body model that choose to reduce the number of pairwise comparisons by clustering bodies and forces as in the fast multi-pole method [4]. These simplifications result in a far simpler model, reducing the computational cost. The cost associated with using the simplified model is a less accurate description of the location of the planetary positions.

Nyland and Harris [1] GPU approach provides an excellent example of an efficient parallel n-body simulation on the GPU. They were able to obtain about 300 gigaflops on a single G80 GPU with a non-trivial size of n, 16K. This provided an extraordinary speedup over prior implementations on the CPU, around 50 times faster than a highly optimized serial implementation or 250 times faster than a portable C implementation [1].

The scalability of the NVIDIA G80 SLI technology is very efficient. Schive and Chien have built a graphics-card cluster for astrophysics (GRaCCA). The simulator combines 32 G80 GPUs into a single cluster with theoretical computational power of approximately 16.2 teraflops. They provide experimental results using a n-body problem, with one million bodies, where a peak performance of 7.1 teraflops is sustained [2]. The performance-to-cost ratio is extraordinary and shows that architecture to perform real-time physics is possible.

3.2 CME Simulation

Coronal mass ejection can be described in terms of particle gyration, electric, and magnetic drift. The interaction between the particle cloud generated by the CME and the Sun's magnetic and electrical fields is relatively well understood; but, the cause of CME is still being debated. There is a wealth of knowledge on CME simulations, but they all seem to have a common problem: they are rarely real-time. The Multidisciplinary Research Program (MURI) of the University of Michigan studies coronal mass ejections and the effects on space weather. The program has shown wonderful results over the

years and provided an in-depth understanding of space weather, however they do not provide real-time results above 1.8 Hz [5].

Some of the most accurate CME simulations such as [6] and [7] provide amazing precision and results but are not real-time. The reality is that there are simulators available, but none of them can compute these simulations in real-time.

4 Tools and Methods

The SUNVIZ simulator described in this paper provides an environment to investigate solar events. The simulator is very different from most solar simulators because it models all events by simulation rather than scripts. This means the results being viewed are not scripted events but are actually computed in real-time.

4.1 Accurate Stellar Data of the Solar System

SUNVIZ uses the most accurate stellar dataset, from the Hayden Institute primarily funded by NASA. The dataset provides heliocentric stellar coordinate maps for approximately 120K stars [8]. The Hayden dataset provides the means to render our galaxy, enhancing the visual appeal of the application and providing a great sense of realism during simulation. It also provides a means to accurately simulate particle bursts toward distant stars because the position of those stars is known.

4.2 The Solar System n-body Simulator

The goal of the simulator was to introduce n-body interaction in real-time. The computational problems associated with large n-body problems was solved by parallel execution on the GPU. The heliocentric position *xyz* and *mass* of a body was mapped, one-to-one, to the *rgba* texture parameters. Velocities *vx, vy, vz* were mapped to the *rgb* components of another texture. The alpha component, *a*, was not needed for any computation in the velocity texture so only the components *rgb* were used. This reduced the amount of memory required on the GPU for computation. Fewer memory transfers result in fewer GPU cycles wasted waiting for data. This n-body problem formulation on the GPU provided real-time n-body interaction, provides a novel way to introduce new bodies and analyze the trajectories. For instance, one could investigate CME interaction with moving planets, asteroids, space shuttle or space station.

The algorithm for mapping the n-body problem to the GPU was adopted from [1]; a complete review of the technique can be found in [1]. The algorithm consists of pairwise computation of forces between bodies using classical Newtonian mechanics, as shown in Equation (1) to (4):

$$\boldsymbol{F} = \boldsymbol{r}(G * M_1 * M_2)/r^3 \tag{1}$$

$$\boldsymbol{a} = \boldsymbol{F}/M \tag{2}$$

$$\boldsymbol{v}_{new} = \boldsymbol{v}_{old} + \boldsymbol{a} * \triangle t \tag{3}$$

$$\boldsymbol{P}_{new} = \boldsymbol{P}_{old} + \boldsymbol{v}_{new} * \triangle t, \tag{4}$$

where $\boldsymbol{F}$ = force generated between two bodies, G = gravitational constant, M_1 = mass of body 1, M_2 = mass of body 2, $\boldsymbol{r}$ = directional vector between body 1 and 2, $\boldsymbol{a}$ = acceleration, $\boldsymbol{v}$ = velocity, $\boldsymbol{P}$ = position, and $\triangle t$ = timestep.

The total force experience for a given timestep for a given body is simply the summation of all forces experienced. This provides a relatively simple parallel algorithm to carry out the computation of forces. For a complete review of the technique refer to [1].

4.3 Particle and Magnetosphere Interaction

Simulation of particle interactions with the magnetosphere is a little more complicated. A simplified definition of particle interaction with the Sun's magnetosphere is given below. The particle interaction with the Earth's magnetosphere is similar. One should note that the forces acting on a particle is the summation of the forces generated by the magnetic and electric fields produced by the Sun and Earth. The formal definition for the electric and magnetic field can be found in [9].

The most significant forces encountered are the electric and magnetic fields as given in Equations (5-7). These forces are simply summed as given in Equation (8).

$$\boldsymbol{v}_E = (E \times B)/B^2 \tag{5}$$

$$w_g = (q * \boldsymbol{B})/m, \tag{6}$$

$$\boldsymbol{v}_B = (v_{\parallel}^2 + 0.5v_{\perp}^2)((\boldsymbol{B} \times \nabla \boldsymbol{B})/(w_g B^2)) \tag{7}$$

$$\boldsymbol{R}_{new} = (\boldsymbol{v}_E + \boldsymbol{v}_B + \boldsymbol{v}_{\parallel})\triangle t \tag{8}$$

where $\boldsymbol{R}_{new}$ = new particle Position, $\boldsymbol{E}$ = electric Field, $\boldsymbol{B}$ = magnetic Field, $\boldsymbol{v}_{\parallel}$ = particle velocity component parallel to the magnetic Field, $\boldsymbol{v}_{\perp}$ = varticle velocity component perpendicular to the magnetic Field, $\nabla \boldsymbol{B}$ = gradient of $\boldsymbol{B}$, m = mass, and q = charge.

There are a couple options for mapping the magnetic and electric fields to the GPU; the processor-intensive on-the-fly particle interaction or the memory-intensive precomputed table look-up technique.

The processor-intensive per-particle computation on the GPU implies that, for every particle and position, the equations are solved and the positions and velocities are updated. The GPU is amazingly efficient with this type of multi-data single-instruction set.

The second method, which relies primarily on precomputation and table look-ups, recognizes that the magnetic field is stable and only shifts every 11 years. This is interesting because a 3D grid can be precomputed for the magnetic and electric forces. If a given particle lies in a cell of the grid, a simple table look-up is conducted. If the particle lies on the boundary of a many grid cells then an average of the boundary cell forces can be computed and applied to that particle. The boundary technique is much more expensive and is directly related to the density of the precomputed grid. The precomputed grid technique is very efficient in terms of computation, suffers however from a number of problems.

The density of the 3D grid is proportional to the amount of memory available on the GPU. The current upper bound for memory on the GPU is about 1.5Gb. This does not give a very dense grid considering the space the magnetic and electric fields occupy.

This results in more of the expensive boundary technique of many table look-ups to calculate the average forces exerted.

The cost associated with the computation of the table look-up on the GPU is a major slowdown. This is apparent because any memory look-up on any system requires the processor to wait for the data to be fetched and loaded into registers. This is even more of a problem because the relatively low density of the grid due to the memory constraints on the GPU; using more boundary cases requiring multiple look-up table access to calculate average forces exerted on a particle.

The per-particle computation on the GPU provides an infinitely dense 3D grid because its computed on the fly, depending on particle location. No averaging or approximation is used. This is the technique currently implemented. When computing in real-time, one also has the potential to computationally steer the simulation. This is another highlight to using the on-the-fly technique.

The computation involved with this simulation can be performed in parallel, making the GPU an ideal candidate for this simulation. The objective is not trivial and special problem formation is needed in order to use the high speed stream processors on the GPU. We are currently investigating GPU problem formulation for Equation (5) and Equation (7). Preliminary results show that an upper bound of approximately one million particles, which is nearly four times the number of particles used in the NASA CME simulation, with real-time results at about 35 Hz.

4.4 Real-Time Rendering

Rendering the four million particles in a particle engine is difficult in itself. There are a couple techniques currently being tested.

The first technique is based on keeping all data on the graphics card. This provides extremely fast positional look-ups for particle rendering because the data does not need to flow from the i/o bus to the GPU. The technique uses the vertex shader as a means to translate a particles to the correct rendering location using a texture look-up. Preliminary results show that this technique is quicker and provides real-time rendering of stellar weather.

The second technique relies upon storing the position of every particle at every timestep of the simulation. This technique is good for repeatability, the data is stored in main memory and can be written to disk if need be. However, it affects the frame rate of the simulation significantly because all the very fast processing on the GPU is slowed down waiting for positional information contained on the GPU to be transferred through the i/o bus to the main memory.

4.5 Implementation Details

In order to represent the large number of particles we abstract the representation using texture arrays on the GPU. In order to update the dynamic properties of the simulation we use reading and writing textures. The textures are mapped to openGL color attachment points, mainly one of the currently eight available color attachment points

```
GL_COLOR_ATTACHMENT0_EXT -> GL_COLOR_ATTACHMENT7_EXT
```

Finally, the visualization is updated by a vertex texture lookup that obtains the current particle position. The matrix product of the Model view projection (MVP) matrix and the updated position of the particle is computed, which provides the proper rendering of the particle.

Obviously many of the implementation details are not included but available on our website.

5 Results

The n-body simulator is currently implemented using vertex and fragment shaders and provides real-time results with approximately 8K bodies running at 45 Hz. The simulation is very accurate, while running with small time-steps over 400 years there is only a 4 day difference between the known analytical position of the Earth and the simulated n-body position. For a given simulation like that of a CME over the course of 1-10 days the amount of positional error is negligible.

Simulation of the charged particle cloud with approx. four million particles can be visualized and computationally steered at 60 Hz, seen in Figures(2, 3 , and 4). This is much faster than previous implementations, mainly because the GPU is used to solve the analytical formulas for each particle in parallel. In general, about 128 particles are being solved in parallel by the stream processors on the GPU. This technique is much quicker in terms of available computational power and cache coherency in terms of loaded instruction sets.

The results are much more accurate than previous real-time implementation that use a pre-computed grids to store magnetic and electric field values because grids rely on approximations for any given particle. For instance, if a particle were to lie on a the border of a cell, this would result in only an approximate value for the forces

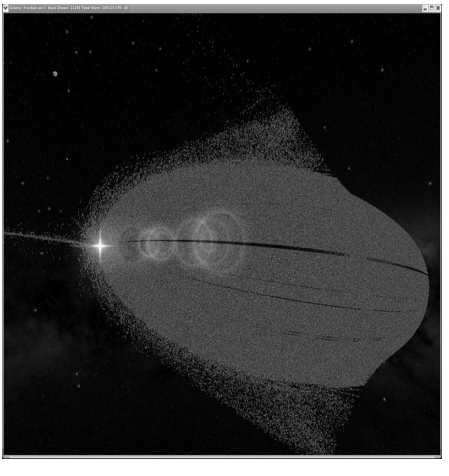

Fig. 2. Charged particle cloud with approx. four million particles.

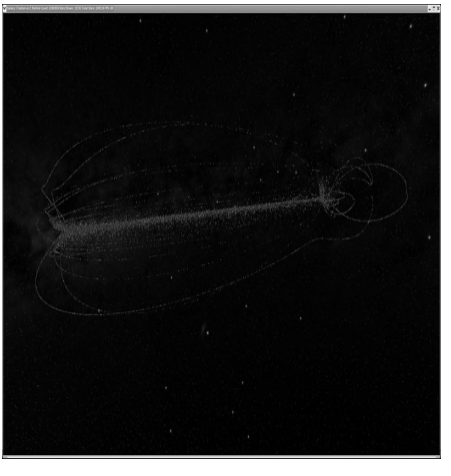

Fig. 3. Charged particle cloud with approx. four million particles, solar magnetic field lines can be seen.

Fig. 4. Charged Particle cloud with approx. four million particles, relative distances to Earth can be seen.

influencing the particle at a given time-step. Secondly, grid techniques are much slower when implemented on a GPU, this is rather surprising because look-up tables are generally faster on scalar computers. However on the GPU, memory lookups are the bottleneck, and these are what is required when using the grid based table lookup technique.

Computer Specifications for simulation, Intel Dual Core 3.0 Ghz processor, 4Gb of DDR2 memory, and Nvidia Quadro FX 5600 GPU 1.5 Gb of memory.

Computer Specifications for 'updated' results simulations, Intel Extreme Quad Core QX9770 3.2 Ghz processor (overclocked to 4.0 Ghz) with 12 Mb of cache, 4Gb of DDR3 2000Mhz, and 2 * Nvidia 9800 GX2 GPU (affectively quad-sli) 2.0 Gb of memory.

6 Limitations

The level or precision when using general purpose GPU computing is an issue. Floating point values were used for all computation involving the n-body and magnetosphere interaction. The current G92 GPU does have the potential to use double precision, but at the time of the implementation the drivers were not available. Future work will involve comparing CPU double precision simulations with GPU floating point simulations.

This paper ignores a major problem with CME numerical computation, namely the simulation of location and the initial cause of the celestial phenomenon. The simulator uses instead, a user-specified location of the CME or using backward integration from the body to the Sun to derive the origin location. The particle energy for this emission is set to be in the 1-100Ev range which we can derive the particle velocities from. Further explanations of the causes of coronal mass ejections and their approximation can be found in [10].

Finally, a charged particle cloud represents a good approximation and proof of concept, this technique shows a GPU based CME simulator is possible. Our latest 'updated' results switching to a quad sli 9800 GX2 setup has provided four times the

computational power and approx. 190 Hz simulations for 4 million particles. See appendix for system specifications.

7 Conclusion

Our plans to test the simulator not only on a single G80 GPU but a cluster of GPUs are well underway. The cluster consists of qual sli nvidia 9800 GX2 GPU's. This will provide an increased level of performance for on-the-fly computationally expensive particle magnetosphere interaction. The increase in memory size will also give the possibility to have a denser grid for memory intensive look-up table technique.

With the increased in computational power one could increase the number of particles for the corona mass ejection providing a greater level of detail to the simulation, such as the case when we increased to 16 million particles. There would even be the possibility to analyze the affects of the moons magnetic fields on the charged particle cloud if enough computational power is available.

SUNVIZ provides the means to adequately understand and predict a CME. Current CPU architecture does not provide the computational power required to carry out physically accurate simulations in real-time. This limitation has resulted in simulations that require many days, weeks or months to carry out. These simulations do not provide the information fast enough to notify those whose infrastructure may be affected. The GPU provides a very powerful framework for simulations. With careful problem formulation, one can achieve real-time physically accurate simulation. We present a new technique, using general purpose computing on the GPU, to provide real-time CME simulations which could lead to space weather prediction. Further research is needed to understand the cause of CME's and simulations provide a way to visualize this complex phenomenon.

References

1. Nyland, L., Harris, M., Prins, J.: GPUGEMS3. Addison-Wesley, Boston (2007); Fast n-body simulation with CUDA
2. Schive, H.Y., Chien, C.H., Wong, S.K., Tsai, Y.C., Chiueh, T.: Graphic-card cluster for astrophysics (gracca) – performance tests (2007)
3. Manchester, W.B., Gombosi, T.I., Roussev, I., Zeeuw, D.L.D., Sokolov, I.V., Powell, K.G., T'oth, G., Opher, M.: Three-dimensional mhd simulation of a ux rope driven cme. Journal of Geophysical Research (Space Physics) 109, 1102 (2004)
4. Greengard, L., Rokhlin, V.: A Fast Algorithm for Particle Simulations. Journal of Computational Physics 73, 325–348 (1987)
5. Toth, G.: Space weather modeling framework: A new tool for the space. Journal of Geophysical Research 110, 1029–1091 (2005)
6. Gombosi, T., DeZeeuw, D., Groth, C., Powell, K., Stout, Q.: Multiscale mhd simulation of coronal mass ejection and its interaction with the magnetosphere-ionsphere system. Journal of Atmospheric and Solar-Terrestial Physics 62, 1515–1525 (2000)
7. Wu, C., Fry, C., Dryer, M., Liou, K.: Three-dimensional global simulation of interplanetary coronal mass ejection propagation from the sun to the heliosphere: Solar event of 12 may 1997. Journal of Geophysical Research 123, 1061–1078 (2007)

8. Drimmel, R., Abbott, B., Emmart, C., AMNH/Hayden, J.A., NCSA/UIUC, S.L: (Amnh star catalog, digital universe atlas version 3)
9. Treumann, R., Baumjohann, W.: Basic space plasma physics/advanced space plasma physics. Plasma Physics and Controlled Fusion 43, 371 (2001)
10. Roussev, I.I., Lugaz, N., Sokolov, I.V.: New physical insight on the changes in magnetic field topology during coronal mass ejections: Case study for the 2002 apr 21 event. In: American Astronomical Society Meeting Abstracts. American Astronomical Society Meeting Abstracts, vol. 210, p. 58.04 (2007)