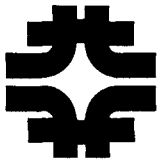


Fermi National Accelerator Laboratory

FERMILAB-Pub-95/077

# On the Use of Scientific Visualization for High Energy Physics Data Analysis

P. Lebrun and J. Kallenbach

*Fermi National Accelerator Laboratory*
*P.O. Box 500, Batavia, Illinois 60510*

April 1995

Submitted to *Computers in Physics*

Operated by Universities Research Association Inc. under Contract No. DE-AC02-76CHO3000 with the United States Department of Energy

## Disclaimer

# On the Use Of Scientific Visualization for High Energy Physics Data Analysis.

P. Lebrun, J. Kallenbach

*Fermilab, Batavia, IL 60510*

(May 18, 1995)

## Abstract

The use of Scientific Visualization for High Energy Physics analysis is described in the context of a commercial Scientific Visualization package (IRIS-EXPLORER). A few examples are given, covering briefly the conventional use volumetric data analysis, event display, relativistic kinematics, analysis of decay amplitudes on Dalitz plot. We comment on our experience with this method.

Typeset using REVTEX

While Computer Graphics has been used widely in High Energy Physics (HEP) for many decades, scientific visualization is now a well-established technique used in other disciplines, such as chemistry and bio-medicine.[1]. The purpose of this paper is to briefly explore some examples of real use of Scientific Visualization for HEP data analysis and describe how the technology is perceived in our community, from a technical as well as sociological standpoint. We start by briefly stating what we mean by Scientific Visualization. We follow that with a brief history recent advances which have made sci-vis technology a viable tool. We then describe the essence of HEP data analysis with respect to graphical and human-computer interfaces. The examples are derived from the experimental work of two Fermilab-based collaborations (E687 and D0). We conclude by commenting on the pros and cons of this technology for HEP analysis. Our experience has been almost exclusively with SGI-Explorer [1]. We assume that similar results can be obtained with *IBM Data Explorer* [2] or *AVS* [3].

## I. REQUIREMENTS FOR SCIENTIFIC VISUSALIZATION

One of the most compelling reasons for integrating sci-vis programs at this time, is the increased availability of the correct computing machinery capable of advanced graphics. Roughly speaking, a desktop visualization system has the following minimum requirements:

- Hardware - At least 30 MIPS, 32 MB memory, high resolution 8-bit color screen, co-processors for graphical functions. Until recently, such functionality was costly. Prices are now sufficiently low that most scientists can have ready access to such a system. The examples described below can run satisfactorily on almost any new machine, including sufficiently high-end off-the shelf personal computers.

---

[1]For instance compare the frequency and quality of molecule display in journal such as Science versus HEP Event Display in Physics Review.

- Operating Systems - A multi-tasking operating system is a crucial element of a sci-vis computer. The ability to have multiple processes dedicated to specific tasks (background data processing, transformation of abstract structures to a geometrical model, graphical rendering based on independent color maps...) allows great flexibily while making the programming environment robust against local bugs.

## II. A DEFINITION OF HEP VISUALIZATION

High Energy physicists have been using Computer Graphics for decades. In order to recognize new capabilities, it is appropriate to clearly state our working definition of "scientific visualization". Such systems are characterized by the following key components:

- Pre-programmed modules which perform primitive graphing functions. Typically these would include curves, contours, color maps, 3D rendering modules and image processing tools.
- Software modules to easily input volumetric data into the system.
- A *Graphical User Interface Builder* (GUIB). When viewing large amounts of data, the capability to hide portions of it, color it, etc. is crucial. A good GUIB lets the programmer easily add intuitive filters which allow the user to add and remove data from the display.
- The *Application Programmer Interface* (API). A library of F77, C, and/or C++ callable functions is necessary to build the modules that convert "physics" data to data structures that can be visualized. Along with the GUI, these libraries allow the advanced user to easily build and modify exploratory modules that can test ideas and models quickly.

Explorer, the sci-vis package provided with SGI systems, includes all of the components mentioned above. The 3-D display is done by means of the *Render* module, a GL-based

drawing area with controls for rotation, translation, Zoom, etc. API's are provided for FORTRAN, C, and C++. A useful GUI builder, providing the user with the capability to add his/her own control buttons, dials, etc. is included. The package provides the programmer with straightforward means to input the HEP data and convert it to a form which *Render* can display.

Let us turn our attention to HEP data analysis itself. Beyond the usual handling of volumetric data (for instance, the modeling of the magnetic field inside a large solenoid), HEP data handling can be very challenging. The total amount of raw data is large (multiple Terabytes). The bulk of this data consist usually of a sequence of *events*, each event describing the recording of the collision between two subatomic particles. The raw data consist of elementary detector responses, and must be reconstructed into a coherent list of traces, particles trajectories and relationships among these traces. These reconstruction algorithms tend to be CPU intensive [4]. In many experiments, thousands or even millions of these events are recorded and reconstructed, out of which only a fraction are considered.

*Event Displays* consisting of pictorial representation of the traces produced by the particles interacting with the detector elements can obviously help us in selecting and tuning correct reconstruction algorithms. Such graphical applications have been in used in HEP since the invention of computer grpahics. In addition, statistical analysis is used to infer average detector responses, establish calibration constants, monitor the data reconstruction and so forth. The combination of statistical analysis and event displays allows us to make event selection, for instance, to select a few Top-Quark candidate events out of millions of recorded collisions at the Tevatron collider.

Scientific Visualization is usually perceived as not dealing with simple histograms or 1D curves, which are typical graphical output from statistical packages. An optimized, highly interactive histogram browser such as HistoScope [5] uses the same underlying technologies (multi-processing under UNIX, RPC, MOTIF windowing system) as IRIS-EXPLORER does. This distinction is a bit arbitrary, and the last example described below explores a bridge between advanced statistical analysis and conventional visualization.

## III. EXAMPLES OF HEP VISUALIZATION USING SGI-EXPLORER

### A. Volumetric Data Analysis

Although not particularly typical of HEP, volumetric data is noteworthy, because such data can be very easily understood with scientific visualization and without writing much code. SGI-EXPLORER has been used in HEP to study three dimensional magnetic fields.

### B. Event Displays

Visualization and HEP Event Display are traditionally considered as separate techniques [8]. Until very recently, HEP event displays have been written using graphics toolkits, which the programmer uses by calling graphics primitives routines. Fixed, two-dimensional plots were generated. Such toolkits are used because the programming environment is familiar (although time consuming!). They were the best solution available when they were written, more than a decade ago.

Because SGI-EXPLORER has a built-in Geometry Data Type[2] , and an application interface along with a corresponding GUI builder, coding an Event Display with such high level tools becomes very easy. The amount of code the programmer actually types in is minimal. Once the Geometry has been assembled, it is sent to the *Render*, and all the corresponding functionality is automatically available. Using conventional graphics toolkits (e.g. GKS or PHIGS), the investment in such specific graphical programs is prohibitive; a large amount of low-level FORTRAN code must be typed in, debugged, and repeatedly

---

[2]SGI-EXPLORER application interface comprises numerous utilities to describe simple geometrical objects(lines, polygons...). Once created, these data structures can be transcribed to linear data streams that can be transfered to files or among processes. The data format is opaque to the user.

linked to generate even the basic still views. Even more code would be needed to provide direct, interactive picture manipulation.

We describe two examples that are highly optimized to solve specific problems; they are not global displays. The first example deals with a well known problem: the representation of simulated elementary particle tracks on top of a high precision tracking detector. The second attempts to represent more refined constructs, such as fitted tracks that can be directly mapped onto the physical 3-momentum vectors of particles, and vertices that are associated to decays.

### *1. D0 Trigger Upgrade*

In our first example, shown in figure 1, we consider a birds-eye view (the beam axis being perpendicular to the page) of the proposed D0 central fiber tracking system. [13]. Individual layers are grouped into four cylindrical superlayers, each fiber being represented as long polygons, wire-framed (the outside edge is described by segments; no surfaces are displayed). As the fibers run along the beam axis, they are shown as small circles (colored in blue) on figure 1. The user obviously has the choice of representing all the fibers ($\approx$ 60000 of them) or simply the ones that have been hit by a particle. The centroid position of these hits for each superlayer is represented as red markers. The simulated tracks, computed by the D0 Monte-Carlo simulation program, are shown as green lines. The curvature of the charged tracks is inversely proportional to the transverse momentum, as the detector resides in solenoidal magnetic field. Tracks of high transverse momentum can be identified by a fast pattern recognition algorithm that can be used in the trigger. These tracks are shown in red. The GUI builder of Explorer was found to be very useful to implement quickly various options, such as the capability of selecting events, turning on/off various components. The geometrical modeling of such a detector is not always trivial, because of the large aspect ratio involved (these fibers are longer than 300 cm and less than 0.1 cm in diameter), the spatial range (the tracking volume is measured in cubic meters, while elements must be

positioned with $\approx$ few hundred $\mu$m. accuracy) and the large number of elements involved (hundreds of tracks, thousands of hits). The flexibility and ease of use of the Geometry Application Interface were useful as we tried different drawing algorithms for the various components.

### *2. E687 Vertex Analysis*

In this experiment, we reconstruct particle trajectories emerging from the inclusive reaction $\gamma + Be \rightarrow D + \bar{D} + X$, where $D$ and $\bar{D}$ are charmed and anticharmed mesons and X an unspecificed set of particles [9]. The $D(\bar{D})$ particles subsequently decay within the aparatus, with a typical lifetime of a fraction of a picosecond, which corresponds to a range of about a centimeter in the laboratory system. Particle positions are recorded downstream of the production point (primary vertex) or decay point (secondary vertices), allowing us to reconstruct the direction and momentum (tracks). Primary or secondary vertices are therefore not directly measured and must be inferred from the 3D track parameter themselves. In order to untangle the decay chain correctly, the correct track-to-vertex relationships must be found. The basic criteria to decide if tracks come from a single vertex is based on how well these tracks cross each other at single 3D point, as quantitatively determined by the $\chi^2$ of the vertex fit. Our goal is to identify the correct pattern, keeping in mind error measurement as well detector acceptance and efficiency (not all particles are reconstructed). This is shown on figure 2. The square wire frames represent target markers used as a crude scale. The tracks are represented as filled polygonal arrows. Their length represents the norm of the 3-momentum vector. Some tracks are "fading" away, as their momentum could not be determined by the spectrometer. The transverse size of the arrow body is roughly matched to the size of the transverse position error of the track. The color of the arrow is mapped to the particle type or identity (Kaon vs pions...). The vertices are represented by ellipsoids, whose size are determined by the 3D position error returned by the vertex fitting routine. The color of the ellispoid is mapped to the $\chi^2$ vertex, through the use of a standard col-

ormap. Thus, Color is used in two distinct ways in this example:(i) discretely, to represent the particle type (an old trick used in conventional HEP event displays) and (ii) using a continuous mapping bewtween a scalar number and the color, a bread-and-butter technique used in visualization. Since the range of this scalar number is sometimes hard to guess, this mapping may change. The *GenerateColormap* module, provided with SGI Explorer, allows interactive adjustment of the color mapped to this $\chi^2$.

The 3D-rendering is essential in this application as our primary goal is to assess whether tracks cross each other. Animated rotation and zooming in the *Render* module are useful for this purpose.

A final remark on Event Display using visualization: because of the intrinsic modularity of the SGI-EXPLORER environment, it is straightforward to combine the display of magnetic field (volumetric data on a lattice) and the Event Display, giving us a global view of the basic physics principle on which spectroscopy is based.

### C. Kinematics studies, 3-body decays

Once the tracks and vertices are reconstructed and particles identified, the kinematics of the decay of the charmed particle can be studied based on the knowledge of the all 4-momentum vectors in the final state. For 3-body decays, a standard technique for this sort of kinematical analysis is the *Dalitz Plot*, where decay probabilities are displayed versus the square of two invariant masses of the decay products. In our case, $D^+ \rightarrow K^-\pi^+\pi^+$, the decay amplitude is expressed as a function of the square of the two invariant masses $M^2_{K\pi}$ and $M^2_{\pi\pi}$ [11].

The basic components of Explorer provide all of the functionality one needs to generate a simple Dalitz Plot. Using the GUIB, a control panel was constructed with slider widgets to set the two masses. A 2-D *Lattice* structure[3] assembled which described the border of

---

[3]In SGI-EXPLORER, as the *Geometrical* data structure are used to describe geometrical object,

the Dalitz plot (that is, the region in the $M^2_{K\pi}$, $M^2_{\pi\pi}$ space where the decay is kinematicaly allowed). Finally, a *Geometry* structure representing the 3-momentum vectors as simple dots or sphere in phase space was coded. The last two structures are then sent to the *Render* module where all of the standard viewing features are available. The output is shown on figure 3. Although this example expresses rather simple kinematical relationships, it is a good start to get acquainted with EXPLORER, because none of the underlying lattices or geometries are complex while the basic tools are used.

### D. Decay Amplitude Analysis on the Dalitz plot

The total decay probability *Gamma* is assumed to be derived from of a coherent sum of various quantum mechanical amplitudes, each of these amplitudes corresponding to a resonant decay, for instance $D^+ \to K^*(892)\pi^+$, $K^*(892) \to K^-\pi^+$. Interference effects between these various Breit-Wigner resonances leads to interesting patterns for $\Gamma$ across the Dalitz plot, as shown on 4. These models could depend on a plethora of parameters, partly because of the possibly large number of strong resonance that could contribute to a particular decay mode [10]. The GUI and the application interface (API) used to create lattice data type are very useful to build an EXPLORER module that expresses the complexity of these interefference patterns. One could conceivably study this problem by having a standalone program that generates the data on a 2D grid, and feed that data to a simpler graphical package, such as PAW [12]. However such an environment is not very interactive, as we have no GUI to quickly change the parameters of the model and get an immediate graphical feedback, while EXPLORER gives us the opportunity to fully explorer this complex parameter space.

---

the *Lattice* data structure is used to convert and handle volumetric data on a discrete lattice.

### E. 2D Adaptive Histogram, colored fits

This last example explores the link between statistical analysis fitting techniques and conventional Scientific Visualization. On one hand, we have processed HEP data, in the form of a list of events, each event characterized by a point on the Dalitz plot. On the other hand, we have a complex mathematical model describing the relative probability distribution across the same Dalitz plot. Conventional graphical techniques, such as scatter plots, can either display the raw data or, conversely, a Monte-Carlo generated sample based on the theoritical model, but can not do both effectively. Moreover, for high statistics sample, scatter plots become too dense and therefore unreadable. These difficulties arise primarly from the multi-dimensionality of the problem.

We use the 2D adaptive histograming technique to map the raw data onto the Dalitz plot. A 2D histogram is plot of two variables, in our case, the probability density at a given $M^2_{K\pi}$, $M^2_{\pi\pi}$ is mapped to the height of the corresponding bin. Thus, we have to deal a 3D object. An adaptive histogram is a histogram whose bins, rather than being spaced evenly, are partioned according to the density of the data. A threshold evaluates the number of fills at which a single bin is divided into additional bins. Thus, the resolution of an adaptive histogram is highest where the data is densest, and lowest where the data is sparse. We then have to express how the model fits the data. For each bins, a statistical quantity such as the local $\chi^2$, or the confidence level that measures how well the data agrees with the fit, can be computed based on the parameters obtained by a global fit. This 4th dimension is expressed through color, a standard visualization trick. The result is shown on figure 5. Another standard procedure is to histogram both the raw data, the theoretical fit, and being able to flip quickly between one view and the other. Finally, one can also histogram the difference (Data - fit).

## IV. SUMMARY AND CONCLUSIONS

HEP visualization should be considered a research tool. Since the physicist needs to stay very close to his own private code, the API shell is essential. The GUI builder is also a heavily used feature. An important point in our analysis is the need for "interactive simplicity". The more glitzy aspects of such programs are sometimes completely superfluous to the HEP users. For instance, the interpretation of a colored rendering can sometimes be completely washed out by inappropriate use of lighting in the *Render* module. In fact, many of us have an almost instinctive suspicious reaction to glossy and glitzy pictures. A very simple X-Y curve sometimes describes the physics better than an impressive, colorful, 3D set of objects in the *Render* module. If so, a good visualization system should supply user interactions by providing simple access to color mapping and selection of which data to display. Currently, although simple 2D graphing modules are available (such as *GnuPlot*), they do not have the required level of quality and "interactivity", such as rescaling by axis dragging, popup menus to display statistical information, and so forth.

The inherently modular nature of Explorer makes it useful for large collaborations. Different pieces can be developed by different groups, and integrated fairly easily. In addition, Explorer offers a useful Intermediate File Format, the *Inventor File*, which has proven useful for remote collaborators to share their results. Data usually sent to the Render module can be sent in different ways into an ASCII Inventor File. The Inventor File can then be e-mailed to collaborators, who can look at it with the SGI program *ivview*.

A common misperception of visualization is that it should be considered as a nice and impressive presentation tool. Unfortunately, the best media to record and present a visualization session is not a static printout but a video clip. Although video technology is now cheap and easy to handle, and the non-professional quality is probably good enough for scientific presentations, it is rarely used in HEP conferences. Thus, one is left with static color images, whose resolution rarely exactly matches what's on the screen, and of course becomes completely unreadable when transfered into black & white photocopies.

Obviously, no such complex research tools are perfect. For instance, it is clear that some HEP-specific modules for SGI-EXPLORER have to be written, such as a built-in knowledge of Maxwell's equation used in tracing particles in magnetic fields. In addition, the lack of good statistical tools, based on a complete mathematical library, should be addressed[4]. Although Scientific Visualization is great for bringing a global cognition on particular problem, the real challenge is to be able to focus quickly and easily on a quantitive issue, and have direct acces to the critical numbers. For example, in SGI-EXPLORER, how does one easily get numerical feedback on the *Render* transformation matrix, and a exact scale of the display, as it does not come up with a set of 3D axes? This information is available to a C or C++ programmer (through the *cxPick* data structure), and making it readily available is the next crucial step. Another interesting issue is the use of of color, when mapped to a continuous variable: Although very intuitive and appealing, color is often inaccurate or subjective[5].

To conclude, although HEP data analysis has specific needs, conventional Scientific Visualization packages such as SGI-EXPLORER or DATA EXPLORER can be used successfully, particularly in the context of Event Displays. This technology should be perceived as a next step towards greater cognition of our data, a step as important as the introduction of 2D, core-based, computer graphics introduced in the late sixties, or 3D color wire-frame graphics in the seventies.

The examples described above could not have been written without the support of Fermilab experiment 687 (a study of Charm photoproduction at high energy) and D0, one of the two large Collider detector experiments at Fermilab. Other groups have done similar work [6], and other Scientific visualization systems (SCIAN, from FSU-SCRI) besides SGI-

---

[4]We obviously welcomed the support of NAG in this matter

[5]Improving the *Legend* display is more relevant than naming dozens of nice colors in the *ColorEditor*!

EXPLORER have been used at D0 [7]. We also whish to thank the Fermilab Computing Division for their support. In particular, the comments and ideas from Mark Edel and Patricia McBride are warmly acknowledged.

# REFERENCES


[1] IRIS-EXPLORER$^{TM}$ has been written at Silicon Graphics, and is now available through the NAG consortium.

[2] Data Explorer$^{TM}$ is a trademark of IBM.

[3] AVS$^{TM}$ is a trademark of Advanced Visualization Systems, Inc.

[4] S. Wolbers and F. Rinaldo (Fermilab), FERMILAB-PUB-93-006 Computers in Physics, Vol 7, 184-190 (March/Apr, 1993))

[5] Mark Edel*et al*, *Using Workstation in HEP, X-Window, MOTIF and Nirvana project*, FERMILAB-CONF-92-259, Oct. 1992, presented at the at 10th International Conference on Computing in High Energy Physics (CHEP 92), Annecy, France, 21-25 Sept 1992. See also, *HistoScope User's Guide* Fermilab Computing Division Library, Product # SP0034.

[6] Joe Boudreau, from the CDF collaboration, Private communication.

[7] S. Youssef, from the D0 collaboration, Private communication.

[8] R.K. Bock, H. Grote, D. Notz and M. Regler *Data analysis techniques for high-energy physics experiments* p.361.

[9] P.L. Frabetti *et al*, Nucl. Instrm. Methods A 320 (1992) 519

[10] P.L. Frabetti *et al.*, Phys. Lett. B**331**, (1994) 217

[11] Phys. Rev. D50 ( 1994) 1290.

[12] R.Brun *et al Physics Analysis Workstation* CERN Program Library entry Q121.

[13] S. Glenn, D0 Note 2039, *Simulation of the D0 Scintillating Fiber Trigger*, Jan, 1994.

## FIGURES

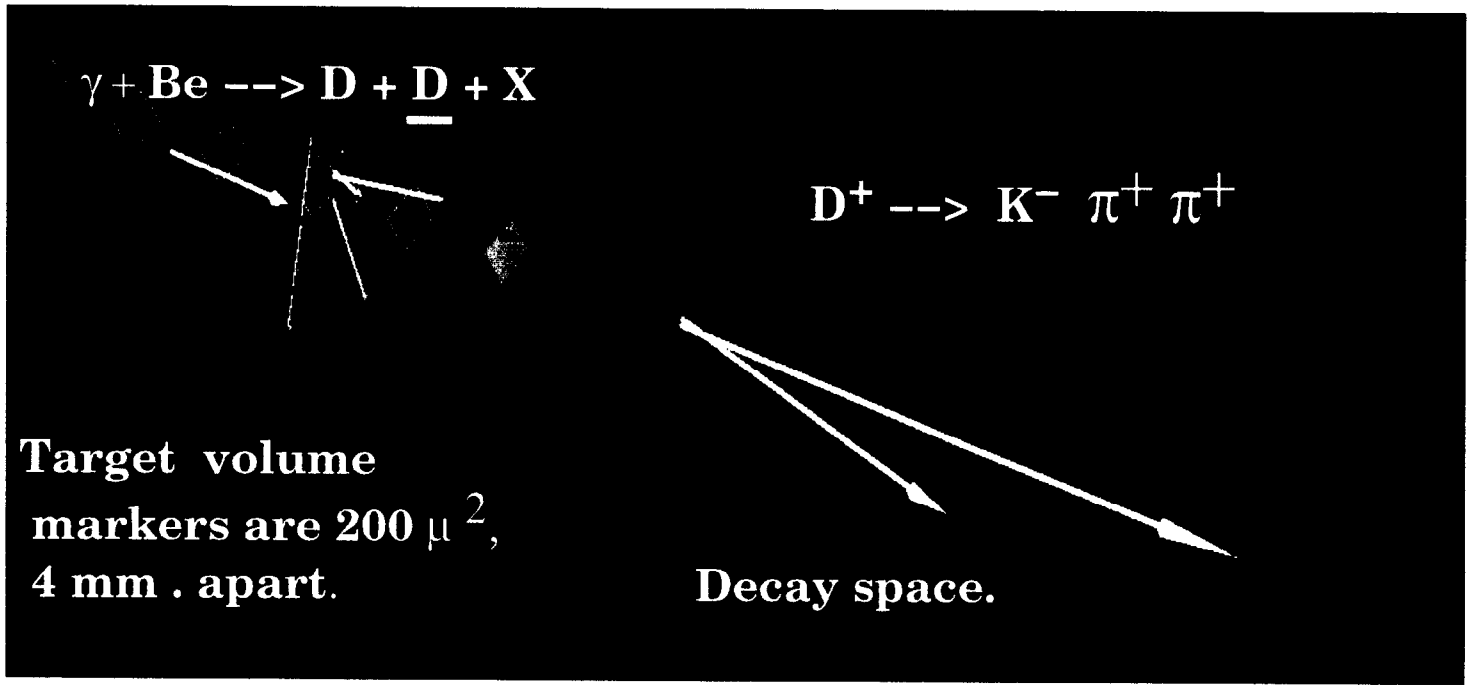
FIG. 1. Example of the D0 Fiber-tracker display. Small blue circles are the fibers, red markers are hit centroids, green lines are particle traces, red lines are high momentum traces that can be used in the triggering system.

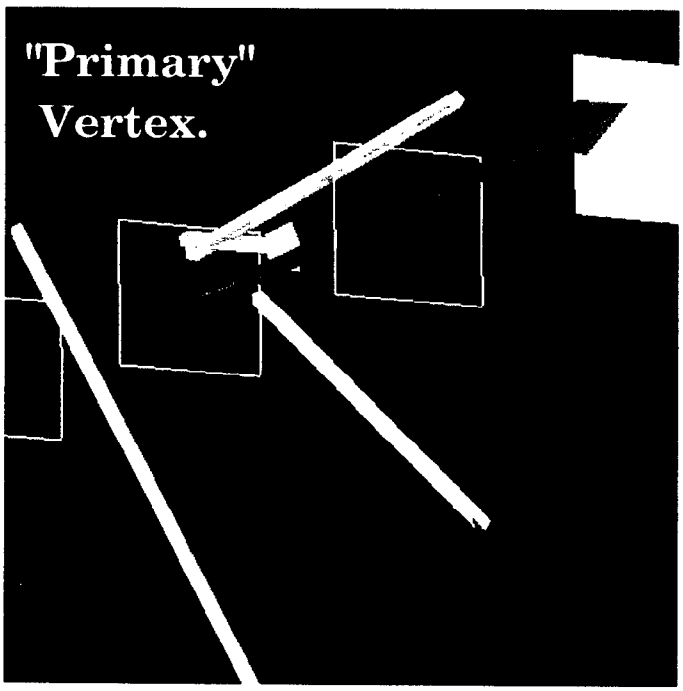
FIG. 2. A typical Charm event is shown. Top are target and decay region, showing a D vertex cleanly separated from the production target. Below, the primary vertex (left) is poorly defined due to the antiCharm decay vertex, while the other secondary vertex is characterized by a good vertex $\chi^2$.

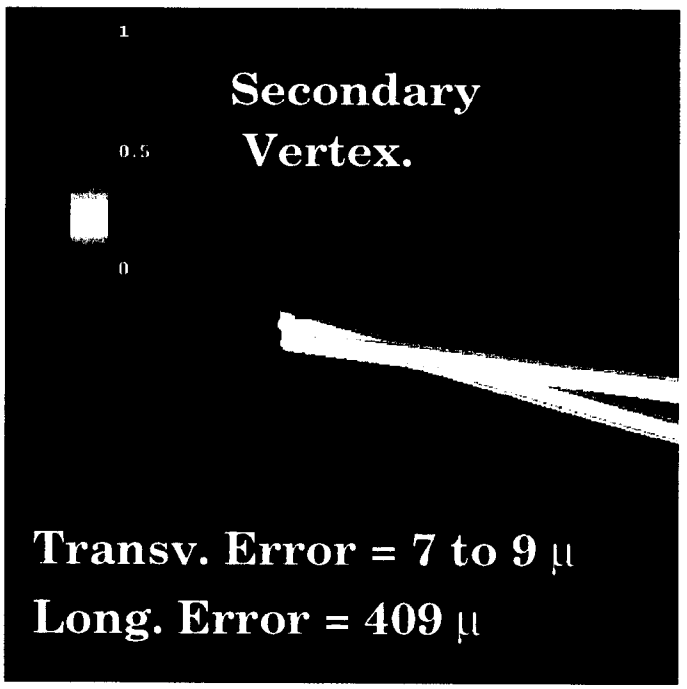
FIG. 3. A very simple SGI-Explorer map used to demonstrate the kinematics of a 3-body decay. The control module (top left) generate a 2D image representing the Dalitz plot (bottom left), and a simple geometry featuring 4 spheres, whose radius is proportional to their mass. In this case, the D (blue) decays into a Kaon (green) and two pions (red). The event can be viewed in the center of mass (top right) or in the center of mass of two daughter particles (bottom right).

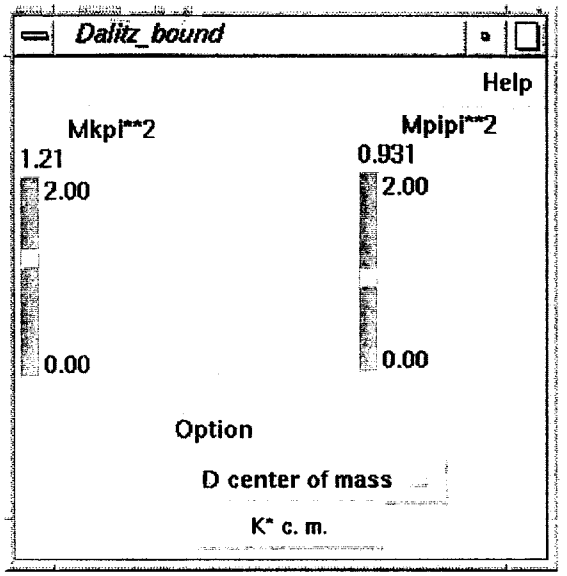
FIG. 4. A representation of the decay probability across the Dalitz plot,for the transition $D^+ \rightarrow K^-\pi^+\pi^+$, where the subresonances $K*(892)$ and the $K*(1430)$ interferes with a non-resonant component (N.R.). The relative phases are adjusted to reflect the data on the leftside plot. On the rightside plot, the $K*(892)$ phase relative to the N.R. is flipped by deg 180.

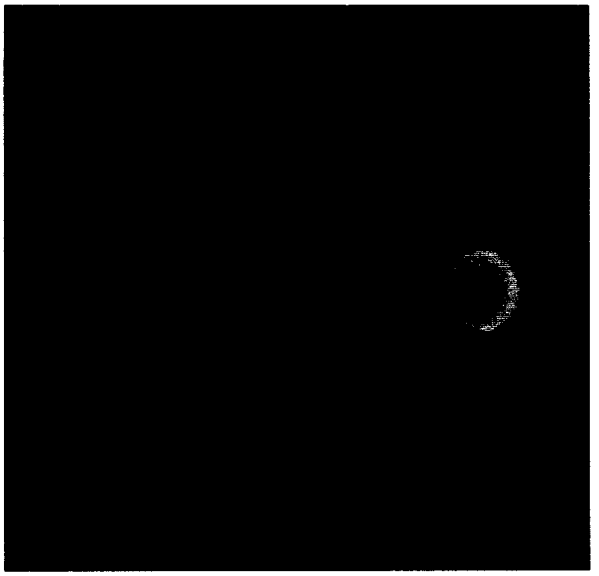
FIG. 5. The Adaptive Histogram Dalitz plot. The color of each bin is mapped to the square of the deviation of the data with the respect to the fit, wheighted by the statistical error. For instance, red bins are within statistical error, while dark blue ones are more than $5.\sigma$ away, a significant deviation. The data, the result of the fit and (data - fit) are shown in the top left corner, top right and lower left, respectively. On these last two plots, the relative phases are adjusted to reflect the data. On the rightside plot, the $K*(892)$ phase relative to the N.R. is flipped by deg 180, giving obviously an unacceptable fit in broad region of the Dalitz plot.

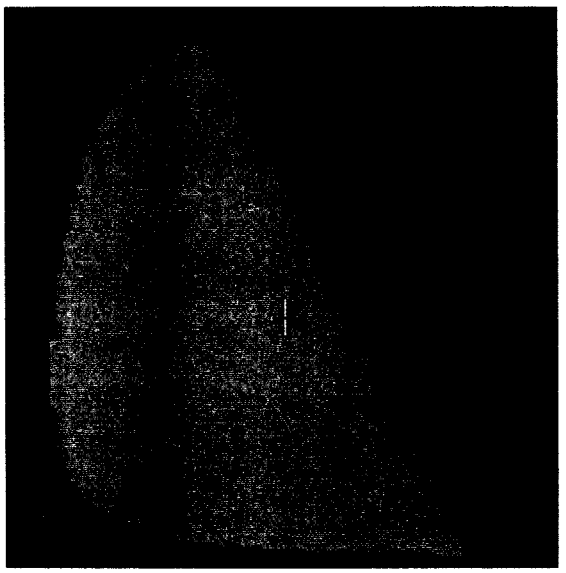

γ + Be --> D + D + X
D+ --> K− π+ π+
Target volume
markers are 200 μ 2,
4 mm . apart.
Decay space.

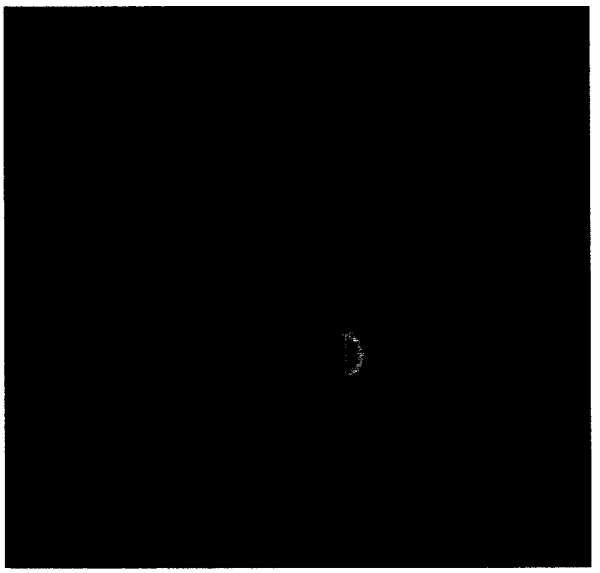

"Primary"
Vertex.

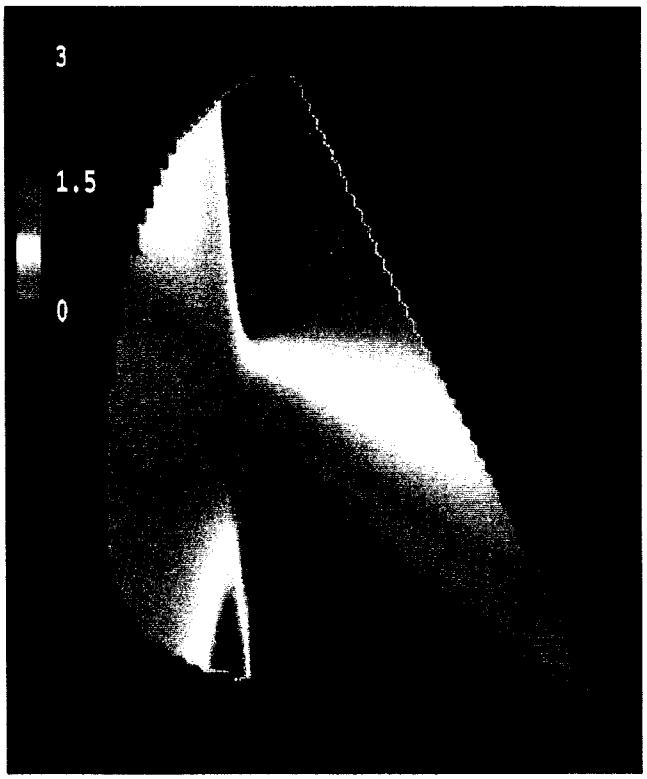

1
0.5
0
Secondary
Vertex.
Transv. Error = 7 to 9 μ
Long. Error = 409 μ

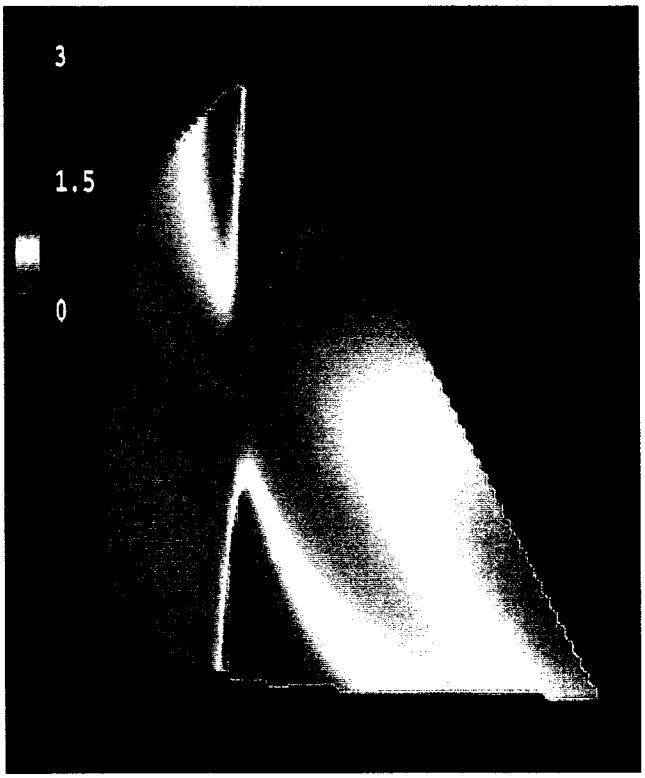

Dalitz_bound
Help
Mkpi**2
1.21
2.00
0.00
Mpipi**2
0.931
2.00
0.00
Option
D center of mass
K* c. m.








3
1.5
0


3
1.5
0