Generic Large Scale 3D Visualization of Accelerators and Beam Lines

Andreas Adelmann and Derek Feichtinger

Paul Scherrer Institut (PSI),
CH-5323 Villigen, Switzerland
{Andreas.Adelmann, Derek.Feichtinger}@psi.ch
http://www.psi.ch

Abstract. We report on a generic 3D visualization system for accelerators and beam lines, in order to visualize and animate huge amount of multidimensional datasets.
The phase space data on together with survey information obtained from MAD9P runs, are post-processed and then translated into colored ray-traced POV-Ray movies. We use HPC for the beam dynamic calculation and for the trivially parallel task of ray-tracing a huge number of animation frames.
We show various movies of complicated beam lines and acceleration structure, and discuss the potential use of such tools in the design and operation process of future and present accelerators and beam transport systems.

1 Introduction

In the accelerator complex of the Paul Scherrer Institut the properties of the high intensity particle beams are strongly determined by space charge effects. The use of space charge effects to provide adequate beam matching in the PSI Injector II and to improve the beam quality in a cyclotron is unique in the world. MAD9P (**m**ethodical **a**ccelerator **d**esign version **9** - **p**arallel) is a general purpose parallel particle tracking program including 3D space charge calculation. A more detailed description of MAD9P and the presented calculations is given in [1]. MAD9P is used at PSI in the low energy 870 keV injection beam line and the separate sector 72 MeV isochronous cyclotron (Injector II), shown in Fig. 1, to investigate space charge dominated phenomena in particle beams.

2 The mad9p Particle Tracker

2.1 Governing Equations

In an accelerator/beam transport system, particles travel in vacuum, guided by electric or magnetic fields and accelerated by electric fields. In high-current accelerators and transport systems the repulsive coulomb forces due to the space

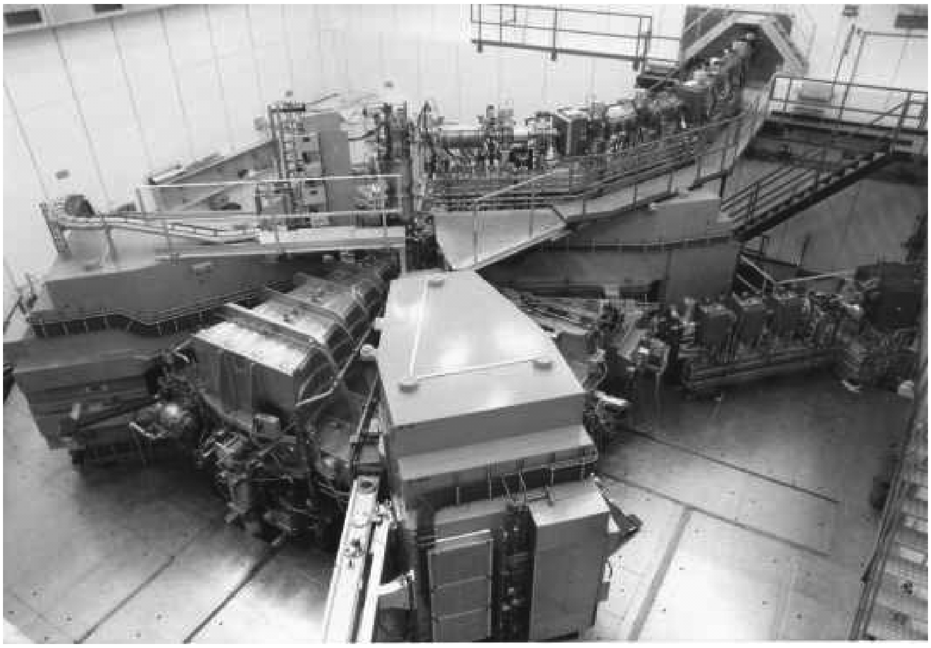

Fig. 1. PSI Injector II 72 MeV cyclotron with beam transfer lines

charge carried by the beam itself play an essential role in the design of the focusing system, especially at low energy. Starting with some definitions, we denote by $\Omega \in \mathcal{R}^3$ the spatial computational domain, which is cylindrical or rectilinear. $\Gamma = \Omega \times \mathcal{R}^3$ is the six dimensional phase space of position and momentum. The vectors $\mathbf{q}$ and $\mathbf{p}$ denote spatial and momentum coordinates. Due to the low particle density and the 'one pass' character of the cyclotron, we ignore any collisional effects and use the collisionless Vlasov Maxwell equation:

$$\partial_t f + \frac{\mathbf{p}}{m} \cdot \partial_{\mathbf{q}} f - (\partial_{\mathbf{q}} U + e \partial_{\mathbf{q}} \phi) \cdot \partial_{\mathbf{p}} f = 0. \tag{1}$$

Here the first term involving U represents the external forces due to electric and magnetic fields

$$U = \mathbf{E}(\mathbf{q}; t) + \frac{\mathbf{p}}{m} \times \mathbf{B}(\mathbf{q}; t) \tag{2}$$

and from Maxwell's equation we get:

$$\nabla \times \mathbf{E} + \frac{\partial \mathbf{B}}{\partial \mathbf{t}} = \mathbf{0}, \nabla \cdot \mathbf{B} = \mathbf{0}. \tag{3}$$

The external acting forces are given by a relativistic Hamiltonian $\mathcal{H}_{ext}$, where all canonical variables are small deviations from a reference value and the Hamiltonian can be expanded as a Taylor series. This is done automatically by the use of a *Truncated Power Series Algebra Package* [2], requiring no further analytical expansion.

The self-consistent Coulomb potential $\phi(\mathbf{q};t)$ can be expressed in terms of the charge density $\rho(\mathbf{q};t)$, which is proportional to the particle density

$$n(\mathbf{q};t) = \int d\mathbf{p} f(\mathbf{q},\mathbf{p};t) \text{ using } \rho(\mathbf{q},\mathbf{p};t) = en(\mathbf{q};t) \tag{4}$$

and we can write:

$$\phi(\mathbf{q};t) = \int_{\Omega} d\mathbf{q}' \frac{\rho(\mathbf{q}';\mathbf{t})}{|\mathbf{q}-\mathbf{q}'|}. \tag{5}$$

The self-fields due to space charge are coupled with Poisson's equation

$$\nabla \cdot \mathbf{E} = \int f(\mathbf{q},\mathbf{p};t) d\mathbf{p}. \tag{6}$$

2.2 Parallel Poisson Solver

The charges are assigned from the particle positions in continuum onto the grid using one of two available interpolation schemes: cloud in cell (CIC) or nearest grid point (NGP). The rectangular computation domain $\Omega := [-L_x, L_x] \times [-L_y, L_y] \times [-L_t, L_t]$, just big enough to include all particles, is segmented into a regular mesh of $M = M_x \times M_y \times M_t$ grid points. Let Ω^D be rectangular and spanned by $l \times n \times m$ with $l = 1 \ldots M_x$, $n = 1 \ldots M_x$ and $m = 1 \ldots M_t$. The solution of the discretized Poisson equation with $\mathbf{k} = (l, n, m,)$ reads

$$\nabla^{2^D} \phi^D(\mathbf{k}) = -\frac{\rho^D(\mathbf{k})}{\epsilon_0}, \mathbf{k} \in \Omega^D. \tag{7}$$

The serial PM Solver Algorithm is summarized in the following algorithm:

PM Solver Algorithm

```
▷ Assign particle charges $q_i$ to nearby mesh points to obtain $\rho^D$
▷ Use FFT on $\rho^D$ and $G^D$ (Green's function) to obtain $\widehat{\rho}^D$ and $\widehat{G}^D$
▷ Determine $\widehat{\phi}^D$ on the grid using $\phi^D = \rho^D * G^D$.
▷ Use inverse FFT on $\widehat{\phi}^D$ to obtain $\phi^D$
▷ Compute $\mathbf{E}^D = -\nabla\phi^D$ by use of a second order finite difference method
▷ Interpolate $\mathbf{E}(\mathbf{q})$ at particle positions $\mathbf{q}$ from $\mathbf{E}^D$
```

The parallelization of the above outlined algorithm is done in two steps: first Ω^D is partitioned into subdomains $\Omega^D_k,\ k = 1 \ldots p$ where p denotes the number of processors. On each processor there are N/p particles using a spatial particle layout. The spatial layout will keep a particle on the same node as that which contains the section of the field in which the particle is located. If the particle moves to a new position, this layout will reassign it to a new node when necessary.

This will maintain locality between the particles and any field distributed using this field layout, and it will help keep particles which are spatially close to each other local to the same processor as well.
The second important part is the parallel Fourier Transformation, which allows us to speed up the above described serial PM solver algorithm. For more details on the implementation and performance see [3] [1].

To integrate the particle motion, we use a second order split-operator scheme [4]. This is based upon the assumption that one can split the total Hamiltonian in two solvable parts: $\mathcal{H}_{ext}$ and the field solver contribution $\mathcal{H}_{sc}$. For a step in the independent variable τ one can write:

$$\mathcal{H}(\tau) = \mathcal{H}_{ext}(\tau/2)\mathcal{H}_{sc}(\tau)\mathcal{H}_{ext}(\tau/2) + \mathcal{O}(\tau^3) \tag{8}$$

2.3 Design or Reference Orbit

In order to describe the motion of charged particles we use the local coordinate system seen in Fig. 2. The accelerator and/or beam line to be studied is described as a sequence of beam elements placed along a reference or design orbit. The

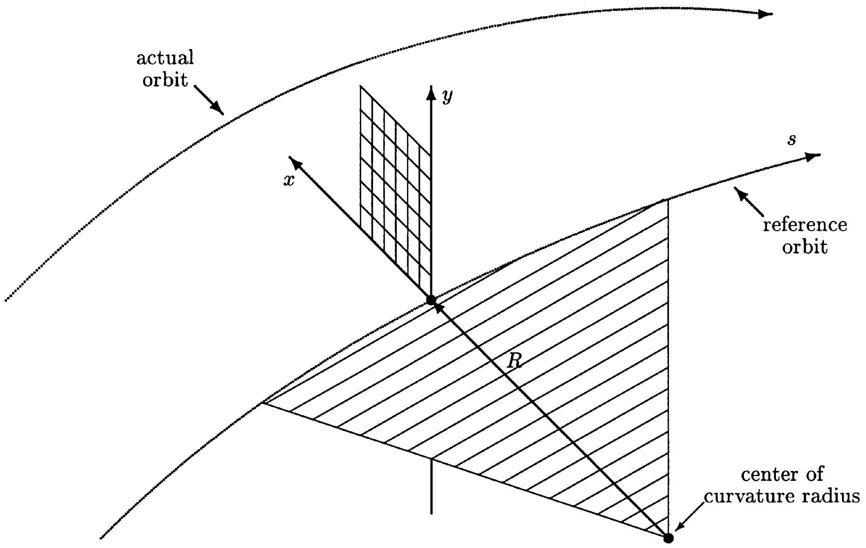

Fig. 2. Local Reference System

global reference orbit (see Fig. 3), also known as the design orbit, is the path of a charged particle having the central design momentum of the accelerator through idealized magnets with no fringe fields.

The reference orbit consists of a series of straight sections and circular arcs. It is defined under the assumption that all elements are perfectly aligned along the design orbit. The accompanying tripod (Dreibein) of the reference orbit spans a local curvilinear right handed system (x, y, s).

2.4 Global Reference System

The local reference system (x, y, s) may thus be referred to a global Cartesian coordinate system (X, Y, Z). The positions between beam elements are numbered $0, \dots, i, \dots n$. The local reference system (x_i, y_i, s_i) at position i, i.e. the displacement and direction of the reference orbit with respect to the system (X, Y, Z) are defined by three displacements (X_i, Y_i, Z_i) and three angles $(\Theta_i, \Phi_i, \Psi_i)$.

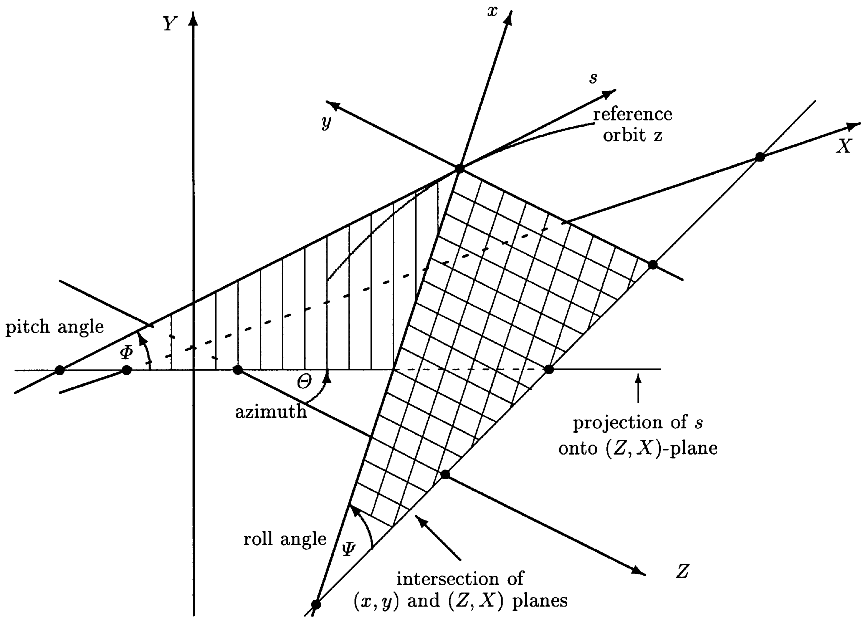
Y
x
s
y
reference orbit z
X
pitch angle
Φ
Θ
azimuth
projection of s onto (Z, X)-plane
roll angle
Ψ
intersection of (x, y) and (Z, X) planes
Z

Fig. 3. Global Reference System

The above quantities X, Y and Z are displacements of the local origin in the respective direction. The angles (Θ, Φ, Ψ) are **not** the Euler angles. The reference orbit starts at the origin and points by default in the direction of the positive

Z-axis. The initial local axes (x, y, s) coincide with the global axes (X, Y, Z) in this order. The displacement is described by a vector $\mathbf{v}$ and the orientation by a unitary matrix $\mathcal{W}$. The column vectors of $\mathcal{W}$ are unit vectors spanning the local coordinate axes in the order (x, y, s). $\mathbf{v}$ and $\mathcal{W}$ have the values:

$$\mathbf{v} = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}, \qquad \mathcal{W} = \mathcal{S}\mathcal{T}\mathcal{U} \tag{9}$$

where

$$\mathcal{S} = \begin{pmatrix} \cos\Theta & 0 & -\sin\Theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\Theta \end{pmatrix}, \quad \mathcal{T} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\Phi & \sin\Phi \\ 0 & -\sin\Phi & \cos\Phi \end{pmatrix}, \tag{10}$$

$$\mathcal{U} = \begin{pmatrix} \cos\Psi & -\sin\Psi & 0 \\ \sin\Psi & \cos\Psi & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{11}$$

Let the vector $\mathbf{r}_i$ be the displacement and the matrix $\mathcal{S}_i$ be the rotation of the local reference system at the exit of the element i with respect to the entrance of the same element.

When advancing through a beam element i, one can compute $\mathbf{v}_i$ and $\mathcal{W}_i$ by the recurrence relations

$$\mathbf{v}_i = \mathcal{W}_{i-1}\mathbf{r}_i + \mathbf{v}_{i-1}, \qquad \mathcal{W}_i = \mathcal{W}_{i-1}\mathcal{S}_i. \tag{12}$$

This relation (12) is used in the generation of ray-tracing movies.

3 Architecture of mad9p and accelVis

Today we use Linux Farms (also known as Beowulf clusters) with up to 500 Processors (Asgard ETHZ) as well as traditional symmetric multiprocessor (SMP's) machines like IBM SP-2 or SGI Origin 2000. Having such a wide variety of platforms available put some non negligible constraint on the software engineering part of a simulation code. MAD9P is based on two frameworks:[1] CLASSIC [6] and POOMA [3], shown schematically in Fig. 4. CLASSIC deals mainly with the accelerator physics including a polymorphic differential algebra (DA) package and the input language to specify general complicated accelerator systems. In order to ease the task of writing efficient parallel applications we rely on the POOMA framework which stands for Parallel Object-Oriented Methods and Applications. POOMA provides abstraction for mathematic/physical quantities

[1] We use the notion of framework in the following sense: a framework is a set of co-operating classes in a given problem frame. On this and other software engineering concepts see [5]

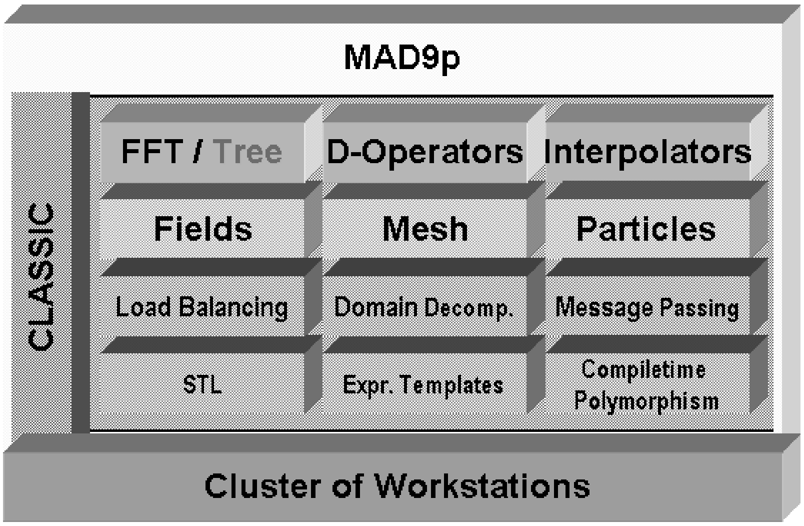

Fig. 4. Architectural overview on MAD9P

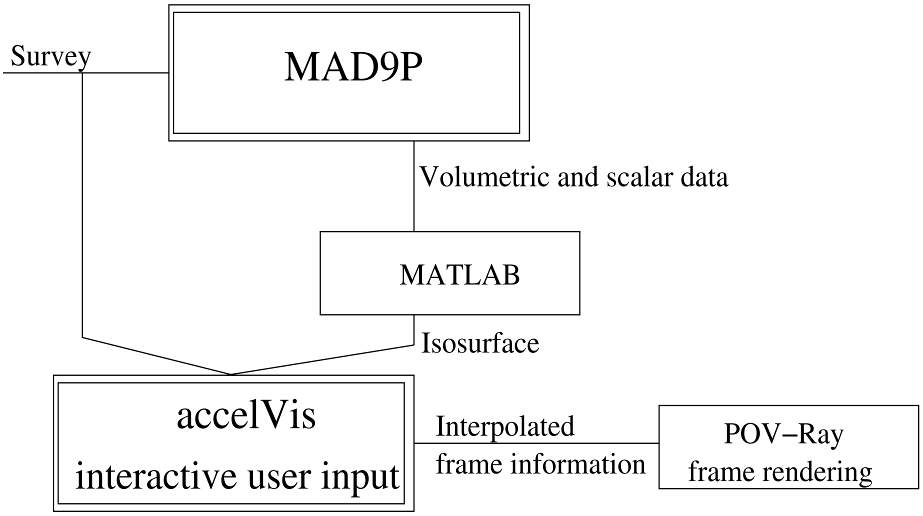

Fig. 5. Data flow between MAD9P

like particles, fields, meshes and differential operators. The object-oriented approach manages the complexity of explicit parallel programming; it encapsulates the data distribution and communication among real or virtual processors. POOMA and all the other components are implemented as a set of templated C++ classes. The computing nodes can be a single (real) cpu or a virtual node

(VNODE). Usually MAD9P uses the message passing interface MPI [7] in order to interconnect the individual nodes.

ACCELVIS is currently implemented using ANSI C. The program interfaces to the OpenGL graphics library and it's GLU and GLUT extensions to render the interactive 3D graphics. These libraries (or the compatible Mesa library) as well as POV-Ray [8] and MATLAB [9] are available on a wide range of platforms. Therefore, although the application was developed on a Red-Hat Linux System, only very minor modifications are necessary to transfer it to a variety of other architectures. The data-flow, involving MAD9P is shown schematically in Fig. 5.

3.1 Program capabilities

The ACCELVIS application enables the user to view a graphical interpretation of volumetric and scalar data provided by a MAD9P run. The reference trajectory and ISO-surfaces illustrating the particle density can be investigated interactively by gliding with a virtual camera through a representation of the accelerator (Fig. 6). By defining a trajectory for the camera the user is able to produce high quality animations for teaching and illustration purposes.

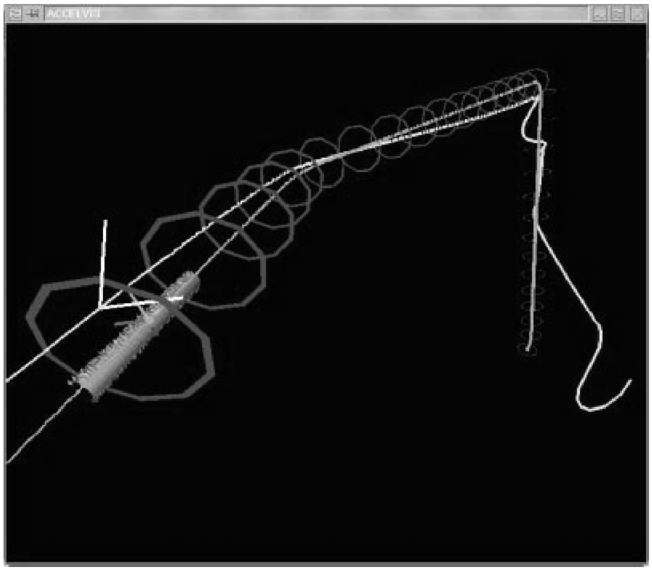

Fig. 6. ACCELVIS view of the particle cloud ISO-surface, the beam trajectory (red line), the camera trajectory (yellow line), and the camera viewing orientation (white lines)

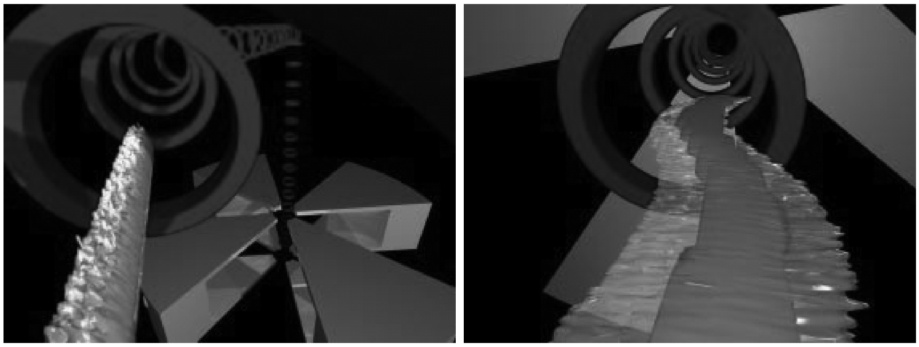

Fig. 7. Animation frames generated from the ACCELVIS setup shown in Fig. 6. Two ISO-surfaces for cloud core (red) and halo (yellow) are used in visualizing the particle density

3.2 Program Input

The reference trajectory is read in as a sequence of displacement vectors v_i and the matching rotation angles Θ_i, Φ_i, Ψ_i defining origin and orientation of the local coordinate systems. The particle density data ϕ^D, or other scalar data like rms quantities (beam size, emittance), is taken from a MAD9P run.

3.3 Information Processing

To obtain a fluid animation of the particle clouds, it is necessary to interpolate between displacements as well as between rotations yielding the local coordinate systems along the reference trajectory. A simple spline interpolation was chosen for the displacement vectors of the reference particle. The rotations were interpolated through a spline interpolation of their quaternion representations since this provides smoother interpolation and avoids some of the problems that appear if the defining angles Θ_i, Φ_i, Ψ_i or elements of the rotation matrix are directly used [10].

The particle density is processed by interfacing to a MATLAB [9] script which transforms the data into a series of connecting triangles representing a density ISO-surface. To increase the smoothness of the generated graphics, the surface normal vectors at every triangle corner are also calculated (This information is customarily used by 3D visualization surface lighting models). Currently two ISO-surfaces are used in each frame of the animation to provide more insight into the density distribution. The surface gained from the higher iso value is termed the cloud core, the other the cloud halo. The halo is rendered translucent (Fig. 7).

The camera view is represented by yet another local coordinate system. For the production of the high quality animations a number of camera views are defined by interactively moving the camera to the desired position for the respective simulation frame. The camera views are then interpolated over the whole course of the simulation using the same procedure as described above for the interpolation of the reference trajectory orientations.

3.4 Generation of the Animations

The application creates input and command files for the free and commonly used POV-Ray [8] ray-tracing program. If desired a series of command files are produced where each one assigns a subset of the frames to be rendered to the nodes of a computing cluster. This trivially simple parallelization scheme enabled us to compile the 1600 frames ($320 * 240$ pixels each, 24 bit color depth) of this current animation in a rendering time of about 20 minutes on the 64 node Merlin Linux cluster at PSI. By using standard software the frames can be converted to one of the common movie formats (usually MPEG).

4 Application to the PSI Injector II Cyclotron

The use of high level visualization is one of the key aspects in interpretation of multi-dimensional datasets. In the presented approach, it was attempted to tightly couple large scale accelerator system simulations (using MAD9P) with advanced visualization techniques (ACCELVIS). Using principles of generality in the design of both components, one can easy adapt ACCELVIS to other accelerator system simulation frameworks.

First simulations of complicated structures, as shown in Fig. 1, were successful. The application area of such visualization ranges from education to the (re)design phase of existing or new machines. This might evolve into an indispensable tool for the use in the accelerator control-room.

References

1. Adelmann, A.: 3D Simulations of Space Charge Effects in Particle Beams. PhD thesis, ETH (2002)
2. Berz, M.: Modern Map Methods in Particle Beam Physics. Academic Press (1999)
3. Cummings, J., Humphrey, W.: Parallel particle simulations using the POOMA framework. In: 8th SIAM Conf. Parallel Processing for Scientific Computing. (1997)
4. J.M. Sanz-Serna, M.C.: Numerical Hamiltonian Problems. Chapman and Hall (1994)
5. E. Gamma, *et.al.*: Design Patterns. Addison Wesley (1995)
6. Iselin, F.: The classic project. Particle Accelerator **Vol. 54,55** (1996)
7. William Gropp, *et.al.*: Using MPI : portable parallel programming with the message-passing interface. Cambridge, Massachusetts : MIT Press (1999)
8. the POV-Team: Pov-ray 3.1 (1999) ray-tracing software.
9. MathWorks: MATLAB 6.1. URL: http://www.mathworks.com (2001)
10. E. B. Dam, M. Koch, M.L.: Quaternions, interpolation and animation. Technical Report DIKU-TR-98/5, Department of Computer Science, University of Copenhagen, Universitetsparken 1, DK-2100 Kbh, Denmark (1998)