Joint EUROGRAPHICS - IEEE TCVG Symposium on Visualization (2003)
G.-P. Bonneau, S. Hahmann, C. D. Hansen (Editors)

MCMR: A Fluid View on Time Dependent Volume Data

Wim de Leeuw and Robert van Liere

Center for Mathematics and Computer Science CWI, Amsterdam, Netherlands

Abstract

Mass Conservative Motion Reconstruction is a new method for estimating motion in time dependent volume data. A time dependent vector field representing the movement of the data is computed from a sequence of scalar volume data sets. The principle of mass conservation in a continuum is used during the reconstruction. Standard flow visualization techniques are used for the visualization of the derived vector field.
This paper presents the underlying concepts of MCMR, its implementation, its accuracy and applicability.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation I.3.6 [Computer Graphics]: Methodology and Techniques
Keywords: conservation of mass, motion reconstruction, vector fields, flow visualization, volume visualization.

1. Introduction

Visualization and analysis of time dependent volume data is of key importance for understanding many scientific problems such as confocal microscopy and material transport simulations, e.g. fluid mixing, convection and combustion. An important aspect in these problems is movement or transport of data. However, often there will be no explicit information about movement available in the data itself. Animated volume rendering techniques are often not sufficient for understanding complex movement patterns in the data.

We present MCMR: a mass conservative motion reconstruction method for estimating motion in time dependent volume data. The objective is to reconstruct the velocity field representing the movement of the input data. MCMR can be used in time dependent volume data in which the phenomena within the input data obey the principle of mass conservation in a continuum. MCMR is particularly useful in volumes where complex movements occur, and where no a-priori knowledge about movement is available from other sources.

The output of MCMR is a time dependent vector field. Standard flow visualization techniques are used to visualize the vector field. For example, particle paths and stream surfaces can be used to show the movement and evolution of interesting phenomena. In addition, the resulting vector field can serve as a basis for data analysis. Tracking a feature in the original volume data set is reduced to the computation of a particle path in the generated vector field.

2. Related work

The derivation of a vector field from two dimensional image data was introduced as optical flow by Gibson[1]. Optical flow is the distribution of apparent velocities from movement of brightness patterns in an image. It provides information about the spatial arrangement of the objects in the image and the rate of change of this arrangement. The construction of an optical flow field from digital image sequences was addressed by Berthold et al. and Nagel[2, 3]. These algorithms use intensity gradients in the spatial and temporal domain in combination with additional constraints to obtain the velocity. One choice for additional constraints is the minimization of the velocity gradients to ensure the smoothness of the velocity field. Other application dependent criteria can also be used as constraints.

The problem of the derivation of a vector field from two subsequent data sets can also be treated as a non-rigid registration problem. The vector field and displacement field are equivalent and can be easily transformed into each other. Much work has been done in this area to match two volume data sets, for example matching scans of two different patients brains using elastic deformation. Several approaches to this problem have been proposed such as opti-

cal flow-based methods and the use of smooth basis functions. An overview of non-rigid registration methods can be found in Dawant[4]. MCMR is most closely related to elastic deformation methods which rely on a physical model of the underlying data. Gee uses the Navier equation for elastically isotropic and homogeneous substances to govern the deformation[5]. Davatzikos adapted this method to permit the spatial adaptation of the elastic properties of the material[6]. Christensen et al. use an alternative approach based on viscous-fluid transformations[7].

MCMR differs from these approaches in two ways. Firstly, MCMR uses conservation of mass in a continuum as the underlying physical model for the movement of the data. This leads to a different problem formulation and solving strategy. Secondly, although MCMR and the mentioned techniques are general purpose, the application domain is different. The focus for the mentioned techniques is on medical applications and specifically brain matching. MCMR focuses on time series of biological data, specifically the movement of chromatin during formation of the cell nucleus of a newly formed cell.

In a previous paper, we introduced a different method for the reconstruction of a vector field from scalar volume data[8]. The method, called BM3D, uses a block matching algorithm to find corresponding regions of data in two successive time steps of volume data. The displacement of the corresponding regions determine the movement in the data. BM3D can be seen as the 3D extension of the well known 2D block matching algorithms found, for example, in MPEG encoding. MCMR differs from BM3D in two ways. Firstly, instead of comparing rigid blocks of data, MCMR takes deformation occurring in the data into account. Secondly, BM3D matches data blocks based solely on local information of data values whereas MCMR is a global technique based on the transport of data throughout the volume. Both differences result in a more accurate estimation of the motion, particularly when deformations of continuous mass movements are involved.

3. Method

3.1. Problem formulation

Assume a mass distribution ρ defined over a domain Ω. If the evolution of mass is continuous, then conservation of mass is satisfied:

$$\int_{C.S.} \rho \vec{V} \cdot d\vec{A} = \frac{\delta}{\delta t} \int_{C.V.} \rho d\vec{\omega} \tag{1}$$

This equation states that the rate of inflow of mass across a control surface *C.S.* is equal to the rate of accumulation of mass inside the volume *C.V.* enclosed by the control surface. In differential form, the conservation of mass is captured as:

$$\frac{\partial \rho}{\partial t} = -\nabla \cdot (\rho \vec{V}) \tag{2}$$

in which $\vec{V}$ is the vector field over Ω describing the movement of the mass distribution.

For volume data the mass distribution ρ is discretized in space $\vec{x}$ and time t as a sequence of scalar data sets $M(\vec{x},t)$. The differential form of the conservation of mass equation can be rewritten in discrete form as

$$M(\vec{x}, t+\Delta t) \;=\; \Phi\left(M(\vec{x},t),\, \vec{V}(\vec{x},t)\right)) \tag{3}$$

in which Φ describes the transport of mass (right hand side of equation 2).

The objective of MCMR is to compute the vector field $\vec{V}(\vec{x},t)$. Unfortunately, there are many vector fields that satisfy this objective. For example, consider figure 1. Two time steps of a sphere-like object are drawn in 2D as stippled and solid lines. The movement of mass can be described as a rigid rotation of the sphere (left), or as a deformation of the sphere (right). In this example both vector fields describing the movements would be a valid solution. Therefore, an additional constraint is needed to select one of the two vector fields. In the example, if a constraint that minimizes the movement of the data is chosen, then the solution in the right image would be preferred. When a constraint that assumes rigid objects is used, then the solution in the left image would be chosen. We denote this constraint as $S(\vec{V})$.

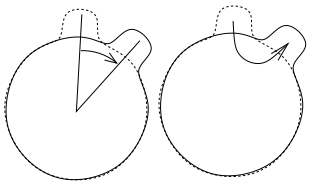

Figure 1: *The movement of a sphere-like object can be described as a rigid rotation (left) or as a deformation (right) of the sphere. If only the movement of mass is known, then an additional constraint over the vector field is need to distinguish between these movements.*

The computation of $\vec{V}(\vec{x},t)$ can now be formulated as an optimization problem:

$$\textit{minimize}\; D(M(\vec{x},t+\Delta t),\, \Phi(M(\vec{x},t),\vec{V}(\vec{x},t)))\; \textit{subject to}\; S(\vec{V})$$

$M(\vec{x},t+\Delta t)$	mass distribution at $t+\Delta t$
$\Phi(M(\vec{x},t),\vec{V}(\vec{x},t))$	computed mass transport
$D(M_1,M_2)$	difference metric over M_1 and M_2
$S(\vec{V})$	additional constraint over $\vec{V}(\vec{x},t)$

3.2. Implementation

The vector field is defined on a regular grid G of nodes over Ω. A mass distribution at time $t+\Delta t$ is computed by using $\vec{V}(\vec{x},t)$ to transport mass from $M(\vec{x},t)$. This is realized by sampling M on a regular grid at a resolution that is a multiple of the number of voxels in M. Each sample is advected using linear interpolation on G to a position in $M^\star$. The data value for each sample is added to the voxel at the position of the advected sample in $M^\star$. We denote the computed mass

distribution as $M^{\star}_{\vec{V}}(\vec{x}, t+\Delta t)$. The two dimensional case is illustrated in figure 2.

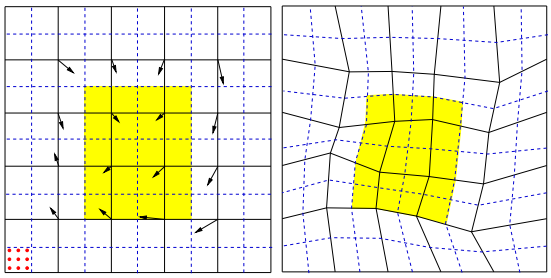

Figure 2: *Computing $M^{\star}_{\vec{V}}(\vec{x}, t+\Delta t)$: Φ is applied to a mass distribution at time t (shown in yellow, left) to compute a new mass distribution at time $t+\Delta$ (right). Black solid lines indicate nodes in the velocity grid G. Blue dashed lines indicate voxels in M. Red dots indicate the sampling pattern in each voxel.*

The problem formulation in section 3.1 indicated an additional constraint which is necessary to uniquely construct the vector field. The implementation defines the constraint as a tension function $S()$ over the velocity grid G. The measure of tension is the distance of the velocity vector to the average of the neighboring velocity vectors (see figure 3). The tension at node i, j, k is defined as:

$$|(v(i\pm 1, j, k)+v(i, j\pm 1, k)+v(i, j, k\pm 1))/6) - v(i, j, k)|$$

in which $v(i, j, k)$ denote the velocity at $\vec{x}$. The total tension in the vector field is the sum of all tension values for the individual nodes:

$$S(\vec{V}) \;=\; \sum_{i,j,k} S(i, j, k)$$

In this way, nodes in G are forced in the direction of a regular grid and a solution that minimizes the tension will result in a vector field that minimizes the movement of mass.

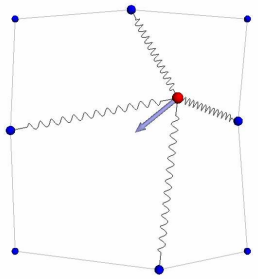

Figure 3: *Tension function used to minimize mass movement. Minimizing the tension function will result in a vector field that minimizes the movement of mass.*

The function to compare M_1 and M_1 is defined as

$$D(M_1, M_2) \;=\; \sum_{i \in voxels} |M_1(i) \;-\; M_2(i)|$$

This function simply computes the sum of differences between each corresponding voxel in M_1 and M_2.

Given the tension function $S()$ and difference function $D()$, the problem formulated in 3.1 is implemented as the minimization of

$$minimize\ D\left(M(t+\Delta t),\ M^{\star}_{\vec{V}}(t)\right) \;+\; S(\vec{V})$$

in which $\vec{V}(\vec{x}, t)$ is the independent variable. For each time step, the time dependent vector field $\vec{V}(\vec{x}, t)$ is computed using only data from two subsequent time steps $M(t)$ and $M(t+\Delta t)$.

It is possible to solve this minimization problem using, for example, Marquardt based fitting methods[9]. However, since the order of the problem is proportional to 3 times the number of the nodes in the velocity grid, these direct solution methods are prohibitively expensive to compute $\vec{V}(\vec{x}, t)$ on a grid with a fine resolution. Instead, our implementation uses a grid refinement scheme combined with a hill climbing algorithm to compute an optimal velocity field at each level of refinement. The hill climbing method was chosen instead of the Marquardt based methods because these methods require the evaluation of the gradient of the function. Evaluation of the function consists of a computation of $M^{\star}$ and $D()$, which would be very expensive. The number of evaluations required for the hill climbing method are significantly less.

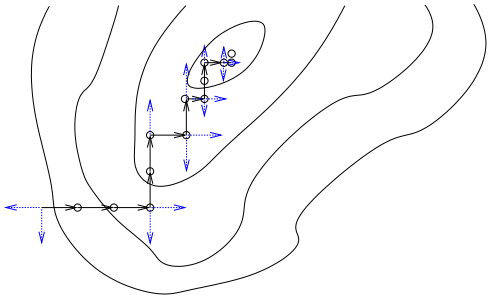

Figure 4: *Improving the fit of a particular node using hill climbing.*

Initially the vector field on a coarse grid is computed. The velocities found in the coarse grid are propagated as initial values on the finer grid. Figure 4 illustrates the hill climbing scheme in 2D. Each of the three velocity components are perturbed one at a time for a certain displacement, and if the fit improves the algorithms continues from that node. If no improvement occurs for a certain displacement, the node is subdivided. The fit is computed as $D\left(M(t+\Delta t), M^{\star}_{\vec{V}}(t)\right) \;+\; S(\vec{V})$ in which $M^{\star}_{\vec{V}}(t))$ computed using the perturbed vector field (see figure 2).

The pseudo-code of the grid refinement procedure is:

```
AdaptRefine(G)
{
```

```
   if resolution(G) > MAX_RES return

   for (i=0; i < MAX_VELO_ITER; i++)
      foreach   node ∈ G
         BestFit (node, XPOS, init_stepsize)

   AdaptRefine(RefineGrid(G))
}
```

The procedure `AdaptRefine()` describes the grid refinement procedure of the velocity grid. If `MAX_RES`, the finest resolution, is not reached a relaxation scheme is used to compute the velocity field. This is realized by computing `BestFit()` at each node of the refined grid for `MAX_VELO_ITER` iterations. Computing an optimal fit at one node may influence the fit at another node. However, when a large value for `MAX_VELO_ITER` is used, then the best fit on each node will converge to an optimal solution over the complete velocity grid.

After a velocity field is computed at one level, the grid is refined and `AdaptRefine()` is called recursively.

The pseudo-code of the fitting procedure is:

```
BestFit(node, dir, ssize)
{
   if (ssize < MIN_STEP_SIZE) return

   fit1 = ComputeFit()

   Peturb (node, ssize, dir)
   fit2 = ComputeFit ()

   if (fit1 < fit2)
   {
      fit1 = fit2
   }
   else
   {
      UndoPeturb(node, ssize, dir)

      if (SwapDirection (dir,ssize))
         dir = NewDirection(dir)
      else
         ssize = NewStepSize(ssize)
   }
   BestFit (node, dir, ssize)
}
```

This procedure finds the best fit for a node in the velocity grid using a hill climbing scheme. The vector at the node is peturbed for a certain step size in one of the six directions. `ComputeFit()` computes the difference function using the peturbed velocity field; i.e. $D\left(M(t+\Delta t), M^{\star}_{\vec{V}}(t)\right) + S(\vec{V})$ in which $M^{\star}_{\vec{V}}(t))$ is computed using the perturbed vector field (see figure 2). The direction is continued if the fit improves. If the fit does not improve, then `SwapDirection()` decides if a new direction or a new step size should be chosen (see figure 2). Possible directions are `{XPOS, XNEG, YPOS, YNEG, ZPOS, ZNEG}` The current implementation reduces the step size by a factor of 0.5.

Finally, `BestFit ()` is called recursively with the perturbed node and a new direction or step size.

4. Results

4.1. Synthetic data

An important aspect of the utility of MCMR is its accuracy. To get more insight into the accuracy of the method a number of synthetic data sets were generated. The movement of the data in these data sets simulate the movement of a parameterized Gaussian function G. The movement of the Gaussian is known analytically, it can be compared with the movement computed by MCMR.

The parameters of the Gaussian are: initial position $\vec{p}_0$, initial size σ_0, velocity $\vec{v}$, mass M, and diffusion rate R_d. Diffusion was used because it is an often occurring phenomenon in volume data. Diffusion is approximated by decreasing the maximum intensity in combination with an increasing sphere size such that the integral over all intensity values remain constant. This leads to the following equation for the Gaussian function :

$$G(\vec{x}) = \frac{M}{\sqrt{\pi^3}(\sigma_0 + R_d t)^3} exp\{-\left(\frac{|\vec{p}_0 + \vec{v}t - \vec{x}|}{\sigma_0 + R_d t}\right)^2\} \quad (4)$$

The initial conditions $\vec{p}_0$, σ_0, and R_d determine the movement of Gaussian over time. Data sets at a resolution of $64 \times 64 \times 64$ were generated.

Three metrics are used to judge the accuracy of the method:

1. Absolute velocity error: Compare the difference between the velocity of the Gaussian function and the MCMR constructed velocity field. The error is defined as the average norm of the difference vector at each node in the velocity grid that are within a distance of σ from the center of the Gaussian:

$$\frac{1}{N}\sum_{i=1}^{N} \mid \vec{v}(i) - \hat{\vec{v}}(i) \mid \quad (5)$$

   in which $\vec{v}(i)$ is the input velocity at node i, $\hat{\vec{v}}(i)$ is the MCMR computed speed at node i and N is the number of velocity samples.
2. Relative velocity error: The relative velocity error is computed by dividing the is absolute with the input speed of the Gaussian.
3. Mass fitting: Compare the mass distributions defined by the Gaussian and the MCMR estimated mass distribution. The error is defined as a fitting measure of how well the

MCMR method deforms the data at a time step t to fit the data at time $t + \Delta t$. The fitting metric is computed as

$$f = 1 - \frac{D(M(t),\ M^{\star}_{\vec{V}}(t))}{D(M(t),\ M(t\ +\ \Delta t))} \qquad (6)$$

in which $D()$ is the difference function given in section 3.2. The fitting metric is equal to 1 if the mass distribution at time $t + \Delta t$ perfectly matches the computed mass distribution at time t subject to the computed vector field $\vec{V}$

	no diffusion			diffusion		
$\vec{v}$	error	rel err.	fit	error	rel err.	fit
0.5	0.007	0.014	0.88	0.078	0.156	0.90
1.0	0.005	0.005	0.97	0.021	0.021	0.92
1.5	0.052	0.035	0.95	0.084	0.056	0.92
2.0	0.006	0.003	0.97	0.136	0.067	0.92
2.5	0.033	0.013	0.95	0.131	0.052	0.93
3.0	0.108	0.036	0.96	0.106	0.035	0.95
3.5	0.105	0.030	0.96	0.068	0.019	0.94
4.0	0.085	0.021	0.95	0.096	0.024	0.92
4.5	0.092	0.020	0.95	0.076	0.017	0.92
5.0	0.095	0.021	0.95	0.083	0.016	0.92

Table 1: *Accuracy of the method*

Table 1 shows the results of the tests performed on a data set in which a single Gaussian moves at various speeds. The left side of the table tabulates the cases when no diffusion is involved; i.e. these cases simulate a moving rigid sphere. The first column gives the absolute velocity error (equation 5) in voxels. The second column gives the error relative to the speed. The third column gives the fit (equation 6). The right side of the table tabulates the cases when diffusion R_d of 1 is used.

The absolute error varies over the speed and diffusion parameters, however it remains well below one fifth of a voxel. The relative error is within 4 percent of the input velocity for the rigid motion of the Gaussian and 6 percent in the case when diffusion is involved. The case of $\vec{v} = 0.5$ with diffusion results in an significantly higher error. The most probable cause is the relatively large diffusion speeds outweigh the relatively low translation speed of the Gaussian.

The table shows that the fit is usually between 0.9 and 1.0. For the case in which the Gaussian moves at a speed of 0.5 with no diffusion a lower fit is found. In this particular case, the rigid Gaussian moves less than one voxel per time step and small errors introduced sampling do weigh strong on the result.

4.2. Chromatin decondensation

MCMR was used to analyze the movement of chromatin during formation of the cell nucleus of a newly formed cell. Chromatin was visualized in living cells and movement was followed using 3D confocal microscopy.

The data set consists of a series of 134 3D data sets. Each time step consists of a stack of 32 optical sections of 256 $\times$ 256 pixels. Due to physical characteristics of a confocal microscope the resolution along the z-axis is four times less than in the x-y plane. The 3D images are corrected for this by scaling in the z-direction during rendering.

MCMR was applied to the decondensation data set. The number of fitting iterations performed was 12 and the initial resolution of the velocity grid was $5 \times 5 \times 3$. The resolution of the final velocity grid is $129 \times 129 \times 25$. The used sampling density was $3 \times 3 \times 3 = 27$ samples per voxel. The average fit improvement was 0.85.

The average time to compute a time step of the vector field was approximately 1.5 hours on a single CPU desktop PC. The total time to compute the complete time dependent vector field was 8 days.

Figure 5 shows the decondensation process at various stages. The top row shows 4 time steps of the process using volume rendering on the original data. The bottom row shows the same time steps using particle paths extracted from the MCMR computed vector field. Particle paths are drawn as red/white line segments, with each line segment representing one time step in the data.

Figure 6 shows a combination of volume rendering and particle paths. Particle paths are traced backward in time starting from the 21 highest local maximum values in the final time step of the time series. Volume rendering shows the scalar field at the initial time step.

Figure 7 shows two stream tubes in the data. Each tube is constructed by advecting a ring of seed points in the vector field. Color represents time: red is used at the start of the advection and blue is used at the end. A checkerboard pattern was used to indicate the speed and divergence/convergence of the data. The length of a checkerboard cell represents the movement of data in the advected direction. The width of a checkerboard cell represents divergence. Checkerboard cells are initialized with equal widths.

Various relevant biological insights can be obtained from these images:

- The particle paths and stream tubes clearly show the decondensation process. The stream tubes show that different chromatin regions in the cell nucleus drift apart whilst keeping a more or less coherent shape. This insight seems support the theory that that the expansion of chromatin is linear and reorganization of the cell nucleus does not occur during this phase.

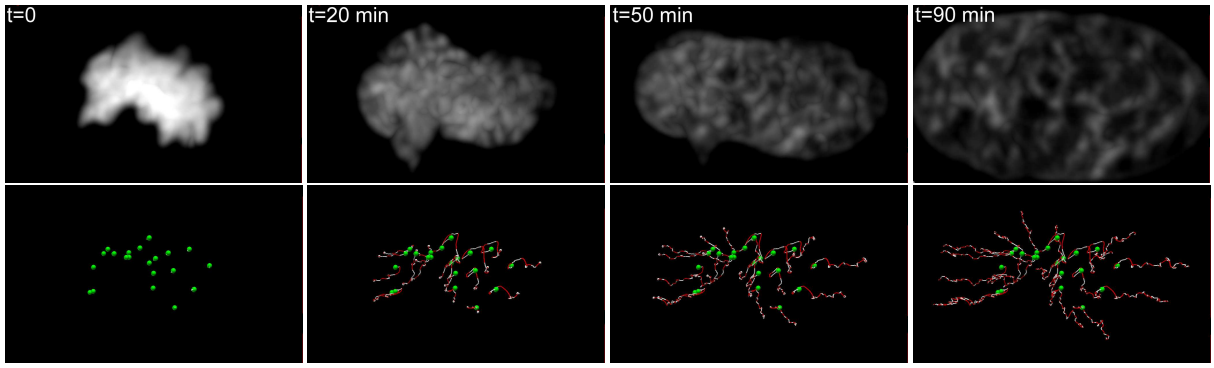

Figure 5: *The process of decondensation visualized using volume rendering (top) and particle paths (bottom). The red/white segments in the particle paths show the dynamics of the data movement. Seed points are drawn as small green spheres.*

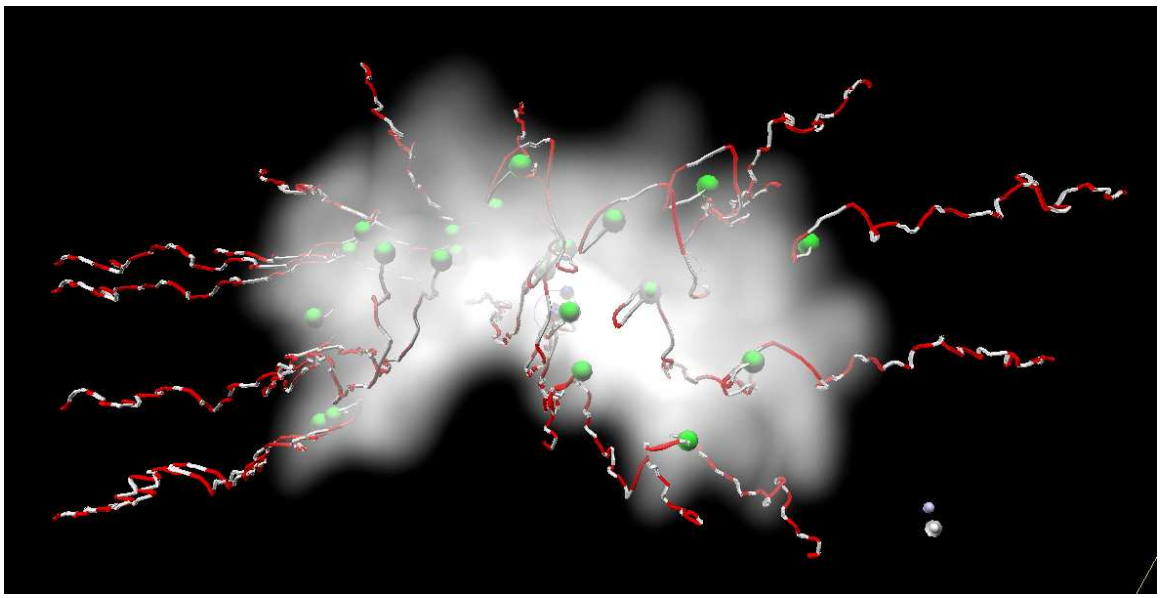

Figure 6: *The origin of dense chromatin regions: The data at the initial time step is shown using volume rendering. The 21 highest local maximum values in the final time of the data are chosen as seed points for the particle paths. Paths are traced backward in time. End points of the particle path are drawn as small green spheres.*

- Individual particle path segments and stream tube checkerboard cells show that chromatin movement is not uniform. The shape of checkerboard cells also clearly shows the variation in divergence of the flow.
- Interactive positioning of seed points of particle paths allow a user to trace the origin of dense chromatin regions. The end points of the particle path show the position of these dense areas in the condensed cell at the start of the decondensation. Also, the paths of these dense regions can be easily followed through time without the need of animation.

5. Discussion

In this paper, we have presented Mass Conservative Motion Reconstruction, a method for estimating motion in time dependent volume data. The resulting vector field can be used to trace the evolution of points and lines, which can be visualized as particle paths and stream surfaces.

The physically based principle of mass conservation in a continuum govern the implementation of MCMR. Given only a sequence of mass distributions, it is not possible to determine the velocity field underlying the data movement. However, the principle of mass conservation assures that the computed velocity field is a valid solution to this problem.

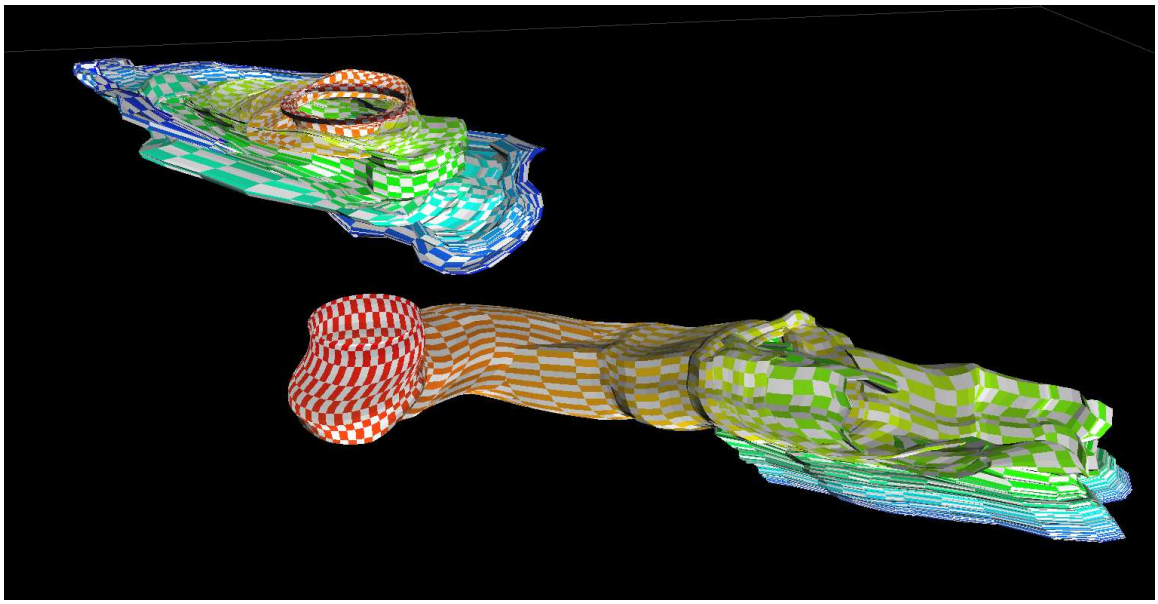

Figure 7: *The movement of two chromatin regions: stream tubes show movement in two different regions. The shape of checkerboard cells in each stream tube show the dynamics of the data movement. The shape of stream tube can be compared to get insight to the dynamics of two different chromatin regions.*

We now discuss a number of issues related to the MCMR method itself and how MCMR compares with other techniques that combine motion analysis and visualization:

- MCMR parameters
  The efficiency and accuracy of MCMR rely on three parameters:
  - sampling density
    The implementation describes a regular grid is used to sample M. Each sample in M is advected to a position in $M^\star$ and the data value at each sample is added to the voxel at the position of the advected sample in $M^\star$. If the sampling density is to low; i.e. the regular grid does not have a sufficient resolution, then aliasing of the data in $M^\star$ will occur. In the examples given in section 4 we used 3x3x3 = 27 samples per voxel in M. The sampling density is a trade-off between accuracy and computation time.
  - velocity grid
    The implementation uses grid refinement to compute the velocity field. Initially, a grid with a coarse resolution is used which is subsequently refined. The resolution of the finest grid relates to the maximum velocity gradient in the data.
    It would seem that a velocity grid with a high resolution will require more relaxation iterations, resulting in higher computation times. However, by restricting the evaluation of $M^\star$ to only a neighborhood which is influenced by the node, the evaluation time for $D()$ decreases as the resolution of G increases. Hence, the total computation time remains almost constant.
  - tension strength
    The tension strength used in the implementation is related to the mobility of the data. A low tension strength will allow data to move rapidly, while a high tension strength will result in low mobility.
- Performance
  MCMR is a time consuming method. Computing the vector field in the decondensation example took more than eight days. However, the process of acquiring data sets in confocal microscopy is also very time consuming and these data sets are not generated on a daily basis.
  MCMR is a pre-processing step and once the velocity field is computed, flow visualization techniques can be used interactively.
  The computation of velocity field time steps are independent and could be done in parallel. Parallelizing these computations is trivial.
- Usage
  MCMR can be applied to any time dependent volume data in which mass is conserved. MCMR is particularly useful in volumes where complex movements occur, such as deformations. Faster and more accurate methods are available for pure rigid body movements.
  A more elaborate transport equation must be used for those data sets in which mass is not conserved For example, the transport equation can be extended with complex evolution models in which chemical reactions influence

mass equilibrium in the data. However, such extensions will require domain knowledge.

- Feature tracking
  MCMR can be compared with feature tracking as a method for the analysis of motion in time dependent volume data. The essential differences between these methods is that MCMR provides movement information of every voxel in the data set, whereas feature tracking provides information of individual features. In addition, MCMR does not require domain knowledge to compute the vector field, whereas most physically based feature tracking algorithms require domain knowledge to be effective.
  On the other hand, feature tracking algorithms provide an explicit set of events and features that can be useful to reason about properties of the data. MCMR does not provide this information and additional methods will be needed to extract this information from the data or velocity field. For example, in order visualize meaningful particle paths methods will be needed to generate their seed points.

6. Conclusions

MCMR is a new method for estimating motion in time dependent volume data. The physically based principle of mass conservation in a continuum assure that the MCMR velocity field represents the movement of mass.

Standard flow visualization techniques, such as particle paths and stream surfaces, can be used to show the dynamics of the data movement. The vector field can be used as a basis for to automated detection and quantification methods; e.g. feature tracking, flux computations, bifurcations, etc.

MCMR is particularly useful in volumes where complex movements occur, such as reorganization or mixing, and where no a-priori knowledge about movement is available. We have shown that MCMR is very accurate by comparing the generated vector field with a known vector field in an synthetic example. MCMR was instrumental in the formulation of many conjectures about chromatin decondensation in living cells.

References

1. J.J. Gibson. *The Perception of the Visual World*. Riverside Press, Cambridge, 1950.
2. Berthold K. P. Horn and Brian G. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.
3. Hans-Hellmut Nagel. On the estimation of optical flow: Relations between different approaches and some new results. *Artificial Intelligence*, 33:299–324, 1987.
4. Benoit M. Dawant. Non-rigid registration of medical images: Purpose and methods, a short survey. In *Proceedings IEEE International Symposium on Biomedical Imaging*, pages 465–468, 2002.
5. J.C. Gee, D.R. Haynor, and R. Bajcsy. Finite element approach to warping of brain images. In M.H. Loew, editor, *Medical Imaging 1994: Image Processing*, Bellingham, 1994. SPIE.
6. Ch. Davatzikos. Spatial transformation and registration of brain images using elastically deformable models. *Computer Vision and Image Understanding*, 66(2):207–222, 1997.
7. G.E. Christensen, R.D. Rabbitt, and M.I. Miller. Deformable templates using large deformation kinematics. *IEEE Transactions on Image Processing*, 5:1435–1447, oct 1996.
8. W.C. de Leeuw and R. van Liere. BM3D: Motion estimation in time dependent volume data. In R. Moorehead, M. Gross, and K. Joy, editors, *Proceedings IEEE Visualization 2002*, pages 427–434, Los Alamitos (CA), 2002. IEEE Computer Society Press.
9. W. Press, B. Flannery, S. Teukolskt, and W. Vettering. *Numerical Recipes in C*. Cambridge University Press, Cambridge, 1988.