BM3D: Motion Estimation in Time Dependent Volume Data

Wim de Leeuw* Robert van Liere *

*Center for Mathematics and Computer Science, CWI, Amsterdam, Netherlands.

Abstract

This paper describes BM3D: a method for the analysis of motion in time dependent volume data. From the sequence of volume data sets a sequence of vector data sets representing the movement of the data is computed. A block matching technique is used for the reconstruction of data movement. The derived vector field can be used for the visualization of time dependent volume data. The method is illustrated in two applications.

CR Categories and Subject Descriptors: I.3.3 [Computer Graphics]: Picture/Image Generation; I.3.6 [Computer Graphics]: Methodology and Techniques

Keywords: feature tracking, vector fields, volume visualization, biomedical imaging.

1 Introduction

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

We present BM3D: a general method for motion estimation in time dependent volume data. Our objective is to estimate a velocity for all positions in a volume. We assume that motion can be reconstructed from pattern changes local to these positions. BM3D can be applied to any time dependent volume data. It is useful in volumes where complex movements occur, such as reorganization or mixing, and where no knowledge about the movement is available from other sources.

The output of BM3D is a time dependent vector field. Standard vector field visualization techniques can be used to visualize the vector field. Stream lines or texture techniques can be used to show velocity at a particular moment. Particle paths can be used to show movement paths of objects of interest. In addition, the resulting vector field can serve as a basis for data analysis. The tracking of a feature in the original volume data is reduced to the computation of a particle path in the generated vector field. Also, material flux through an arbitrary surface can be easily determined.

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 of movement of brightness patterns in an image. It gives information about the spatial arrangement of the objects in the image and the rate of change of this arrangement. The computation of an optical flow field from digital image sequences was addressed by Horn et al. [2] and Nagel [3]. These algorithms use intensity gradients in the spatial and temporal domainin 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. However other criteria depending on the application can be used.

Another application area is video processing.. For example, in MPEG encoding motion is estimated using prediction vectors [4]. The primary goals in video processing are compression and processing tasks such as de-interlacing and frame rate interpolation.

A popular technique for implementing video transmission is the block-matching (BM) algorithm [5]. The underlying idea of the BM-algorithm is that each image frame is divided into a fixed number of square blocks of pixels. For each block, a search is made in a reference frame for a matching block. The search is for the 'best' matching block. Best can be defined as minimizing either mean square differences or mean absolute difference of pixels in the matching blocks (see figure 1). Typical block sizes are in the order of 16x16 pixels, and the maximum displacement might be 64 pixels from a block's original position. Several search strategies are possible, ranging from exhaustive search to various sampling mechanisms, [6].

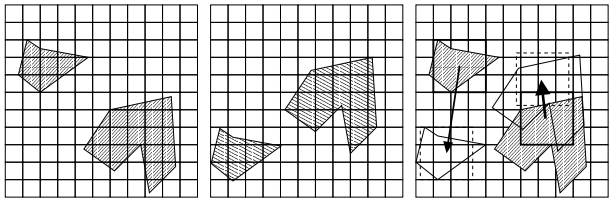

Figure 1: Block matching algorithm: movement is computed by determining the best match using a block of pixels. From left to right: data at time step t, data at time step $t+1$, best matching blocks (stippled boxs) and displacement vectors (arrows) for two blocks in the data.

A well known visualization method for gaining insight into time dependent data is feature tracking, [7, 8]. These algorithms consider feature tracking as a two step process. First, in the detection phase, features extracted from each time step in the data. Features are described by feature attributes. Then, in the tracking phase, the correspondence of features in successive time steps is used for the determination of tracks. Correspondence between features is determined by thresholds on the feature attributes. Often the notion of evolutionary events, such as split, merge, birth and death events are used to describe the evolution of features. Additional correspondence criteria may be formulated in case of evolutionary events.

Our work differs from the previous work on block matching in two ways. First, the two dimensional BM algorithm is generalized

*CWI, Department of Information Systems, P.O. Box 94097, 1090 GB Amsterdam, Netherlands. E-mail wimc@cwi.nl

to three dimensions. Instead of pixels voxels are used, instead of a 2D search window a 3D space is searched, a 3D matching block of voxels is used, etc. Second, in contrast to video transmission where the goal is image compression, our goal is to accurately estimate motion in the volume data sets. The goal of video transmission algorithms is to search for (large) block displacements in a sequence of images. In contrast, BM3D computes velocities at locations.

Compared to feature tracking, the BM3D generated vector field does not rely on the definition of a feature and feature correspondences. Instead, motion vectors are computed by matching blocks of data. A matching operator is used to compute correspondences in the raw data.

3 BM3D Method

The objective is to compute a motion vector $\vec{v}$ at a position $\vec{x}$ for a particular time step T. This is achieved by comparing a data block at position $\vec{x}$ at time step T with a number of displaced data blocks in the vicinity of $\vec{x}$ at time step $T+1$. The motion vector $\vec{v}$ will be chosen to be the displacement vector of the 'best matching' data block.

Stated more formally, the objective is to find a the motion vector $\vec{v}$ which displaces the data at position $\vec{x}$ at time T to a position $\vec{x}+\vec{d}$ at time $T+1$. The displacement $\vec{d}$ is then the sought motion vector. A minimization procedure is used to computed displacement $\vec{d}$.

We first define $\ominus$ as being a data block matching operator $\ominus : R^3 \times R^3 \rightarrow R$. The block matching operator is a metric which is used to compare two data blocks based on voxel values in the block.. Two blocks are defined to be match if $\ominus$ results in a low value and are defined not to match if $\ominus$ returns a high value. We denote $\ominus$ as:

$$B_T(\vec{x}) \ \ominus \ B_{T+1}(\vec{y}) \tag{1}$$

in which $B_T(\vec{x})$ is a data block in the data set at time T centered around position $\vec{x}$, $B_{T+1}(\vec{y})$ is a data block in the data set at time $T+1$ centered around position $\vec{y}$. Both blocks have the same size.

Finding the displacement $\vec{d}$ is realized by computing the minimum of the set of block matches in the vicinity of $\vec{x}$; i.e. the displacement $\vec{d}$ for the smallest element of the set

$$\{B_T(\vec{x}) \ \ominus \ B_{T+1}(\vec{x}+\vec{d}) \mid \vec{d} \in SW\} \tag{2}$$

in which the displacement vectors $\vec{d}$ are chosen from a search window SW position in the vicinity of $\vec{x}$.

This process is repeated for all positions $\vec{x}$ in the volume to construct a vector field for one time step. To construct a time dependent vector field, the process is repeated for all time steps.

We illustrate the method in 2D (see figure 2). A data block B around position $\vec{x}$ is matched with 4 data blocks displaced at different positions at time $T+1$. Each block is highlighted with a unique color. For clarity, only four blocks are drawn; however, all blocks in the search window will be tested. The result of the matching operator is shown in the lower part of the figure. In this example we choose matching operator to be the sum of the absolute differences the voxel values in the block. The right most block has the minimum value.

The pseudo C-code for the block matching algorithm is as follows:

```
void BM3D (timestep T)
{
   ∀ voxel x ∈ dataset (T))
   {
      ∀ d⃗ ∈ SEARCH_WINDOW and GRANULARITY
      {
         value = B_T(x) ⊖ B_{T+1}(x + d))
         if (value < minval)
         {
            minval = value;
            d⃗_min = d⃗;
         }
      }
      motion vector v⃗ = d⃗_min;
   }
}
```

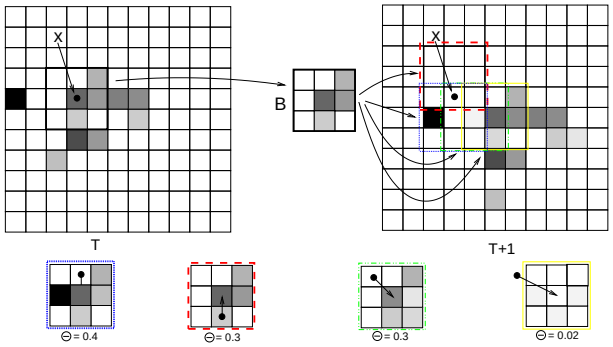

Figure 2: BM3D illustrated in 2D : a block B positioned at $\vec{x}$ at time T is matched with 4 data blocks displaced at different positions at time $T+1$. The result of the matching operator for each displaced block is shown in the lower part of the figure.

The BM3D algorithm sweeps over all voxels $\vec{x}$ and applies the block matching operator to blocks positioned at $\vec{x}$ with blocks positioned a $\vec{x}+\vec{d}$ in which $\vec{d}$ is chosen from a search window. The method of choosing the displacement vector $\vec{d}$ from a search window depends on the search strategy and its granularity.

A number of parameters occur in the algorithm. Here, the influence of parameters on the algorithm and the result is discussed:

- matching operator $\ominus$

  The matching operator usually found in 2D block matching algorithms is the sum of the absolute pixel differences the data blocks.

  In the current BM3D implementation, the absolute voxel difference scaled with a Gaussian weighting function centered at the point of interest is used; i.e.

  $$|B_T(\vec{x}) - B_{T+1}(\vec{x}+\vec{d})| e^{-C|\vec{x}|} \tag{3}$$

  The motivation is that, in contrast to the 2D algorithms used for video compression, BM3D is interested in the motion at a particular position. The data variation close to this position is more important than variations further away. Therefore, a Gaussian weighting function centered at the position of interest is used to scale the voxel differences.

- size of the matching block.

  The block matching operator compares all voxel values of a data block at a particular point with voxels of a data block at a displaced point. The larger the size of the block, the larger the region is used to determine the motion.

  The block size has an optimum which depends on the characteristics of the data. One one hand, matching single voxels will allow the same or almost the same voxel value to be

matched. On the other hand, if the block size is too large the average movement of a region is calculated and not the movement at a particular position. The optimum block size will be the one that matches the voxel patterns in the neighborhood of the position of the block.

In our implementation, the scale factor (C in equation 3) used in the matching operator is proportional to the block size.

- search window

  The search window is the area which is searched for the displacement vector. The size of the search window should be at least as large as the largest data displacement between two time steps. For example, if data represents fast moving media a large search window must be chosen.

- displacement vector granularity

  Until now, we have assumed that the size of the displacement vector is a multiple of voxels in the volume. However, data movement is not restricted to only complete voxels. By using tri-linear interpolation, data blocks can be compared at sub-voxel resolution. Hence, the block matching operator can be computed for arbitrary sized displacement vectors.

  Granularity is a metric that is used to restrict the number of displacement vectors in the search window. It defines the potential displacement of a vector with respect to the size of a voxel. For example, a granularity of 1.0 relates to displacement vector measured in entire voxels. A granularity of 0.5 relates to displacements at half the size of a voxel.

  Figure 3 illustrates the notion of displacement vector granularity. In the left image, the displacement vector granularity is equal to the size of a voxel. In the right image, the granularity is equal to half the size of a voxel. Hence, the displacement vector granularity is 0.5.

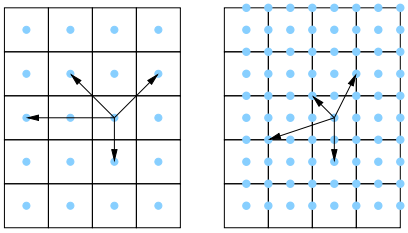

Figure 3: Displacement vector granularity : The granularity in the left image is 1.0. The granularity in the right image is 0.5. Dots show positions of blocks, arrows show some possible displacements.

3.1 Search strategies

The performance of the BM3D algorithm is dependent on the number of block matches that are performed. In general, a search of the complete data set must be performed in order to determine the displacement vector for the block matching operator. This results in $N^3 - 1$ block matches in which N is the resolution of the data set. A standard method of decreasing N is by introducing a search window positioned around the point of interest. In practice, the size of the search window is chosen to be smaller than N.

The current BM3D implementation uses two different search strategies to determine the displacement vector in the search window. The first strategy is a straightforward exhaustive search. Blocks are centered around a displacement vector in the search window.

The second strategy uses a two level steepest ascent method to guide the search along a path to the best block match. Instead of the exhaustive search of all the displacement vectors in the search window, only the 26 blocks adjacent to the current position are examined. The block with the minimum block matching value found is taken as the new current position. The process is repeated until no lower value is found among adjacent blocks. The first pass of the steepest ascent uses displacement vector granularity of 1.0. This is followed by a second pass in which the displacement vector granularity is set to the granularity.

3.2 Test Results

Artificial data sets were generated to study the accuracy and performance of BM3D. The movement of the data in these data sets simulate the movement of a paramterized Gaussian function G; i.e. a blob with an initial position, velocity and diffusion rate. The parameters of the Gaussian are: initial position $\mathbf{p}_0$, initial size σ_0, velocity $\mathbf{v}$, intensity I, and diffusion rate 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 blob size such that the integral over all intensity values remain constant. The following equation was used for the Gaussian function :

$$G(\mathbf{x}) = I\left(\frac{\sigma_0}{\sigma_0 + Dt}\right)^3 e^{-\left(\frac{|\mathbf{p}_0 + \mathbf{v}t - \mathbf{x}|}{\sigma_0 + Dt}\right)^2} \quad (4)$$

Each generated test data set has a resolution of 64x64x64. 32 time steps were computed.

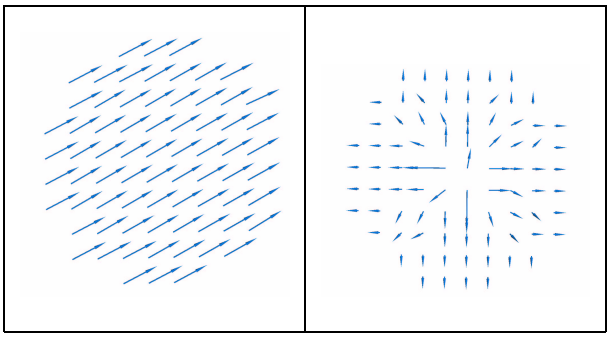

Figure 4: Vector plots of one time step from the moving Gaussian test data sets. Image on the left represents the Gaussian with initial translation speed and a diffusion rate of 0.0. Image on the right represents the Gaussian with initial speed of $\mathbf{0}$ and a diffusion rate of 0.05.

A number of tests with varying input data and parameter settings were performed to gain insight in the accuracy and performance of the algorithm. Figure 4 shows two vector plots of a 2D slice of the BM3D generated vector fields. In the left image, the initial parameters of $G(\mathbf{x})$ were set to $\mathbf{v} = < 1.8,\ 1.0,\ 0.0 >$ and $D = 0.0$. In this case, the Gaussian moves at a uniform speed in the direction of the initial velocity. In the right image the initial parameters of $G(\mathbf{x})$ were set to $\mathbf{v} = < 0.0,\ 0.0,\ 0.0 >$ and $D = 0.05$. In this case, the motion is completely determined by the change of shape of the Gaussian. The radius increases over time based on diffusion rate.

In the case that the diffusion rate is equal to 0, the input velocity can be compared to the BM3D computed velocity in the vector

field. The error is taken to be the average length of the difference between the input velocity and BM3D vector at every voxel;

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

in which $\mathbf{v}(i)$ is the input velocity at voxel i and $\hat{\mathbf{v}}(i)$ is the speed BM3D computed speed at voxel i. Table 1 tabulates the error for three different block sizes, two search strategies and three displacement vector granularities, for the test data set in which $\mathbf{v} = < 1.8, 1.0, 0.0 >$ and $D = 0.0$. From the data it can be seen that decreasing the displacement vector granularity also decreases the average error. Also, increasing the block matching size decrease the average error. Finally, the error caused by using hill climbing search is negligible compared to the error made by the exhaustive search strategy.

granularity = 0.05	3x3x3	5x5x5	9x9x9
exhaustive	0.055	0.016	0.0013
hill climbing	0.061	0.016	0.0013
granularity = 0.3	3x3x3	5x5x5	9x9x9
exhaustive	0.11	0.11	0.10
hill climbing	0.12	0.11	0.10
granularity = 1.0	3x3x3	5x5x5	9x9x9
exhaustive	0.20	0.21	0.20
hill climbing	0.20	0.21	0.20

Table 1: Average error of varying block sizes, search strategies, and displacement vector granularities.

Table 2 shows the performance of the BM3D method for three different block sizes and two search strategies. The performance is measured as the time spent (in milliseconds) to compute the motion of the data at each voxel. The total time of BM3D can be computed by multiplying this value with the number of voxels and the number of time steps. The table shows that for both strategies the time needed is linear over the number of voxels in the matching block. In addition, using the hill climbing search strategy is approximately a factor 100 faster than the exhaustive search strategy.

	3x3x3	5x5x5	9x9x9
exhaustive search	58.1	222.1	796.3
hill climbing search	0.56	2.3	9.0

Table 2: Performance (in milliseconds per voxel) with varying three block matching sizes and two search strategies.

4 Applications

4.1 Convection

A time dependent data set of convection flow – the circulatory motion that occurs in a fluid at a nonuniform temperature owing to the variation of its density and the action of gravity – in the earth's crust was obtained from a numerical simulation [1] done by Dr. Jörg Schmalzl, Institute of Geophysics, Münster University. The data set consists of 30 time steps at a resolution of 128x128x64. Figure 5 show four volume renderings of the data set. Regions of high temperature move upwards from the bottom to the top, while regions of low temperature move downwards from the top to bottom.

[1] The data was kindly provided by Klaus Engel, University of Stuttgart

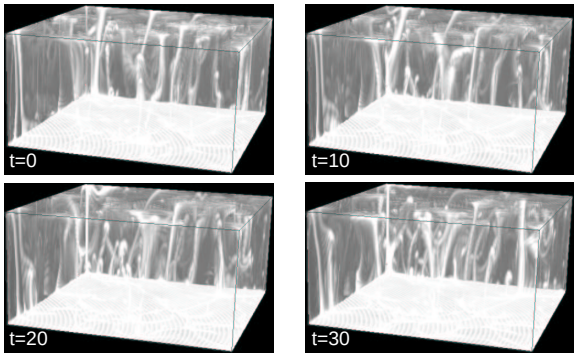

Figure 5: Four time steps of the convection data set. Volume rendering is used to show the scalar field.

BM3D was applied to the convection data set. The hill climbing search strategy with a block matching size of 5x5x5 and a displacement vector granularity of 0.05 was used. Equation 3 was used for the block matching operator. A wide palette of vector visualization techniques have been implemented to visualize the BM3D generated vector field.

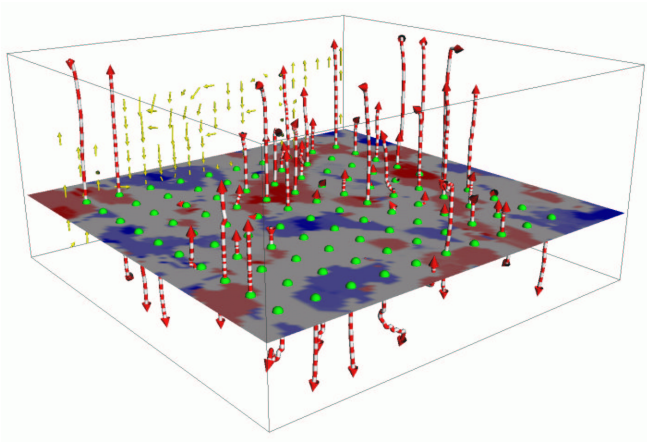

Figure 6: Particle paths, arrow plots, and vector magnitude are used to depict various aspects of the vector field.

Figure 6 illustrates three techniques. A slice of the vector field is taken at $Z = 32$ at time $T = 0$. Red, blue and gray regions are drawn on a plane to represent the magnitude of the motion perpendicular to the plane. Red denotes upward flow, blue denotes downward flow, gray denotes limited flow.

An arrow plot (drawn as small yellow arrows) is drawn from a data slice taken at $X = 16$ at time $T = 0$. Arrows represent the flow magnitude and direction at each voxel in the slice.

Particle paths are shown as red/white segmented tubes taken at a number seed points (shown as small blue spheres). Small arrows are drawn at end point of particle path to indicate direction. Each segment of the particle corresponds with two time steps in the data. Seed points are chosen on a grid-like pattern on the cutting plane. Particle paths represent the motion of the seed point throughout the time steps. Note that the lengths of the particle paths and the path segments vary.

4.2 Chromatin decondensation

We have used BM3D to analyze the movement of chromatin during formation of the cell nucleus of a newly formed daughter cell. Chromatin was visualized in living cells and movement was followed using 3D confocal microscopy. Densely packed areas of chromatin are used to analyse the movement of the entire chromatin. The densely packed areas are represented as high intensity levels in the data.

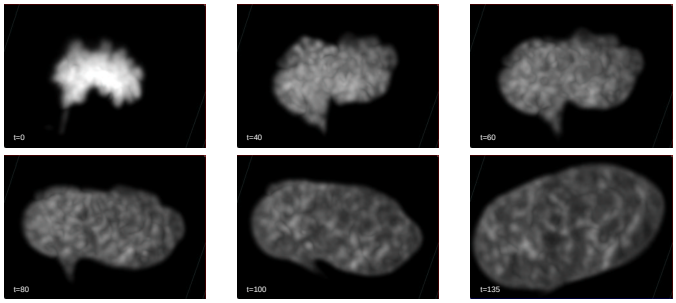

Figure 7: Six volume renderings show movement of compact chromatin domains as concentration variations over time.

The data set consisted of a series of 134 3D data sets. Each time step consists of a stack of 64 optical sections of 256 $\times$ 256 pixels. Due to physical characteristics of a confocal microscope the optical 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.

Figure 7 show six volume renderings of the data set. Densely packed areas of chromatin are represented as high intensity values in the data.

BM3D was applied to the decondensation data set. The hill climbing search strategy, a block matching size of 5x5x5, a displacement vector granularity of 0.05, and equation 3 were used as parameters. Figure 8 shows how the original data can be combined with the BM3D generated vector field. [2] Volume renderings of the original data are combined with particle paths. Seed points are chosen by using a feature detector to find points of high intensities at time $T = 0$. These points were used as the seed points for the particle paths. The complex movement of chromatin can be studied by following particle paths.

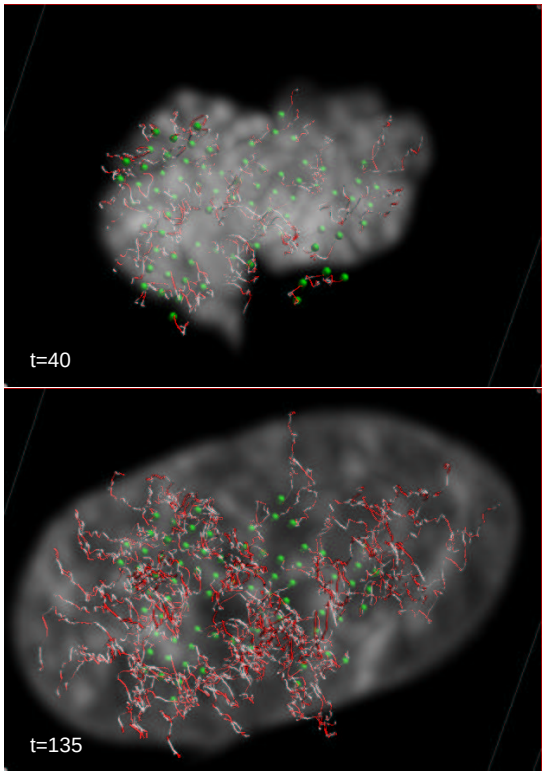

Figure 8: Combining volume data with particle paths.

5 Discussion

BM3D is a technique used to generate a time dependent vector field that represents the movement of data in time dependent volume data sets. Insight into these movements can be obtained by applying standard vector field visualization techniques to the vector field.

Various parameters are used to drive BM3D. For a simple artificial data set we have shown that the errors made by BM3D depend on the value of the displacement vector granularity, block matching size, and search strategy. In real data, however, parameter settings depend on the characteristics of the data:

- The size of the matching block should be chosen to reflect the variation of velocity changes in the data. In data with large velocity fluctuations over space, a small block size should be chosen. In data in which the velocity is more uniform, larger blocks can be applied.
- The size of the search window should be chosen to reflect the maximum velocity occurring in the data; i.e. the maximum displacement of the data remains within the search window.
- The granularity of the displacement vector is a trade-off between performance and the error. A fine granularity results in a more accurate reconstruction, but will require more computation.
- The matching operator $\ominus$ should take special properties of the data into account. Equation 3 implements a weighted sum of absolute differences of the voxels in the data blocks. In cases in which mass conservation holds, the total mass of the compared blocks can be used as a metric.
- As shown in section 3 the additional error due to using hill climbing search is negligible for uniform data. However, data sets with high frequencies may result that the hill climbing algorithm will iterate to a local minimum. In such data sets an exhaustive search may lead to better results.

Since there usually is no a priori knowledge about the characteristics of the data which influence the parameter settings, test runs are usually required. Several approaches can be utilized to gain insight in to quality of the vector field: First, visual comparison of animations of the original data with paths generated from the vector data. Second, an analysis of the matching errors. High error values indicate problems in matching. The lower the value the better the match. Third, using known properties of the data to validate the field. For example, if some data the total flux through some surface is known, this value can be compared to the flux calculated from the generated vector field.

The BM3D generated vector field need not only be used for the visualization of data movement. The vector field can also be used

[2] See also the supplementary MPEG fig8.mpg for an animation.

for data analysis. In particular, we give examples of the vector field can be used for feature tracking and the modeling of physical phenomena.

In [9], we have presented an interactive feature tracking system and its use for the analysis of chromatin decondensation. In this method, features are described as points in a multidimensional attribute space and distances between points are used as a measure for feature correspondence. Tracks are constructed by linking corresponding features. We have also applied BM3D to the same data set and compared the results with our feature tracking method (see Section 4.2 for a description of the data). Figure 9 shows 8 green tracks (obtained from our feature tracking method) combined with 8 red particle paths. At time $T = 50$ 8 features (local maximum intensity values in the data) belonging to the longest found tracks were selected. The positions of the selected features were used as seed points for particle paths traced forward and backward in time using the vector data. A volume rendering of the chromatin data at time $T = 50$ is superimposed over the paths.

It would be expected that the computed points in the feature tracks and particle paths would be roughly the same. Although they obviously are not, figure 9 shows that the shape and direction of the tracks and paths are very similar. [3] There are three possible explanations for these differences. First, the feature tracks are not all the same length. Some tracks terminate because a corresponding feature could not be found in a following time-step. Second, incorrect correspondences in the feature matching algorithm may occur. This problem is inherent in nature of the multidimensional space of our feature tracking system. Finally, integration errors may accumulate, resulting in incorrect particle path computation.

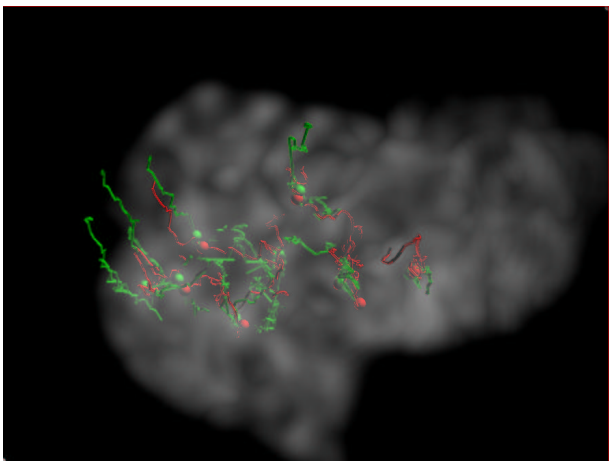

Figure 9: Feature tracks (green) and particle paths (red).

It can be argued that BM3D has a number of advantages over the feature tracking algorithm. First, BM3D does not rely on the definition of a feature. Instead, movement is computed using the underlying data. Second, by placing a seed point and computing a particle path, movement can be examined from any position in the data. In contrast, feature tracks are only available from positions where a feature is detected.

On the other hand, feature tracking provides an explicit representation of properties in the data which are of interest to the user. In addition, BM3D results in a large increase of stored data whereas feature tracking is very useful to compress large volumetric data sets.

[3] See also the supplementary MPEG fig9.mpg for an animation.

The vector field generated by BM3D can also be used for modeling of physical phenomena. In the case of the decondensation data set, densely packed areas of chromatin are represented as high intensity levels in the data. However, from the vector field we also have velocity information and can easily compute acceleration and forces. This information can be used to model various aspects of the decondensation process. For example, it may be possible to determine the individual chromosomes in the chromatin this way.

6 Conclusions

We have presented BM3D, a method to estimate motion in a volume. The assumption is that motion can be reconstructed from local pattern changes in the data. The output of BM3D is a time dependent vector field. Vector field visualization techniques can be applied to the vector field.

BM3D can be applied to any time dependent volume data. It is useful in volumes where complex movements occur, such as reorganization or mixing, and where no knowledge about the movement is available from other sources.

Future work will include further improvements to the velocity estimation by enhancing the matching process. In addition to block matching, more elaborate methods for 2D motion estimation have been proposed such as triangle motion compensation, and control grid interpolation. Such techniques should be applied to volume data as well. Also, the derived velocity information is useful for volume data compression purposes.

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] P. Anandan. A computational framework and an algorithm for the measurement of visual motion. *International Journal of Computer Vision*, 2:283–312, 1989.

[5] A. M. Tekalp. *Digital Video Processing*. Prentice-Hall, Upper Saddle River, NJ, 1995.

[6] G. Gupta and C. Chakrabarti. Architectures for hierarchical and other block matching algorithms. *IEEE Transactions for Circuits and Systems for Video Technology*, 5:477–489, 1995.

[7] R. Samtaney, D. Silver, N. Zabusky, and J. Cao. Visualizing features and tracking their evolution. *IEEE Computer*, 27(7):20–27, 1994.

[8] F. Reinders, F. Post, and H. Spoelder. Attribute-based feature tracking. In E. Groller, H. Loffelmann, and W. Ribarsky, editors, *Data Visualization '99*, pages 63–72. Springer-Verlag, 1999.

[9] W.C. de Leeuw and R. van Liere. Chromatin decondensation: a case study of tracking features in confocal data. In Kenneth Joy, Amitabh Varshney, and Thomas Ertl, editors, *Proceedings IEEE Visualization 2001*, pages 441–444, Los Alamitos (CA), 2001. IEEE Computer Society Press.