FRIEDRICH-ALEXANDER-UNIVERSITÄT ERLANGEN-NÜRNBERG
INSTITUT FÜR INFORMATIK (MATHEMATISCHE MASCHINEN UND DATENVERARBEITUNG)

Lehrstuhl für Informatik 10 (Systemsimulation)

PDE based Video Compression in Real Time

H. Köstler, M. Stürmer, Ch. Freundl, and U. Rüde

Lehrstuhlbericht 07-11

PDE based Video Compression in Real Time

H. Köstler, M. Stürmer, Ch. Freundl, and U. Rüde

Abstract

We implement a video codec with a decompression scheme based on image inpainting using different kinds of diffusion models. The arising PDEs are solved by an efficient multigrid solver in case of linear homogeneous diffusion and by a multilevel scheme applying inexact lagged diffusivity in case of nonlinear isotropic and anisotropic diffusion. We consider real world and medical data sets, and show that it is possible to decompress more than 25 frames per second (fps) of size 320×240 on a PlayStation™ 3.

1 Introduction

Video coding can be done by a variety of methods, e.g., see [1, 2] for an introduction and survey of the field. Based on the ideas presented in [3], we develop a method for video coding that we call *PDEVC (PDE based video compression)*.

In order to encode an uncompressed video sequence we choose an adapted sub-selection of all points in the domain based on a recursive subdivision of each frame [4] and save their color value and coordinates. The information of all other points is discarded.

A variational approach for image inpainting is used for video decompression. The basic idea is to fix the given points as hard constraints or landmarks in the video sequence and to apply different kinds of diffusion based regularizers to fill in the missing information at unknown points.

In Section 2 we describe the algorithm to select the landmarks and explain the compressed file format. After that we introduce the variational image inpainting model in Section 3. We discretize the arising PDEs using finite differences or finite volumes and solve them by a full multigrid solver in the linear case. In the nonlinear case a multilevel Gauss-Seidel solver using an inexact lagged diffusivity method is applied. We present in Section 4 results for inpainting single frames of MRI images using different regularizers to show the quality of the models, as well as results for video sequences to show that real time performance is possible on novel architectures such as a PlayStation™ 3.

2 Video Compression

In order to compress a video file, some points of each frame are selected as landmarks such that they are sufficient for a good reconstruction of the original video sequence. The simplest method is to choose the landmarks randomly. This will be used for comparison to a more evolved method called *B-Tree Triangular Coding (BTTC)* [4].

For BTTC the image is split up into two main triangles, defined by its diagonal, i.e., from top left to the lower right corner of the image domain. The goal is to subdivide these triangles in a way that one obtains a set of triangles where the differences of the color values in each of them are below a certain threshold.

For a particular triangle with corners (x_1, y_1), (x_2, y_2), and (x_3, y_3), the plane

$$G(x, y) = c_1 + \alpha(c_2 - c_1) + \beta(c_3 - c_1) \tag{1}$$

defined by linear interpolation between the three color values c_1, c_2, c_3 at the corners with

$$\begin{aligned} \alpha &= Z^{-1}\left((x-x_1)(y_3-y_1)\right. && \text{(2a)}\\ &- \left.(y-y_1)(x_3-x_1)\right) \\ \beta &= Z^{-1}\left((x_2-x_1)(y-y_1)\right. && \text{(2b)}\\ &- \left.(y_2-y_1)(x-x_1)\right) \\ Z &= (x_2-x_1)(y_3-y_1) && \text{(2c)}\\ &- (y_2-y_1)(x_3-x_1) \end{aligned}$$

is calculated. In order to compress RGB color images, the interpolation is done for the three color channels independently. If the surface given by the color values within the triangle differs more than a given tolerance from the plane $G(x, y)$ in one or more color channels, the perpendicular is dropped on the hypotenuse of the current triangle which splits it up into two. The subsequent subdivision of the triangles creates a tree structure, with the leafs representing triangles which need not to be subdivided, see Fig. 1. This tree can also be written as a bitlist where an inner node is represented by a 1 and a leaf by a 0.

We only have to store in a list the color values at grid points where the perpendicular meets the hypotenuse when subdividing a triangle. The order of these values is given by a breadth first search through the tree.

Now there are two lists which hold the information required for decompression: the bitlist from which the coordinates can be restored, and the list of color values. By compressing each list using a Huffman encoding [5] the amount of memory for storing the bitlist can be reduced even more.

More details on the implementation of the video compression scheme and some results on achievable compression rates showing that PDEVC is in the range of common coding schemes like MPEG-1 can be found in [6].

3 Video Decompression by Image Inpainting

In order to decompress the video data we read in the compressed video file, restore the color values at the landmarks, and then reconstruct the missing color values in the video sequence by image inpainting. Image inpainting is often used to restore image information, if parts of the image data are missing, e.g., were destroyed or some objects are occluded by others [7, 8, 9].

3.1 Model

Next, we describe a variational approach for image inpainting. We assume that in the input image $u^0 : \Omega \to \mathbb{R}$ a finite number $m \in \mathbb{N}$ of landmarks in a subset $\Omega_C \subset \Omega$ of the image domain $\Omega \subset \mathbb{R}^2$ is prescribed. For one frame $u : \Omega \to \mathbb{R}$ we have

$$u^0(x^i) = u(x^i), \quad \forall x^i \in \Omega_C,\, 1 \le i \le m. \tag{3}$$

We also assume that the color values of the landmarks in u^0 are correct, i.e., there is no noise in the input data. If this is not true, one may combine the image inpainting with image denoising [10, 11] and not fix the solution at the given landmarks, but treat them as soft constraints penalizing deviations.

We use a variational approach that requires to solve the (nonlinear) PDE

$$\begin{aligned} -\operatorname{div}(D\nabla u) &= 0 && \text{if } \boldsymbol{x} \in \Omega \setminus \Omega_C \\ u &= u^0 && \text{if } \boldsymbol{x} \in \Omega_C \\ \langle D\nabla u, \boldsymbol{n}\rangle &= 0 && \text{if } \boldsymbol{x} \in \partial\Omega \end{aligned} \tag{4}$$

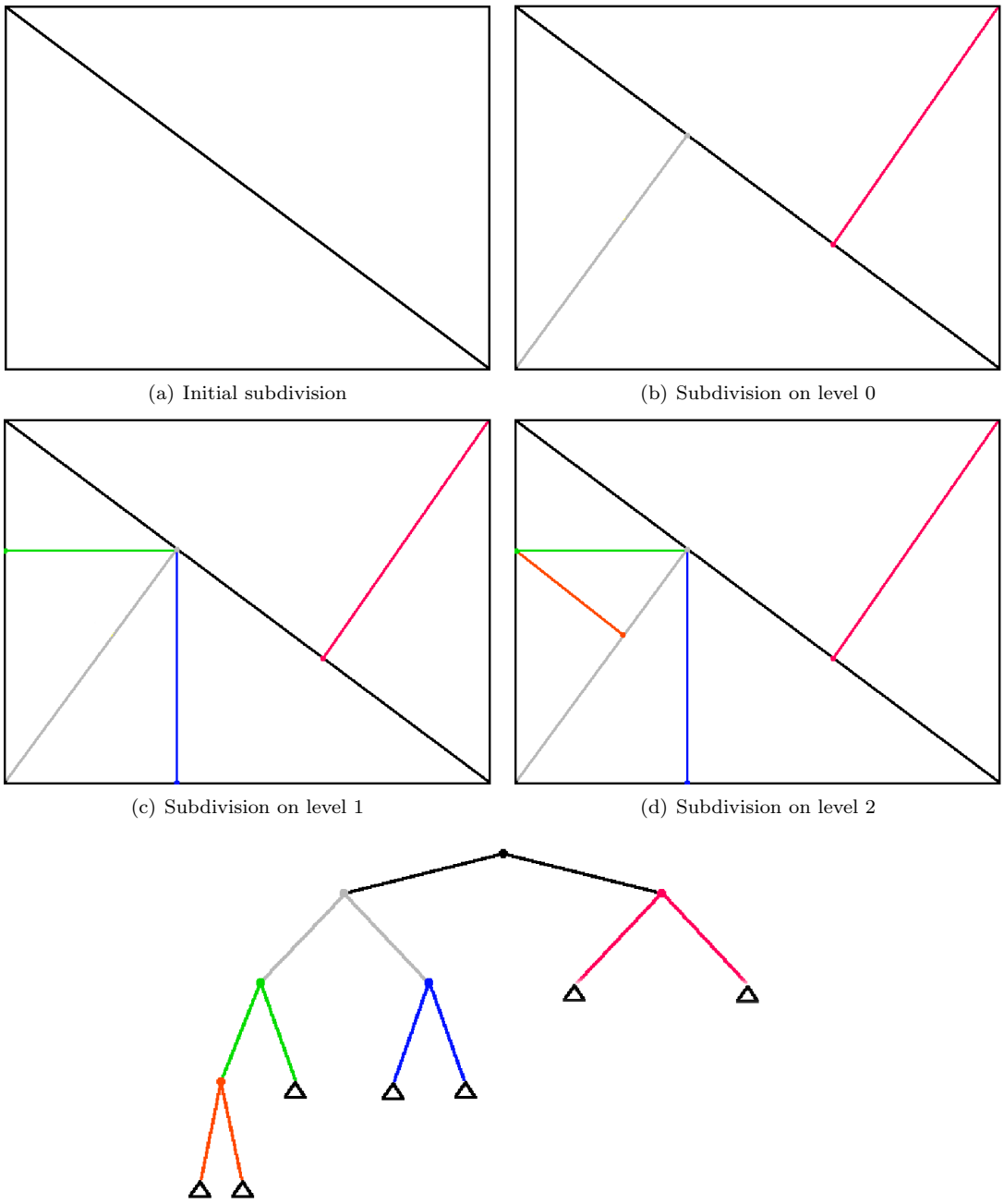
with Neumann boundary conditions to reconstruct the missing color values in frame u.

Depending on the diffusion tensor D we distinguish *linear homogeneous diffusion (HD)*

$$D = E, \tag{5}$$

nonlinear isotropic diffusion (NID)

$$D = g(\|\nabla u_\sigma\|^2)E, \tag{6}$$



(a) Initial subdivision

(b) Subdivision on level 0

(c) Subdivision on level 1

(d) Subdivision on level 2

(e) Corresponding tree

Figure 1: Subdivision of an image domain. The resulting bitlist is 1 1 | 1 1 0 0 | 1 0 0 0 | 0 0.

and *nonlinear anisotropic diffusion (NAD)*

$$D = g\left(D_u\right),\ D_u = \nabla u_\sigma \left(\nabla u_\sigma\right)^T. \tag{7}$$

E denotes the identity matrix and the subscript σ the standard deviation for a convolution of u with a Gaussian mask.

The so-called diffusivity function $g : \mathbb{R}_0^+ \to [0,1]$ is a monotonically decreasing function with $g(0) = 1$ and $\lim_{s\to+\infty} g(s) = 0$. It determines the influence of the image data on the diffusion

process. We use the Charbonnier-diffusivity given by [12]

$$g(s^2) = \frac{1}{\sqrt{1 + s^2/\beta^2}}. \tag{8}$$

$\beta > 0$ denotes a contrast parameter.

The idea of the nonlinear diffusion models is to reduce the diffusion at (NID) or across (NAD) image discontinuities to preserve image edges.

In the anisotropic case [13, 14] the diffusion tensor

$$D = \lambda_1 v_1 v_1^T + \lambda_2 v_2 v_2^T \tag{9}$$

is constructed by applying the diffusivity function g to the eigenvalues $\mu_i, i \in \{1, 2\}$ of D_u, resulting in $\lambda_1 = g(\mu_1), \lambda_2 = g(\mu_2)$, without changing the eigenvectors v_1, v_2 of D_u. The eigenvectors are orthogonal and have the properties

$$v_1 \parallel \nabla u, \quad v_2 \perp \nabla u. \tag{10}$$

To smooth along edges but not across them, the eigenvalues are set to

$$\lambda_1 = g(\|\nabla u\|), \quad \lambda_2 = g(0). \tag{11}$$

The inpainting of the missing points can be thought of as a kind of interpolation, e.g., in 1D, homogeneous diffusion corresponds to linear interpolation, in higher dimensions to using radial basis functions [15]. More details on interpolation methods in general can be found in [16] and information about PDE based interpolation methods and their properties can be found in [17, 8, 18].

3.2 Discretization and Solution

To solve (4) we distinguish the linear and the nonlinear case.

In the linear case we discretize the PDE by finite differences and solve the resulting linear system

$$A^h u^h = f^h \tag{12}$$

by a full multigrid solver [19, 20, 21]. It works on a hierarchy of node-centered grids or levels, starts from the coarsest level and successively performs one or more recursive V-cycles(ν_1, ν_2) (see Algorithm 1) on each level. For simplicity, we describe the V-cycle for two grids only, where we denote the fine grid by h and the coarse grid by H. We start with an initial guess $u_h^0 = 0$, use ν_1 iterations of a red-black Gauss-Seidel method as pre-smoothing operator S^{ν_1}, and ν_2 Gauss-Seidel iterations as post-smoothing operator S^{ν_2}. Then, the residual r^h is computed and restricted by full weighting. On the coarser grids, we solve the so-called error equation $A^H e^H = r^H$ recursively. The coarse grid matrix A^H is constructed by rediscretizing (4) on level H. After that we transfer the error e^H back to the fine grid by bilinear interpolation and correct the current solution. Finally, we apply some post-smoothing steps. As stopping criterion we check if the L_2-norm of the residual per point is below a certain threshold (typically 10^{-8}).

In the implementation we fix the solution u^h at each landmark $\boldsymbol{x} \in \Omega_C$. Thus the landmarks are treated as Dirichlet boundary values inside Ω, because they remain unchanged during the solution process.

The landmarks introduce singularities, which can be detected as small points in the resulting images, since the fundamental solution tends to infinity at these locations. This may lead to a substantial decrease of the multigrid convergence rate, if there is no proper treatment of the slowly converging error components. This problem is also known as *small islands* in the multigrid context and discussed in detail in [22].

Additionally, when using multigrid, the question arises how to represent the landmarks on the coarser levels. In principle there are two possibilities, we may either simply neglect them or we may extend them on coarser levels.

In the first case, we may solve exactly – or do local relaxations – in the neighborhood of each landmark, what is useful if there are only a few landmarks. In our application we typically have many landmarks, therefore we apply Krylov subspace acceleration or iterant recombination [20, 23]

Algorithm 1 Multigrid V-cycle: Compute $u_h^{(k+1)} = M_h(u_h^{(k)}, A^h, f^h, \nu_1, \nu_2)$

$\bar{u}_h^{(k)} = S_h^{\nu_1}(u_h^{(k)}, A^h, f^h)$ {pre-smoothing}
$r^h = f^h - A^h \bar{u}_h^{(k)}$ {compute residual}
$r^H = \mathcal{I}_h^H r^h$ {restrict residual}
if number of coarse grid points $< \epsilon_{min}$ **then**
  Solve $A^H e^H = r^H$ exactly
else
  $e^H = M_H(0, A^H, r^H, \nu_1, \nu_2)$
end if
$e^h = \mathcal{I}_H^h e^H$ {interpolate error}
$\tilde{u}_h^{(k)} = \bar{u}_h^{(k)} + e^h$ {coarse grid correction}
$u_h^{(k+1)} = S_h^{\nu_2}(\tilde{u}_h^{(k)}, A^h, f^h)$ {post-smoothing}

instead. The idea of iterant recombination is to linearly combine two or more previous multigrid V-cycle iterants such that the resulting accelerated solution $u_{h,acc}^{(k)}$ minimizes the residual. In order to compute $u_{h,acc}^{(k)}$ we consider a linear combination of the $\tilde{k}$+1 latest multigrid iterants $u_h^{(k-\tilde{k})}, \ldots, u_h^{(k)}$:

$$u_{h,acc}^{(k)} = u_h^{(k)} + \sum_{i=1}^{\tilde{k}} \alpha_i \left(u_h^{(k-i)} - u_h^{(k)} \right), \; k \geq \tilde{k}, \tag{13}$$

or, since $\sum_i \alpha_i = 1$

$$u_{h,acc}^{(k)} = \sum_{i=1}^{\tilde{k}} \alpha_i u_h^{(k-i)} \quad . \tag{14}$$

The coefficients α_i are chosen such that the residual $r_{h,acc}^{(k)}$ is minimized

$$\| r_{h,acc}^{(k)} \| = \left\| \sum_{i=1}^{\tilde{k}} \alpha_i r_h^{(k-i)} \right\|_2 , \tag{15}$$

what is equivalent to solving the overdetermined linear system

$$A^h \left(u_h^{(k-1)}, \ldots, u_h^{(k-\tilde{k})} \right) \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_{\tilde{k}} \end{pmatrix} = f^h. \tag{16}$$

Since the system (16) may be ill-conditioned we apply a singular value decomposition (SVD) to solve it. To speed up the method, we may apply the acceleration only on coarser grids.

If we extend the landmarks on coarser levels, we fix all points on the coarse grid to 0 that interpolate to one or more landmarks. This leads already to acceptable convergence rates which can be further improved by iterant recombination.

In the nonlinear case we treat the nonlinearity by an *inexact lagged diffusivity* method [24, 25]. The idea is to fix the diffusion tensor D during one Gauss-Seidel step, and update the values of D in the whole domain after each Gauss-Seidel iteration [26, 27, 28]. The discretization is done by finite volumes which corresponds to a symmetric finite difference discretization [20]. Additionally, we apply a multilevel damped Jacobi or red-black Gauss-Seidel solver, i.e., we start on a coarse level and perform enough Gauss-Seidel iterations to obtain a solution there. For this purpose we construct an image pyramid and store landmarks on each level. This solution is interpolated to the next finer level and used there as initial guess for the solver. This scheme is repeated until we have reached the finest level.

Note that for a large number of landmarks ($> 10\%$) there is no need to use multilevel or multigrid solvers any more, since also few Gauss-Seidel iterations lead to an acceptable result.

4 Results

4.1 Compression of Medical Data Sets

In this section we compare the different regularizers for image inpainting on MRI images (see Fig. 2) and show the effect of the landmarks on the linear multigrid solver.

We randomly choose 1% of the points as landmarks and do not use the BTTC scheme, because the effects of the multigrid solver can be visualized better for the randomly chosen points. Of course, this choice does not influence the performance or convergence rate of the multigrid solver. Furthermore, the results from this section enable us to compare the improved image quality of the results when using the BTTC scheme later on.

Figure 2: Original 2D slice of MRI image of human head (size 512×512).

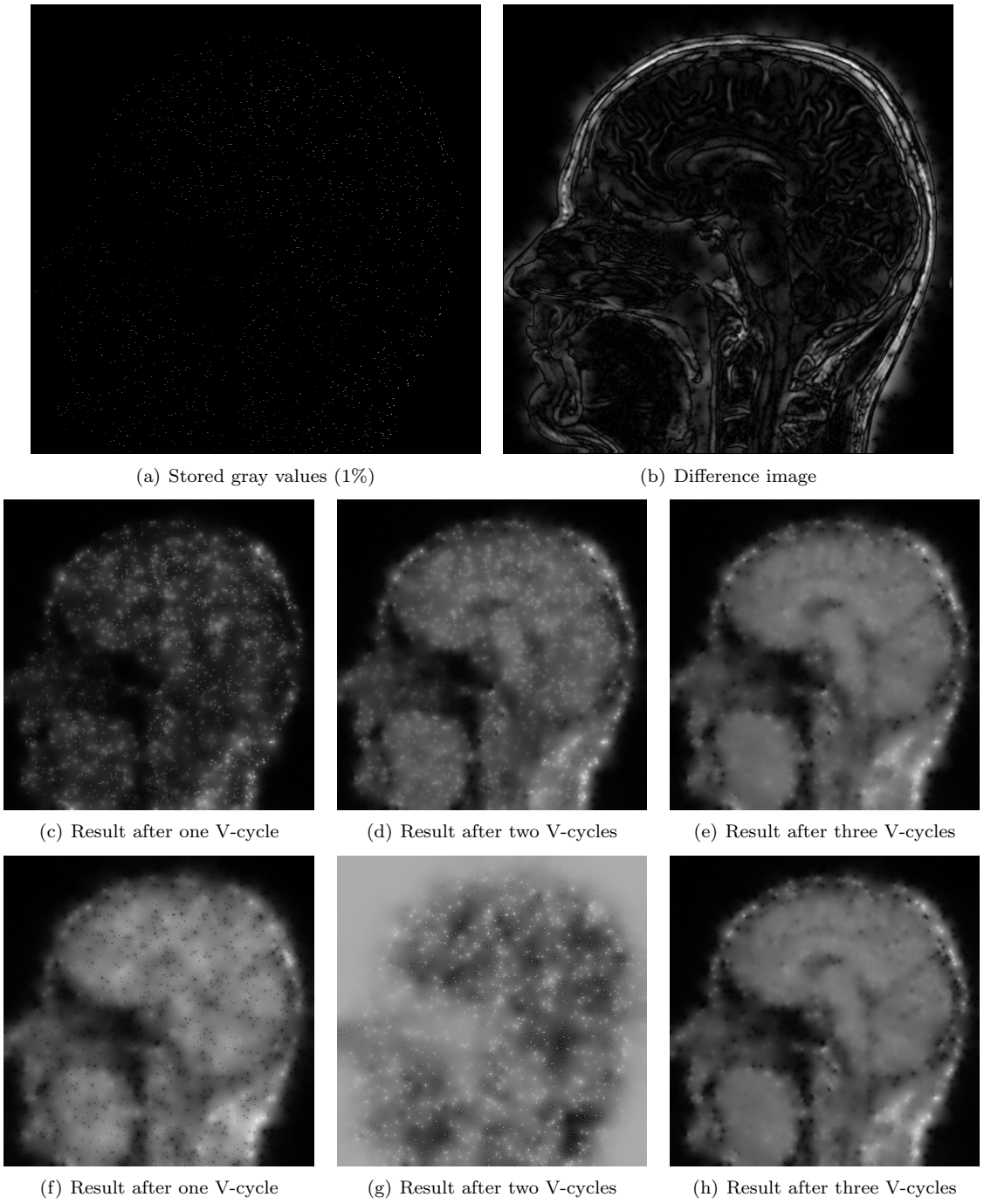
In Fig. 3 we present visual results for linear homogeneous diffusion (HD) using a node-based discretization and multigrid V(2,2)-cycles with a Gauss-Seidel smoother. If we ignore the landmarks on coarser levels, we apply once iterant recombination after the second V-cycle. This method has the advantage that no special treatment of the landmarks is necessary on coarser levels what enables an easier implementation and use of performance optimization techniques for the multigrid solver [29].

We compare the three different kinds of diffusion in Fig. 4. To measure the quality of the image inpainting we compute in Table 1 the relative reconstruction errors defined by

$$e_{rr} = \frac{\left\| u_{\text{orig}} - u^h \right\|}{\left\| u_{\text{orig}} - u^0 \right\|}, \tag{17}$$

where u_{orig} is the original image, u^0 the image consisting of the stored landmarks, and u^h the solution image computed by multigrid. Small values of e_{rr} indicate only that the solution image

is somehow similar – in the L_2-norm – to the original image, but do not quantify the visual image quality, e.g., whether image edges are well represented.



(a) Stored gray values (1%) (b) Difference image

(c) Result after one V-cycle (d) Result after two V-cycles (e) Result after three V-cycles

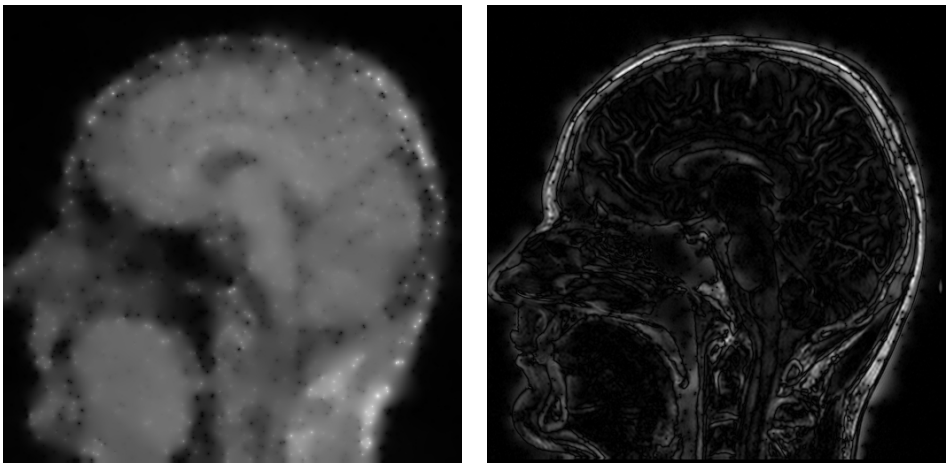
(f) Result after one V-cycle (g) Result after two V-cycles (h) Result after three V-cycles

Figure 3: Restoration of MRI image using homogeneous diffusion and multigrid with (c–e) and without (f–h) special treatment of landmarks. Both methods converge with similar accuracy within 3 V(2,2)-cycles.

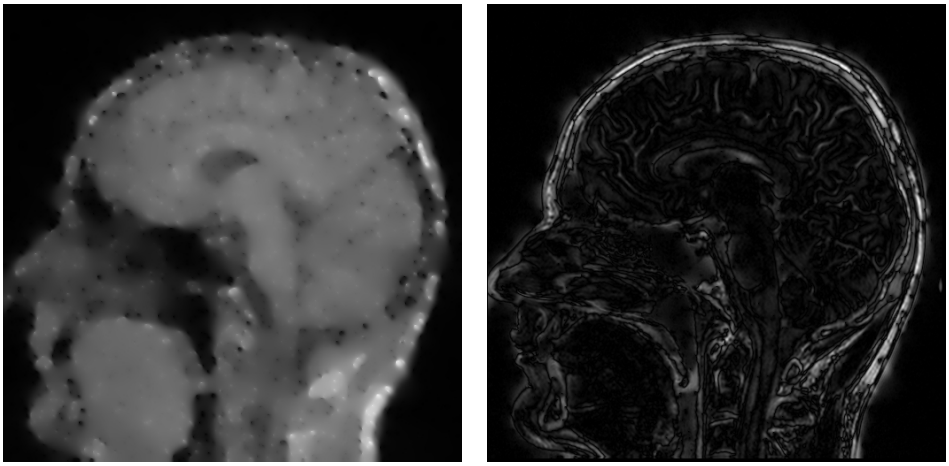
4.2 Achieving Real Time Video Decompression on a PlayStation™ 3

The STI[1]Cell Processor used, e.g., in the new PlayStation™ 3 (see 6) is an innovative, heteroge-

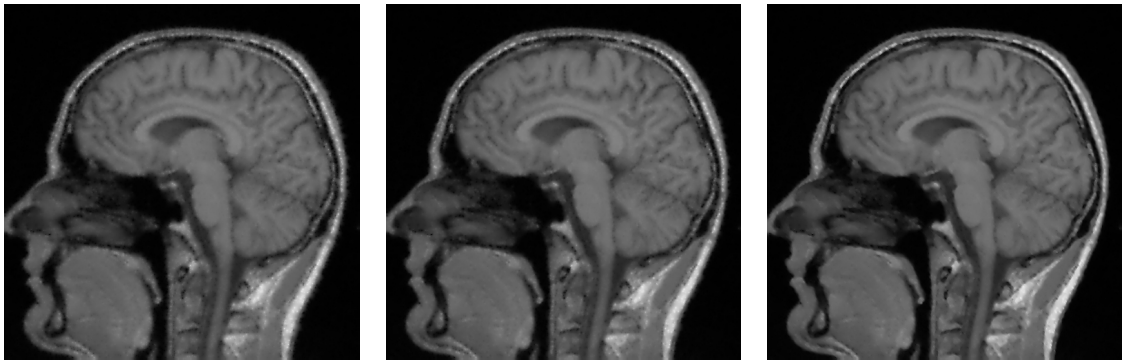
[1]Sony, Toshiba and IBM



(a) NID regularizer (1%) (b) Difference image



(c) NAD regularizer (1%) (d) Difference image



(e) HD regularizer (10%) (f) NID regularizer (10%) (g) NAD regularizer (10%)

Figure 4: Restoration of MRI image using different number of randomly chosen landmarks and different regularizers.

nous multi-core processor that offers outstanding performance on a single chip.

The organization of the processor is depicted in Fig. 7: The backbone of the chip is a fast ring bus – the Element Interconnect Bus (EIB) – connecting all units on the chip and providing a throughput of up to 204.8 GB/s in total at 3.2 GHz. A PowerPC-based general purpose core – the Power Processor Unit (PPU) – is primarily used to run the operating system and control execution. The Memory Interface Controller (MIC) can deliver data with up to 25.6 GB/s from Rambus XDR memory and the Broadband Engine Interface (BEI) provides fast access to IO devices or a

Table 1: Relative reconstruction errors e_{rr} for 2D MRI image.

regularizer	1% landmarks	10% landmarks
HD	0.219	0.082
NID	0.209	0.073
NAD	0.191	0.063

(a) Uncompressed (b) PDEVC

Figure 5: Image quality of real time decompressed PDEVC vs. uncompressed video file (size 320×240).

coherent connection to other Cell processors. The computational power resides in eight Synergistic Processor Elements (SPEs), simple but very powerful co-processors consisting of three components: The Synergistic Processor Unit (SPU) is a custom SIMD only vector engine with a 128 vector register and two pipelines. The SPU operates on 256 kB of own Local Store (LS), a very fast, low-latency memory. Each SPU further has a dedicated interface unit, connecting it to the outside world: the primary use of the Memory Flow Controller (MFC) is to asynchronously copy data between Local Store and main memory or Local Stores of other SPEs using Direct Memory Access. It also provides communication channels to the PPU or other SPUs and is utilized by the PPU to control execution of the associated SPU. Each SPE can be seen as a very simple computer performing its own program, but dependent on and controlled by the PPU.

Usually, a single Cell processor is able to perform 204.8 GFlop/s using fused-multiply-adds in single precision (not counting the abilities of the PPU). However, only six SPEs are available for Linux running as a guest system on the Sony PlayStation™ 3, what reduces the maximum performance accordingly.

C++ was used to implement the code on the PPU and SPUs, using especially compiler intrinsics in the SPU code. The code was developed using IBM's Cell SDK 2.0 and compiled using the bundled IBM xlc compilers.

Data for one color of a single frame, i. e., a partial differential equation, does not fit into 256 kB Local Store. Instead, only some lines can be available at a time. The technique used is very similar to cache blocking and creates a wave propagating line by line through the grid. But in this case the MFC must be explicitly instructed to copy new lines to Local Store and writing updated lines back to main memory. For the homogeneous diffusion red-black Gauss-Seidel solver, e. g., a red update can take place after the next line has become available. The line above had its red update before, so the black values can be updated and the line be written back to memory. Also coefficients, necessary for the more advanced models, can be calculated on the SPEs "just in time", this way main memory space and bandwidth can be saved.

Next we take a closer look at the computational costs of the decompression scheme using different regularizers. Therefore, we measure the number of frames per second (fps) that we are able to

Figure 6: PlayStation™ 3.

decompress for several regularizers in Table 2 for a video file of size 320×240. About 10% of the image points are selected as landmarks on each level by BTTC. The reconstruction errors and the visual quality of the results is better than using random selection.

For the NAD regularizer we apply 130 damped Jacobi iterations on each level that lead to an acceptable image quality and achieve 25 fps. To obtain comparable times, we use for the NID and the HD regularizer 65 red-black Gauss-Seidel iterations. This is typically not enough for convergence, i.e., we require some more $(20 - 30\%)$ iterations for these two regularizers to compute the decompressed image.

Table 2: Comparison of decompression times (in fps) and reconstruction errors for different regularizers using different number of levels for a resolution of 320×240 pixels.

	1 level	3 levels	
Regularizer	fps	fps	e_{rr}
HD	101	89	0.06
NID	48	35	0.06
NAD	34	25	0.05

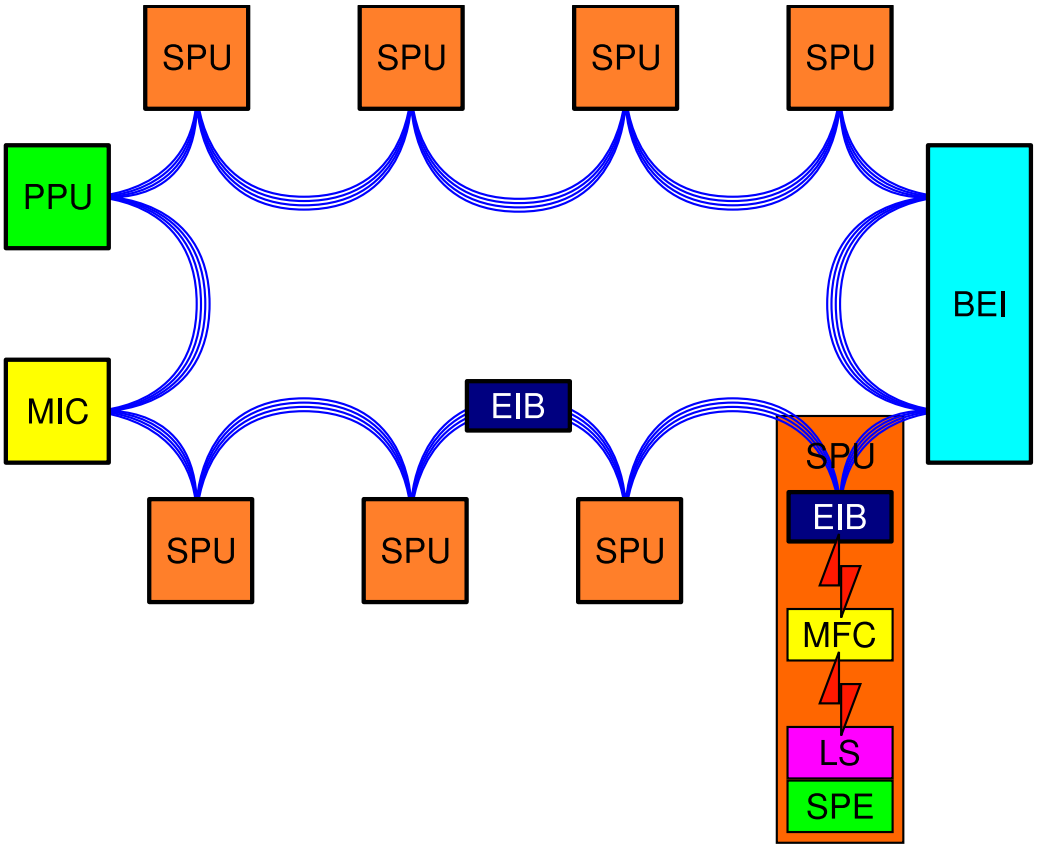
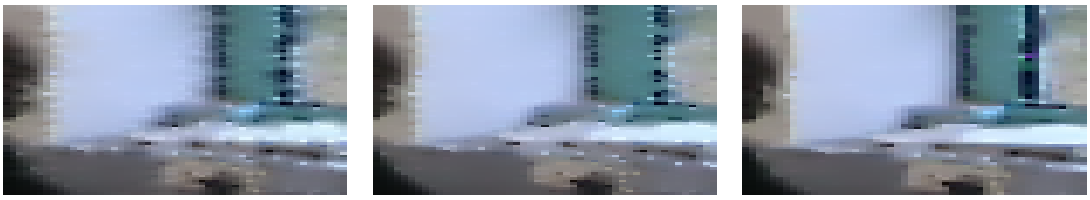


Figure 7: Schematic view of the STI Cell Processor



(a) HD (b) NID (c) NAD

Figure 8: Comparison of HD, NID, and NAD regularizer for PDEVC on PlayStation™ 3 at the section marked in Fig. 5(b).

5 Conclusions and Future Work

Although only a first step towards a real video codec is taken, we have shown that variational approaches may be used for real time video coding.

Potential for some more optimizations lies for example in using the YCbCr color space instead of the RGB color space, which allows subsampling of the chrominance channels. While this is not visible for the human eye it may save a large amount of memory for compression. As done by most other coding schemes, like, e.g., MPEG-1, we should make use of correspondencies between frames when compressing the source data. One idea here would be to store information about the motion field in addition to the landmarks and include this information in the decompression. Finally, also the BTTC scheme could be replaced by a more advanced method choosing the landmarks in a way that takes into account the used diffusion model within the decompression scheme.

References

[1] T. Milde, *Videokompressionsverfahren im Vergleich*, 1st ed. dpunkt Verlag für digitale Technologie GmbH, Heidelberg, 1995.

[2] T. Strutz, *Bilddatenkompression*, 3rd ed. Vieweg, Wiesbaden, Germany, 2005.

[3] I. Galic, *et al.*, "Towards PDE-based image compression," in *Proceedings of variational, geometric, and level set methods in computer vision*, ser. Lecture Notes in Computer Science. Springer-Verlag, Berlin, Heidelberg, New York, 2005, pp. 37–48.

[4] R. Distasi, *et al.*, "Image compression by B-tree triangular coding," *IEEE Transactions on Communications*, vol. 45, no. 9, pp. 1095–1100, 1997.

[5] D. Huffman, "A method for the construction of minimum redundancy codes," *Proceedings of the Institute of Radio Engineers*, vol. 40, no. 9, pp. 1098–1101, 1952.

[6] P. Münch and H. Köstler, "Videocoding using a variational approach for decompression," Department of Computer Science 10 (System Simulation), Friedrich-Alexander-University of Erlangen-Nuremberg, Germany, Tech. Rep. 07-1, 2007.

[7] M. Bertalmío, *et al.*, "Navier-Stokes, Fluid Dynamics, and Image and Video Inpainting," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001, pp. 355–362.

[8] T. Chan and J. Shen, "Nontexture inpainting by curvature driven diffusions (CDD)," *J. of Visual Comm. and Image Representation*, vol. 12, no. 4, pp. 436–449, 2001.

[9] F. Lauze and M. Nielsen, "A variational algorithm for motion compensated inpainting," in *British Machine Vision Conference*, 2004.

[10] P. Perona and J. Malik, "Scale-space and edge detection using anisotropic diffusion," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 7, pp. 629–639, 1990.

[11] F. Catte, *et al.*, "Image Selective Smoothing and Edge Detection by Nonlinear Diffusion," *SIAM Journal on Numerical Analysis*, vol. 29, no. 1, pp. 182–193, 1992.

[12] P. Charbonnier, *et al.*, "Two deterministic half-quadratic regularization algorithms for computed imaging," in *Proceedings IEEE International Conference on Image Processing*, vol. 2, Austin, TX, USA, 1994, pp. 168–172.

[13] J. Weickert, "Theoretical foundations of anisotropic diffusion in image processing," *Computing*, vol. 11, pp. 221–236, 1996.

[14] J. Weickert, *Anisotropic Diffusion in Image Processing*. Teubner Verlag, Stuttgart, Germany, 1998.

[15] M. Buhmann, *Radial Basis Functions*. Cambridge University Press, Cambridge, UK, 2003.

[16] T. Lehmann, *et al.*, "Survey: Interpolation Methods in Medical Image Processing," *IEEE Transactions on Medical Imaging*, vol. 18, no. 11, pp. 1049–1075, 1999.

[17] V. Caselles, *et al.*, "An axiomatic approach to image interpolation," *IEEE Transactions on Image Processing*, vol. 7, no. 3, pp. 376–386, 1998.

[18] J. Weickert and H. Hagen, *Visualization and Processing of Tensor Fields (Mathematics and Visualization)*. Springer-Verlag, Berlin, Heidelberg, New York, 2005.

[19] A. Brandt, "Multi-Level Adaptive Solutions to Boundary-Value Problems," *Mathematics of Computation*, vol. 31, no. 138, pp. 333–390, 1977.

[20] U. Trottenberg, *et al.*, *Multigrid*. Academic Press, San Diego, CA, USA, 2001.

[21] W. Briggs, *et al.*, *A Multigrid Tutorial*, 2nd ed. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, USA, 2000.

[22] A. Brandt and V. Mikulinsky, "On recombining iterants in multigrid algorithms and problems with small islands," *SIAM J. on Scientific Computing*, vol. 16, pp. 20–28, 1995.

[23] I. Christadler, *et al.*, "Robust and efficient multigrid techniques for the optical flow problem using different regularizers," in *Proceedings of 18th Symposium Simulationstechnique ASIM 2005*, ser. Frontiers in Simulation, F. Hülsemann, *et al.*, Eds., vol. 15. SCS Publishing House, 2005, pp. 341–346, also exists as Tech. Rep. 05-6.

[24] T. Chan and P. Mulet, "On the convergence of the lagged diffusivity fixed point method in total variation image restoration," *SIAM Journal on Numerical Analysis*, vol. 36, no. 2, pp. 354–367, 1999.

[25] S. Fućik, *et al.*, "Kacanov-Galerkin method," *Comment. Math. Univ. Carolinae*, vol. 14, no. 4, pp. 651–659, 1973.

[26] A. Bruhn, "Variational optic flow computation: Accurate modeling and efficient numerics," Ph.D. dissertation, Department of Mathematics, Saarland University, Saarbrücken, Germany, 2006.

[27] C. Frohn-Schauf, *et al.*, "Nonlinear multigrid methods for total variation image denoising," *Computing and Visualization in Science*, vol. 7, no. 3, pp. 199–206, 2004.

[28] C. Vogel, *Computational Methods for Inverse Problems.* Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, USA, 2002.

[29] C. Douglas, *et al.*, "Cache Optimization for Structured and Unstructured Grid Multigrid," *Electronic Transactions on Numerical Analysis (ETNA)*, vol. 10, pp. 21–40, 2000.