Fast Interpolation of Dense Motion Fields from Synthetic Phantoms

Andreas Maier[1,2], Oliver Taubmann[1], Jens Wetzl[1], Jakob Wasza[1],
Christoph Forman[1,2], Peter Fischer[1,2], Joachim Hornegger[1,2], Rebecca Fahrig[3]

[1] Pattern Recognition Lab, FAU Erlangen-Nuremberg
[2] Erlangen Graduate School in Advanced Optical Technologies (SAOT)
[3] Radiological Sciences Lab, Stanford University
andreas.maier@fau.de

Abstract. Numerical phantoms are a common tool for the evaluation of registration and reconstruction algorithms. For applications concerning motion, dense deformation fields are of particular interest. Phantoms, however, are often described as surfaces and thus motion vectors can only be generated at these surfaces. In order to create dense motion fields, interpolation is required. A frequently used method for this purpose is the Parzen interpolator. However, with a high number of surface motion vectors and a high voxel count, its run time increases dramatically. In this paper, we investigate different methods to accelerate the creation of these motion fields using hierarchical sampling and the random ball cover. In the results, we show that a 64^3 volume can be sampled in less than one second with an error below 0.1 mm. Furthermore, we accelerate the interpolation of a 256^3 dense deformation field to only $\tilde{6}.5$ minutes using the proposed methods from days with previous methods.

1 Introduction

The systematic evaluation and comparison of non-rigid image registration and reconstruction methods requires the availability of ground truth data in the form of dense deformation fields. Ideally, this data should not come from a regularized registration or interpolation methods in order to to prevent bias towards a certain approach, e.g. ground truth from a thin-plate-spline-based interpolation will favor other thin-plate-spline based methods although they might be a more realistic with respect to anatomy [1].

The 4-D extended cardio-torso (XCAT) phantom [2] defines per-organ deformations over time and can be used to generate medical image sequences with somewhat realistic respiratory or cardiac motion and also provides corresponding sparse displacement vector fields defined on its parametric surfaces.

The main contribution of this article is a graphics processing unit (GPU) accelerated method to compute dense displacement vector fields estimating the motion inside and outside of organs from the given surface motion.

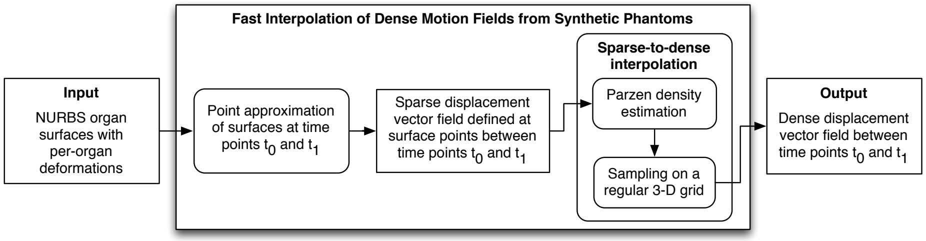
Fast Interpolation of Dense Motion Fields from Synthetic Phantoms
Input
NURBS organ surfaces with per-organ deformations
Point approximation of surfaces at time points t0 and t1
Sparse displacement vector field defined at surface points between time points t0 and t1
Sparse-to-dense interpolation
Parzen density estimation
Sampling on a regular 3-D grid
Output
Dense displacement vector field between time points t0 and t1

Fig. 1. System overview.

2 Materials and Methods

In our two-step method (cf. Fig. 1), we first generate accurate point approximations of the parametric B-spline surfaces [3]. The motion sparsely defined at each of these points is then interpolated efficiently on a dense regular grid to obtain full respiratory and/or cardiac motion fields covering the upper body.

2.1 Efficient Point Approximation of Parametric Surfaces

In order to find point samples that will serve as nodes for the interpolation, we finely tessellate the parametric surfaces $\boldsymbol{s}(u, v, t)$ defined in terms of control points $\boldsymbol{c}_{l,k,i}$ and their weights $\beta(u)$ of degree 3:

$$\boldsymbol{s}(u, v, t) = \sum_i \sum_k \sum_l \boldsymbol{c}_{l,k,i} \beta(u - l)\beta(v - k)\beta(t - i) \tag{1}$$

$$\text{with} \quad \beta(u) = \begin{cases} 0, & |u| \geq 2 \\ \frac{1}{6}(2 - |u|)^3, & 1 \leq |u| < 2 \\ \frac{2}{3} - \frac{1}{2}|u|^2(2 - |u|), & |u| < 1 \end{cases} \tag{2}$$

Note that this tessellation can be implemented even more efficiently using texture units on the graphics card [3] at the cost of slightly decreased accuracy. For the present work, we used double precision CPU computations, as we wanted to investigate the accuracy of our interpolation approaches.

2.2 Fast Interpolation of Deformations on a Regular Grid

We now consider a regular 3-D grid with the desired properties, e.g. size, location and spacing. For each grid point $\boldsymbol{x}$, we aim to find an estimate of its motion $\boldsymbol{d}_\sigma(\boldsymbol{x})$ from a combination of the displacements defined at the closest surface points. In the following, we only consider the evaluation of a deformation field from time t_0 to t_1, i.e. the n surface points at time t_0 are denoted as $\boldsymbol{x}_i := \boldsymbol{s}(u_i, v_i, t_0)$ and their deformation as $\boldsymbol{d}_i := \boldsymbol{s}(u_i, v_i, t_1) - \boldsymbol{s}(u_i, v_i, t_0)$. For interpolation, we employ a Parzen density estimation [4] with a Gaussian kernel function.

$$\hat{\boldsymbol{d}}_\sigma(\boldsymbol{x}) = \frac{1}{\sum_{i=1}^{n} w(\boldsymbol{x}, \boldsymbol{x}_i, \sigma)} \sum_{i=1}^{n} w(\boldsymbol{x}, \boldsymbol{x}_i, \sigma) \boldsymbol{d}_i, \tag{3}$$

$$\text{with} \quad w(\boldsymbol{x}, \boldsymbol{x}_i, \sigma) = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}_i\|_2^2}{2\sigma^2} + K\right) \tag{4}$$

The parameter σ is used to control the smoothness of the interpolation and K is a constant that is added to increase numerical stability of the Gaussian kernel. Note that as the weights $w(\boldsymbol{x}, \boldsymbol{x}_i, \sigma)$ appear in the numerator and the denominator, K cancels out. In our experiments, we chose $K = 70$.

The performance bottleneck of this approach is the summation over i, which has to be repeated for each voxel. Given a grid of 256^3 points and sparse vector field with about 700,000 vectors, $1.0 \cdot 10^{13}$ evaluations of Eq. 4 have to be performed. Such computations usually take hours to days to finish. However, as the weight decays exponentially with increasing distance, we are able to reduce the number of points that have to be considered at each voxel. In the following, we will describe two hierarchical methods to achieve this reduction, and one randomized method.

Slice Selection For parallelization on the GPU, we chose to run each kernel execution for an individual slice. Thus, it makes sense to preselect only points that are close to the current slice. As the distance threshold T_z, we selected 10 % of the volume size in the z direction plus 6 standard deviations σ of the Gaussian kernel. This leads to a reduced list of motion vector points per slice $\boldsymbol{x}_{i,S}$ for use in Eq. 3 instead of $\boldsymbol{x}_i$.

Gridding In addition to Slice Selection, we can further reduce the number of points per voxel by a Gridding approach. For each slice, we first create a list of points $\boldsymbol{g}_j$ forming a sub-grid at a lower resolution than the sampling grid with step size p. Next, a list of close points $\boldsymbol{x}_{i,\boldsymbol{g}_j}$ is created for each $\boldsymbol{g}_j$. Each $\boldsymbol{x}_{i,S}$ that is closer than $T_p := 2p + 6\sigma$ is added to this list. For the evaluation, first the correct sub-list $\boldsymbol{x}_{i,\boldsymbol{g}_j}$ has to be selected by finding the $\boldsymbol{g}_j$ that is closest to $\boldsymbol{x}$. Next, Eq. 3 is applied using $\boldsymbol{x}_{i,\boldsymbol{g}_j}$ instead of $\boldsymbol{x}_i$ which leads to a dramatic reduction of search points. A sub-sampling factor of 16 was used in our experiments.

Random Ball Cover The selection process described in the previous section bears striking similarity to the Random Ball Cover (RBC) [5]. Thus, we propose an adaptation of this method. Instead of using a regular grid as in the previous method, we use the structure of the data vectors and select representatives $\boldsymbol{g}_r$ randomly. Again, a list for each representative has to be generated. In order to pick an appropriate distance threshold, we have to determine the maximal minimum distance p^* between all points. Then the threshold T_r is found as

$$T_r = p^* + \min(6\sigma, p^*)$$

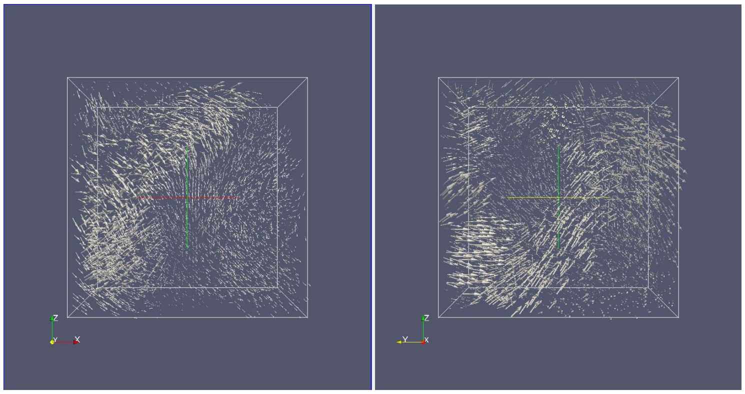

Fig. 2. Two perpendicular views of the cardiac motion field generated with the XCAT heart and the given configuration. The figure nicely shows the complexity of the 3-D pumping motion of the heart.

which evaluates to $p^* + 6\sigma$ for few representatives and $2p^*$ for many representatives. The list $\boldsymbol{x}_{i,\boldsymbol{g}_r}$ is then built in the same way as in the previous approach and used in Eq. 3.

3 Experiments and Results

All steps of the proposed methods have been implemented and optimized to run on the GPU using the Open Computing Language (OpenCL) framework. Our implementation will be made available as part of the CONRAD software platform [6] designed for simulating basic processes in X-ray imaging and the evaluation of image reconstruction algorithms [7].

In order to measure the evaluation accuracy, we investigated our methods using the XCAT heart using a 64^3 voxel volume only. The voxel size was 2.5 mm (isotropic). In total, 4,500 vectors have to be interpolated at each grid point. The ground truth motion field obtained without any search point reduction is shown in Fig. 2. We measured total computation time, kernel execution time, number of points used for interpolation, and the root mean square error (RMSE) between each method and the ground truth deformation field computed with CPU double accuracy. The results are given in Table 1. With varying σ, we observe that the thresholds we chose yield stable results for all methods. Errors are virtually always below 0.1 mm. With the presented parameters, RBC performs best. However, reduction rates are the highest with the Gridding approach, which comes with a slight increase in computational error.

Next, we evaluated the three methods with a bigger sampling grid using 256^3 voxels with an isotropic size of 1.5 mm and 700,000 motion vectors. With this number of vectors, we are not able to evaluate the plain OpenCL and the Slice Selection approach, as the computation time per slice exceeds 2 s. This causes

algorithm	kernel time [s]	total time [s]	used vectors [%]	RMSE [mm]
$\sigma = 1$				
CPU	-	274.107	100.0	0.0
OpenCL	4.961	5.241	100.0	2.44 E-6
Slice Selection	1.591	1.903	27.5	2.44 E-6
Gridding	1.544	1.857	3.9	0.096
RBC 500	0.951	1.498	13.5	0.012
RBC 250	0.874	1.435	15.6	0.029
RBC 50	1.077	1.450	18.7	0.137
$\sigma = 5$				
CPU	-	432.691	100.0	0.0
OpenCL	4.960	5.289	100.0	2.66 E-6
Slice Selection	3.027	3.432	56.4	0.084
Gridding	3.244	3.619	14.5	0.084
RBC 500	1.685	2.247	28.3	0.001
RBC 250	2.013	2.480	38.0	0.001
RBC 50	2.793	3.198	53.1	2.72 E-5

Table 1. Summary of the results using a 64^3 deformation field of the XCAT heart phantom.

the graphics driver to terminate the kernel execution. While there are ways to circumvent this, we generally do not want to pursue this direction as execution times are too long. For the Gridding approach, we were able to compute the complete deformation field in 389 s with about 200 vectors used per representative. For the RBC, we were not able to obtain similar results. With 50,000 representatives, the memory required to store the search list exceeded our graphics card memory. With 500,000 representatives, we already need $3.5 \cdot 10^{11}$ distance comparisons just to build the RBC list. This is almost as complex as the original problem. Thus, only the Gridding approach was suitable for the large problem size. Fig. 3 shows the resulting motion vector field.

4 Conclusion

We have developed a fast method to generate dense motion vector fields from phantom data. In the small problem size, all three proposed methods worked well with an error of less than 0.1 mm. For the large problem size, only the Gridding approach could be successfully applied, as it reduced the number of interpolation vectors by the largest factor. In this case, we report a runtime of about 6.5 minutes.

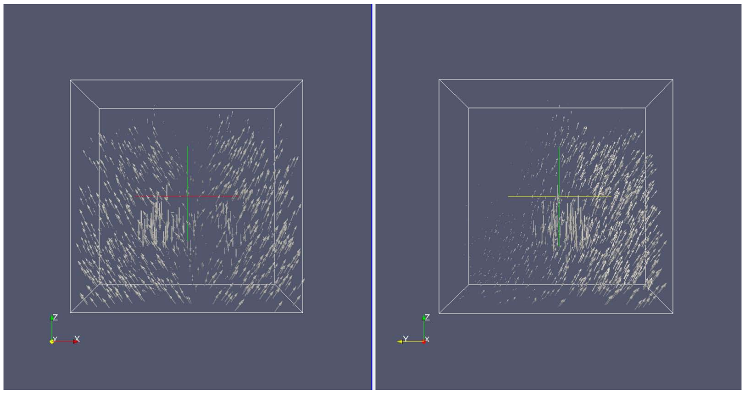

Fig. 3. Two perpendicular views of the breathing motion field representing an inhalation, generated with the XCAT torso and the given configuration. The vectors pointing downward are linked to the downward motion of the diaphragm in the center of the volume. The upward (along the z direction) and outward (against the y direction) motion shows the expanding chest motion.

Acknowledgements The authors gratefully acknowledge funding of the Erlangen Graduate School in Advanced Optical Technologies (SAOT) and the Research Training Group Heterogeneous Imaging Systems by the German Research Foundation (DFG) in the framework of the German excellence initiative.

References

1. von Berg J, Barschdorf H, Blaffert T, Kabus S, Lorenz C. Surface based cardiac and respiratory motion extraction for pulmonary structures from multi-phase CT. In: Proc. SPIE. vol. 6511; 2007. p. 65110Y–65110Y–11. Available from: http://dx.doi.org/10.1117/12.709395.
2. Segars WP, Sturgeon G, Mendonca S, Grimes J, Tsui BMW. 4D XCAT phantom for multimodality imaging research. Medical Physics. 2010;37(9):4902–4915.
3. Maier A, Hofmann H, Schwemmer C, Hornegger J, Keil A, Fahrig R. Fast Simulation of X-ray Projections of Spline-based Surfaces using an Append Buffer. Physics in Medicine and Biology. 2012;57(19):6193–6210.
4. Parzen E. On Estimation of a Probability Density Function and Mode. The Annals of Mathematical Statistics. 1962;33(3):pp. 1065–1076.
5. Neumann D, Lugauer F, Bauer S, Wasza J, Hornegger J. Real-time RGB-D Mapping and 3-D Modeling on the GPU using the Random Ball Cover Data Structure. In: Fossati A, Gall J, Grabner H, Ren X, Konolige K, editors. IEEE International Conference on Computer Vision (ICCV) Workshops; 2011. p. 1161–1167.
6. Maier A, Hofmann H, Berger M, Fischer P, Schwemmer C, Wu H, et al. CONRAD - A Software Framework for Cone-Beam Imaging in Radiology. Medical Physics. 2013;40(11):111914–1–8.
7. Kak AC, Slaney M. Principles of Computerized Tomographic Imaging. Piscataway, NJ, United States: IEEE Service Center; 1988.