Technical Report

TR-2006-011

An OcTree Method for Parametric Image Registration

by

Eldad Haber, S. Heldmann, J. Modersitzki

Mathematics and Computer Science

EMORY UNIVERSITY

An OcTree Method for Parametric Image Registration

Eldad Haber*, Stefan Heldmann* and Jan Modersitzki†

June 9, 2006

Abstract

Already for reasonable sized 3D images, image registration becomes a computationally intensive task. Here, we introduce and explore the concept of OcTree's for registration which drastically reduces the number of processed data and thus the computational costs. We show how to map the registration problem onto an OcTree and present a suitable optimization technique. Furthermore, we demonstrate the performance of the new approach by academic as well as real life examples. These examples indicate that the computational time can be reduced by a factor of 10 compared with standard approaches.

1 Introduction

Image registration is one of today's challenging image processing problems. Given a so-called reference image R and a so-called template image T, the basic idea is to find a "*reasonable*" transformation such that a transformed version of the template image becomes "*similar*" to the reference image. Image registration has to be applied whenever images resulting from different times, devices, and/or perspectives need to be compared or integrated; see, e.g. [3, 15] and references therein.

*Department of Mathematics and Computer Science, Emory University, Atlanta, GA, USA.

†Institute of Mathematics, University of Lübeck, Germany.

The computation of parametric image registration has two main building blocks. The first one is a distance measure $\mathcal{D}$ quantifying distance or similarity of images. The second building block is a typically low dimensional parameterizable search space $\mathcal{V}$ for the desired transformation. For example, if $\mathcal{V}$ is a linear space, with a vector $\boldsymbol{\gamma}$ the collection of parameters and $\boldsymbol{\psi}_j$ some basis functions, any transformation can be expanded as

$$\boldsymbol{\Phi}(\boldsymbol{\gamma})(\mathbf{x}) := \textstyle\sum_j \gamma_j \, \boldsymbol{\psi}_j(\mathbf{x}).$$

The choice of the particular subspace can be thought of as a type of regularization, added to the problem to overcome ill-posedness; see, e.g., [15]. Typical examples include rigid, affine linear, or spline transformations [3].

In the following we model the images R and T as a smooth functions mapping from $\mathbb{R}^d$ to $\mathbb{R}$ (practically, these functions are obtained by interpolation of the given discrete 2D or 3D data). With some abuse of notation, the transformed image is denoted by $T(\boldsymbol{\gamma})$, where

$$T(\boldsymbol{\gamma})(\mathbf{x}) := T(\boldsymbol{\Phi}(\boldsymbol{\gamma})(\mathbf{x}))$$

A common treatment of parametric registration is based on the following minimization problem [21, 19]. Find a parameter $\boldsymbol{\gamma}^*$ minimizing

$$\mathcal{J}(\boldsymbol{\gamma}) := \mathcal{D}(R, T(\boldsymbol{\gamma})). \tag{1.1}$$

Generally, the optimization problem (1.1) cannot be solved analytically and thus numerical schemes are required. Although the number of unknowns $\boldsymbol{\gamma}$ might be small, computing the transformed image $T(\boldsymbol{\gamma})$ can be computationally expensive for 3D images. For example, the computation of $T(\boldsymbol{\gamma})$ for a 512^3 voxel image result in more than 10^8 interpolation points to be evaluated. Thus, efficient algorithms are required in order to perform the registration in a reasonable amount of time.

In order to keep the computational time reasonable, it is desirable to reduce the amount of data which has to be processed. A simple reduction can be obtained through down-sampling. However, naive down-sampling results in a loss of possibly important image features. Nevertheless, down-sampling can be safely done in parts of the image with low variations. A proper use of this observation can reduce the storage needed to represent the image and amount of computation needed for a registration substantially.

OcTrees provide a straightforward approach to such a context driven data sparsification. The OcTree data presentation uses few voxels for regions with small variability and many voxels for regions with high variability.

Although generating an OcTree from a known image is a simple task, solving the registration problem on OcTree's is far less trivial. The main difficulty is that a proper registration requires the OcTree representation of $T(\boldsymbol{\gamma}^*)$, where γ^* is a minimizer of (1.1). Because this minimizer is unknown a priori, we can not generate a reliable OcTree for $T(\boldsymbol{\gamma}^*)$. We overcome this problem by an updating of the current OcTree using only local refinement. Although these refinement steps require additional computations, still the procedure reduces the computational costs by one order of magnitude.

Using OcTrees in image processing is not new. Recent efforts have been made in the field of computer graphics [14, 13]. In particular, the work of Losasso et al on OcTree discretization demonstrates that images of fine-detail flows and smoke can be represented efficiently and reliably with this type of data structure. Other related work on motion estimation using QuadTree splines has been proposed in [20, 12]. Our approach is different extends the previous work to 3D and work with global transformation. Furthermore, we use techniques borrowed from numerical PDE's in order to efficiently solve the minimization problem obtained on the OcTree.

The paper is organized as follows. In Section 2, we discuss the mathematical framework and the OcTree discretization of the problem (1.1). In Section 3 we present an effective numerical optimizing procedure based on a Gauss-Newton type scheme. In Section 4, we propose a locally refined multilevel method for a solution of the nonlinear problem with self adjusting OcTree refinement. Finally, in Section 5, we present numerical examples for 2D and 3D that demonstrate the effectiveness of our algorithm.

The presented theory does not depend on a particular dimension and our implementation supports 2D and 3D images. However, for ease of presentation, we restrict our explanations to the 2D case and comment on 3D whenever an extension is not straightforward.

2 Mathematical Framework

In the literature we could not find a unique definition of an OcTree [11]. Here, we consider an OcTree as a non-uniform but still highly regular and sparse discretization of a domain $\Omega \subset \mathbb{R}^d$, where d denotes the spatial dimension. To this end, we define an OcTree as a collection of discretization points $\mathbf{x}_j$ and mesh-sizes h_j. Next, we present a formal description of OcTrees.

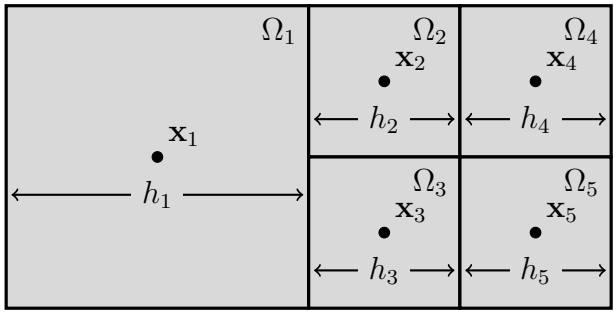


Figure 1: An OcTree discretization $\mathcal{S}_h$ in 2D

2.1 OcTree Definition

Let $\Omega = (a_1, b_1) \times (a_2, b_2) \times \ldots \times (a_d, b_d) \subset \mathbb{R}^d$ be a rectangular domain and $h \in \mathbb{R}$ such that $(b_j - a_j)/h \in \mathbb{N}$ for all j. We define an *OcTree discretization $\mathcal{S}_h$ of Ω with finest mesh-size h* as

$$\mathcal{S}_h = \{(\mathbf{x}_j, h_j) \,:\, j = 1, 2, \ldots, N\} \;\subset\; \Omega \times \{h \cdot 2^\ell \,:\, \ell \in \mathbb{N}\}$$

such that the sub-domains $B_j := B(\mathbf{x}_j, h_j) := \{\mathbf{y} \,:\, \|\mathbf{x}_j - \mathbf{y}\|_\infty < h_j/2\}$ essentially build a partition of Ω, i.e., $B_j \cap B_k = \emptyset$ for $j \neq k$, and $\bigcup_{j=1}^N \overline{B_j} = \overline{\Omega}$.

Furthermore, we make a few auxiliary definitions for neighborhood relations on OcTrees, regularity of OcTrees, and nested OcTrees, that will be useful in the following.

For $(\mathbf{x}_j, h_j) \in \mathcal{S}_h$ the set of *direct neighbors* is defined by

$$\mathrm{Adj}(\mathbf{x}_j, h_j) := \{(\mathbf{x}_k, h_k) \in \mathcal{S}_h \,:\, (\mathbf{x}_j, h_j) \neq (\mathbf{x}_k, h_k) \text{ and } \mathcal{H}^{d-1}(\overline{B}_j \cap \overline{B}_k) \neq 0\}.$$

where $\mathcal{H}^{d-1}$ denotes the $(d-1)$-dimensional surface measure.

An OcTree discretization $\mathcal{S}_h$ is called *regular of m-th order* if for all $(\mathbf{x}_j, h_j) \in \mathcal{S}_h$ it holds

$$2^{-m} \;\leq\; h_j/h_k \;\leq\; 2^m \qquad \text{for all direct neighbors } (\mathbf{x}_k, h_k) \in \mathrm{Adj}(\mathbf{x}_j, h_j).$$

In the following we refer to an OcTree $\mathcal{S}_h$ with the implicit understanding of a first order regular OcTree discretization of finest mesh-size h.

Note that a zeroth order regular OcTree $\mathcal{S}_h^0$ is a cell-centered equispaced discretization of Ω. In other words, $\mathcal{S}_h^0$ is the richest possible OcTree of finest mesh-sizes $h_j = h$ given by

$$\mathcal{S}_h^0 := \left\{(\mathbf{x}, h) \,:\, x_\ell = (a_\ell + h(k_\ell - \tfrac{1}{2})_{\ell=1}^d,\; k_\ell = 1, \ldots, \tfrac{b_\ell - a_\ell}{h}\right\}. \tag{2.2}$$

An OcTree discretization $\mathcal{S}_h$ is *nested* in an OcTree discretization $\mathcal{S}'_h$ if for all $(x, \eta) \in \mathcal{S}_h$ there exists a $(x', \eta') \in \mathcal{S}'_h$, such that

$$B(x, h) \cap B(x', h') = B(x, h).$$

The finest OcTree $\mathcal{S}_h^0$ is nested in any other OcTree discretization of Ω.

There are many possible data structures for a given OcTree. Here, the tree is stored as a sparse d dimensional array S. The relative size h_j/h of each cell is stored in the upper left corner of the cell, e.g., the OcTree shown in Figure 1 is represented by

$$S = \begin{pmatrix} 2 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}.$$

This allows us to use sparse matrix techniques in order to quickly find neighbors which is the major operation in the discretization process. The data structure is closely related to the one suggested in [11]. As a consequence, all that is needed to address a point of the OcTree is the sparse array S.

2.2 OcTree Approximation of Discrete Images

Consider a discrete 2D image with $m_1 \times m_2$ pixels. Assume that this image has large regions with low variability. Then such an image can be efficiently represented by an OcTree. An optimal OcTree is the one that would have the least amount of cells for the given image. However, such an OcTree is difficult to find (i.e. exponentially difficult) [11]. For practical purposes, we compute a sub-optimal OcTree by Algorithm 1. A similar algorithm is suggested in [11].

A central point is that the overall complexity of Algorithm 1 is $\mathcal{O}(N)$ where N is the number of pixels in the image. Furthermore, we added a tolerance parameter tol. It is used as a threshold to decide weather the image is almost constant in a certain region. In our application we have set tol to the estimated noise level in the image.

Although the number of pixels in the OcTree is image-dependent, we have found that for typical MRI and CT images the number of pixels is reduced to less than 20% of its original size in 2D and to less than one 10% its size in 3D, respectively. A representative 2D example, an OcTree discretization of a CT image is given in Figure 2. Table 1 shows compression rates and errors obtained for several tolerances.

Algorithm 1 Generate sparse array representation S of an OcTree for a 2D image T where regions with gray-value differences less than tol are treated as constant

$[S] \leftarrow$ `generateOcTreeFromImage`(T, tol)

set $[m_1, m_2] \leftarrow \text{size}(T)$

Initialize $S(1\!:\!m_1, 1\!:\!m_2) \leftarrow 1$ and $T_{\min} \leftarrow T$, $T_{\max} \leftarrow T$

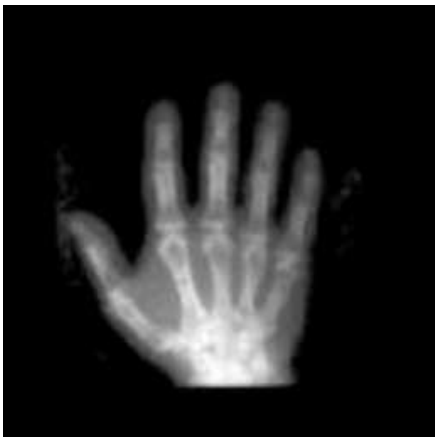
for $s = 2^{[0:\log_2(\min([m_1,m_2]))]}$

- find $(i_1, i_2) \in [1\!:\!s\!:\!m_1] \times [1\!:\!s\!:\!m_2]$ with $S(i_1, i_2) = s$
- get 4 sub-block indices of the super-block stored at (i_1, i_2) with size $2s$: $\mathcal{I} \leftarrow [i_1, i_1 + s] \times [i_2, i_2 + s]$
- compute minimum and maximum in the super-block: $T_{\min}(i_1, i_2) \leftarrow \min(T_{\min}(\mathcal{I}))$ and $T_{\max}(i_1, i_2) \leftarrow \max(T_{\max}(\mathcal{I}))$
- **if** $T_{\max}(i_1, i_2) - T_{\min}(i_1, i_2) \leq \text{tol}$ **then**
  set $S(i_1, i_2) = 2s$ and $S(\mathcal{I} \setminus (i_1, i_2)) = 0$,
  end if

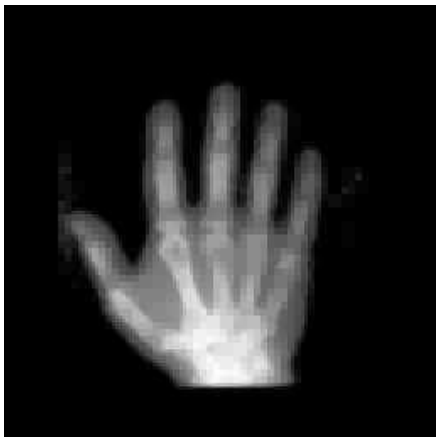
end for

tol	$\#\mathcal{S}_h$	$\#\mathcal{S}_h/\#\mathcal{S}_h^0$	average error	maximum error
0%	28,846	44.02%	0.00%	0.00%
1%	21,703	33.11%	0.02%	0.82%
2%	18,202	27.77%	0.06%	1.43%
5%	10,069	15.36%	0.27%	3.75%
10%	4,138	6.31%	0.59%	7.45%
15%	2,806	4.28%	0.77%	11.49%
20%	1,780	2.72%	1.02%	15.49%

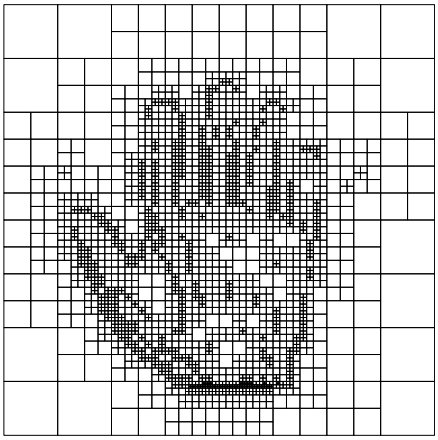
Table 1: OcTree-approximations $\widehat{T}$ of an image T with $N = 256^2$ pixels, cf. Figure 2(a), with scaled gray-values between 0 and 100 for various tolerances, $\#\mathcal{S}_h$ number of OcTree points, $\#\mathcal{S}_h/\#\mathcal{S}_h^0$ ratio between OcTree and full grid discretization points, average error $\frac{1}{N}\|T-\widehat{T}\|_1$, and maximum error $\|T-\widehat{T}\|_\infty$



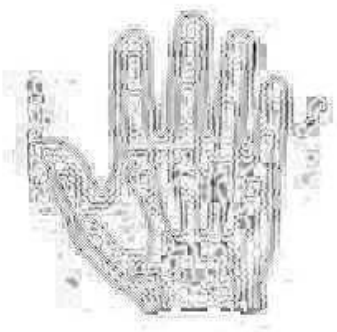
(a) original image T



(b) OcTree approximation $\widehat{T}$

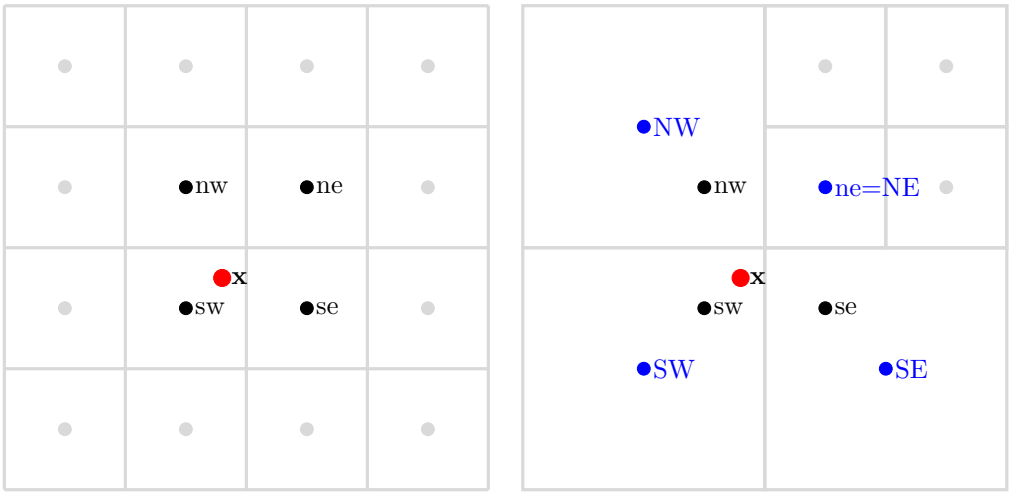


(c) OcTree



(d) absolute error $|T - \widehat{T}|$ (scaled)

Figure 2: Approximation of an image with an OcTree (cf. Table 1, tol=15%)



(a) equispaced grid $\mathcal{S}_h^0$ (b) OcTree $\mathcal{S}_h^1$

Figure 3: Linear interpolation on $\mathcal{S}_h^0$ (a) and $\mathcal{S}_h^1$ (b)

2.3 OcTree Image Interpolation

For the minimization of (1.1) we must be able to evaluate R and $T(\boldsymbol{\gamma})$ where we model images as smooth functions. Typically, an image comes in digital form as discrete data and we construct a continuous function by interpolation. Here, we consider the case of linear interpolation where the data is given on an OcTree. We start our considerations on the full grid $\mathcal{S}_h^0$.

Consider a pixel located at a point $\mathbf{x}$ which is not a grid point and we are interested in the value $T(\mathbf{x})$. To compute this value we use a linear interpolation scheme. In the 2D case, let nw, sw, ne, and se denote the centers of the four adjacent pixels of $\mathbf{x} = (x_1, x_2)$, cf. Figure 3. Then, defining $\xi_j = x_j/h - \lfloor x_j/h \rfloor \in [0, 1]$ we have

$$T(\mathbf{x}) \approx \xi_2\Big(T(\text{nw})\xi_1 + T(\text{sw})(1-\xi_1)\Big) + (1-\xi_2)\Big(T(\text{ne})\xi_1 + T(\text{se})(1-\xi_1)\Big). \quad (2.3)$$

Now, assume that we have an OcTree discretization as plotted in Figure 3(b). In this case we have the OcTree values of the pixel in SW, SE, NW, and NE=ne, but we do not have values in nw, sw, and se. Nevertheless, since we assume that the image is almost constant over each of the larger cells we can use the prescribed value of the larger cells instead of the values of the original small

cell:

$$T(\mathrm{sw}) \approx T(\mathrm{SW}), \quad T(\mathrm{se}) \approx T(\mathrm{SE}), \quad T(\mathrm{nw}) \approx T(\mathrm{NW}), \quad T(\mathrm{ne}) \approx T(\mathrm{NE}). \tag{2.4}$$

Finally, formula (2.3) is used for interpolation.

Lemma 1 *Let $T \in C^2(\mathbb{R}^d, \mathbb{R})$ be two times continuously differentiable. The interpolation* (2.3) *with function value substitution* (2.4) *is of order $\mathcal{O}(h)$.*

Proof: We present the proof for $d = 1$, the extension to higher dimensions is straightforward.

Let x be an arbitrary point which lies between two gridpoints x^- and $x^+ = x^- + h$ such that $x^- \leq x \leq x^+$ and $\xi := x/h - \lfloor x/h \rfloor \in [0, 1]$. From classical interpolation we have

$$T(x) = T(x^- + h\xi) = T(x^-)(1 - \xi) + T(x^+)\xi + \mathcal{O}(h^2).$$

Perturbing for example the first control point by h, the statement follows from $T(x^- - h) = T(x^-) + \mathcal{O}(h)$ and $0 \leq 1 - \xi < 1$. The proof for the perturbed second control point is along the same lines. ■

A direct consequence of this interpolation scheme is summarized in the next corollary. This corollary suggests to pick a non-vanishing tolerance in Algorithm 1 even if the image data is not perturbed by noise.

Corollary 1 *Let $\mathcal{S}_h^0$ be an equispaced and $\mathcal{S}_h^m$ be a m-th order regular OcTree discretizations of a domain $\Omega \subset \mathbb{R}^d$. For an image $T \in C^2(\overline{\Omega})$ let T^0 and T^m denote the linear interpolations based on the discretization points from the OcTrees $\mathcal{S}_h^0$ and $\mathcal{S}_h^m$, respectively, and corresponding function values. Then*

$$|T^m(\mathbf{x}) - T^0(\mathbf{x})| = \mathcal{O}(h).$$

2.4 OcTree Based Distance Measures

The OcTree based distance measure approximation is exemplarily discussed for the L_2-norm based sum of squared differences distance measure [3]. Here,

$$\begin{aligned}
\mathcal{D}(R, T(\boldsymbol{\gamma})) &= \frac{1}{2}\|T(\boldsymbol{\gamma}) - R\|^2_{L_2(\Omega)} && (2.5)\\
&= \frac{1}{2}\int_\Omega \big(T(\boldsymbol{\gamma})(\mathbf{x}) - R(\mathbf{x})\big)^2 \, d\mathbf{x} \\
&\approx \frac{1}{2}\sum_{(\mathbf{x}_j, h_j)\in\mathcal{S}_h^0} h_j^d \big(T(\boldsymbol{\gamma})(\mathbf{x}_j) - R(\mathbf{x}_j)\big)^2 && (2.6)\\
&\approx \frac{1}{2}\sum_{(\mathbf{x}_j, h_j)\in\mathcal{S}_h^1} h_j^d \big(T(\boldsymbol{\gamma})(\mathbf{x}_j) - R(\mathbf{x}_j)\big)^2 && (2.7)\\
&= \frac{1}{2}\|T(\boldsymbol{\gamma}) - R\|^2_H =: D(\mathcal{S}_h, R, T(\boldsymbol{\gamma})) && (2.8)
\end{aligned}$$

with a weighted l_2 norm, where the diagonal weighting matrix is given by $H = \operatorname{diag}(h_j^d)$ with the cell sizes h_j^d. where (2.6) is the standard midpoint quadrature rule and (2.7) is its approximation on a first order OcTree. Note that $h_j = h$ in (2.6) whereas h_j can vary in (2.7). In (2.8), we make the discretized distance measure explicitly depended on the OcTree. In our implementation, we use the OcTree generated for the transformed image $T(\boldsymbol{\gamma})$. Therefore, it is important to note that generally an OcTree discretization depends on $\boldsymbol{\gamma}$, i.e., $\mathcal{S}_h = \mathcal{S}_h(\boldsymbol{\gamma})$.

3 Optimization

Ideally, our aim is to minimize

$$J(\boldsymbol{\gamma}) := D(\mathcal{S}_h^0, R, T(\boldsymbol{\gamma})), \tag{3.9}$$

where $\mathcal{S}_h^0$ is the full equispaced grid (2.2). To do so efficiently, we replace this problem by a problem on a sparse and computationally less involved OcTree discretization. A straightforward approach is to substitute (3.9) by

$$J(\boldsymbol{\gamma}) := D(\mathcal{S}_h(\boldsymbol{\gamma}), R, T(\boldsymbol{\gamma})). \tag{3.10}$$

such that a minimizer $\boldsymbol{\gamma}^*$ leads to a transformed image $T(\boldsymbol{\gamma}^*)$ for which (2.6) can be replaced by (2.7) up to a small error.

One approach to minimize (3.10) is to consider $\boldsymbol{\gamma} = \boldsymbol{\gamma}(t)$ as a time dependent function and to embed the problem into a time stepping process such as the dynamical system

$$\boldsymbol{\gamma}_t = -\nabla_{\boldsymbol{\gamma}} J(\boldsymbol{\gamma}) = -\nabla_{\boldsymbol{\gamma}} D(\mathcal{S}_h(\boldsymbol{\gamma}), R, T(\boldsymbol{\gamma})).$$

Then, one can use methods that has been developed for adaptive mesh refinement for time-dependent problems when integrating to steady state. At each step one estimates both $\boldsymbol{\gamma}$ and the OcTree $\mathcal{S}_h(\boldsymbol{\gamma})$. However, such an approach is inefficient in the context of parametric registration since we are not interested in the dynamics of the process and evaluating the OcTree at each time step is not necessary.

Since the OcTree in (3.10) depends on $\boldsymbol{\gamma}$, the minimization is far away from being trivial. Therefore, we propose a two phase procedure similar to coordinate search. Starting with $k = 0$ and an initial parameter vector $\boldsymbol{\gamma}^{(k)}$, the first step is to compute the corresponding OcTree $\mathcal{S}_h^{(k)} := \mathcal{S}_h(\boldsymbol{\gamma}^{(k)})$. The next step is to compute a minimizer $\boldsymbol{\gamma}^{(k+1)}$ of

$$J^{(k)}(\boldsymbol{\gamma}) := D(\mathcal{S}_h^{(k)}, R, T(\boldsymbol{\gamma}))$$

for the (fixed) OcTree $\mathcal{S}_h^{(k)}$. These two steps are repeated to convergence. The strategy is very similar to refinement strategies used for adaptive grid problems in Ordinary Differential Equations and Partial Differential Equations; see, e.g. [1, 18, 2] and reference therein. The algorithm is summarized in Algorithm 2.

A crucial point in our method is to ensure decrease towards a solution on the full grid. This is done by a backtracking line search.

Assume that $\mathcal{S}_h^{(k)}$ is an OcTree valid for $\boldsymbol{\gamma}^{(k)}$ and $\widehat{\boldsymbol{\gamma}}$ is a minimizer of $J^{(k)}$. A naive approach is to continue the iteration by setting $\boldsymbol{\gamma}^{(k+1)} := \widehat{\boldsymbol{\gamma}}$. However, this procedure may not guarantee convergence to a minimizer of 3.9. In fact, while $\widehat{\boldsymbol{\gamma}}$ minimizes $J^{(k)}$ it may increase the value of J. We therefore use a (weak) line search for computing the new transformation and the new OcTree. Algorithm 2 implies that we use the OcTree in order to compute a search direction $\delta\boldsymbol{\gamma}$ and then use this direction to decrease the objective function J, i.e., $D(\mathcal{S}_h^0, \cdot\,)$ on the full grid. The obvious question whether $\delta\boldsymbol{\gamma}$ computed on a coarse OcTree leads to a descent direction for the full OcTree is positively answered in the literature [16, 9].

Finally, we have two natural stopping criteria. First, if $\mathcal{S}_h^{(k+1)} = \mathcal{S}_h^{(k)}$, we set $\boldsymbol{\gamma}^* \leftarrow \boldsymbol{\gamma}^{(k)}$ and stop. Second, if we are unable to decrease the value of the objective function in the direction $\delta\boldsymbol{\gamma}$ then we set $\boldsymbol{\gamma}^* \leftarrow \boldsymbol{\gamma}^{(k)}$ and stop.

Algorithm 2 Outer iteration to minimize $J(\boldsymbol{\gamma}) := D(\mathcal{S}_h^0, R, T(\boldsymbol{\gamma}))$

$\boldsymbol{\gamma}^* \leftarrow \texttt{minimizeJ}(T, R, \boldsymbol{\gamma}^{(0)}, h)$

while not converged

- for given $\boldsymbol{\gamma}^{(k)}$ generate $\mathcal{S}_h^{(k)}$ such that $D(\mathcal{S}_h^{(k)}, \boldsymbol{\gamma}^{(k)}) \approx J(\boldsymbol{\gamma}^{(k)})$
- compute a minimizer $\widehat{\boldsymbol{\gamma}}$ of $J^{(k)} := D(\mathcal{S}_h^{(k)}, \cdot\,)$
- set $\delta\boldsymbol{\gamma} \leftarrow \widehat{\boldsymbol{\gamma}} - \boldsymbol{\gamma}^{(k)}$
- perform line search on $\mathcal{S}_h^0$: find λ such that $J(\boldsymbol{\gamma}^{(k)} + \lambda\delta\boldsymbol{\gamma}) < J(\boldsymbol{\gamma}^{(k)})$
- update $\boldsymbol{\gamma}^{(k+1)} \leftarrow \boldsymbol{\gamma}^{(k)} + \lambda\delta\boldsymbol{\gamma}$,

end while
return $\boldsymbol{\gamma}^* \leftarrow \boldsymbol{\gamma}^{(k+1)}$.

3.1 Solving for $\boldsymbol{\gamma}$ on an OcTree

To compute a minimizer $\widehat{\boldsymbol{\gamma}}$ of $D(\mathcal{S}_h, \cdot\,)$ on a given fixed OcTree $\mathcal{S}_h$ we use a Gauss-Newton like method. Since the OcTree is constant in this step, we may rewrite the distance function (2.8) by

$$D(R, T(\boldsymbol{\gamma})) = D(\mathcal{S}_h, R, T(\boldsymbol{\gamma}))$$

To use the Gauss-Newton method, we first linearize $T(\boldsymbol{\gamma})$:

$$T(\boldsymbol{\gamma} + \delta\boldsymbol{\gamma}) \approx T(\boldsymbol{\gamma}) + T_{\boldsymbol{\gamma}}(\boldsymbol{\gamma})\ \delta\boldsymbol{\gamma}$$

where $T_{\boldsymbol{\gamma}}$ is the Jacobian matrix of T with respect to $\boldsymbol{\gamma}$. We then linearizing D,

$$D(R, T(\boldsymbol{\gamma} + \delta\boldsymbol{\gamma})) \approx D(R, T(\boldsymbol{\gamma}) + T_{\boldsymbol{\gamma}}(\boldsymbol{\gamma})\delta\boldsymbol{\gamma})$$

Differentiating with respect to $\delta\boldsymbol{\gamma}$ we obtain the following equation which can be solved for $\delta\boldsymbol{\gamma}$

$$T_{\boldsymbol{\gamma}}(\boldsymbol{\gamma})^\top\ D_T(R, T(\boldsymbol{\gamma}) + T_{\boldsymbol{\gamma}}(\boldsymbol{\gamma})\delta\boldsymbol{\gamma}) = 0 \tag{3.11}$$

where D_T is the gradient of the distance measure w.r.t. T.

For the example, for the SSD distance measure discussed in Section 2.4, $D(R, T(\boldsymbol{\gamma})) = \frac{1}{2}\|T(\boldsymbol{\gamma}) - R\|_H^2$, we end up with the usual Gauss-Newton scheme [5], where $\delta\boldsymbol{\gamma}$ is obtained by solving the system

$$T_{\boldsymbol{\gamma}}^{\top} H(T_{\boldsymbol{\gamma}}\, \delta\boldsymbol{\gamma} + R - T) = 0$$

However, for more complicated distance measures (e.g. mutual information [4, 22] or normalized gradient fields [8]) an inexact solution of this system can be obtained using an inexact Newton's method.

After solving for $\delta\boldsymbol{\gamma}$ we update $\boldsymbol{\gamma}$ using a weak line search which insures sufficient reduction in the value of the objective function [17]. The algorithm is summarized in 3.

Algorithm 3 Inner iteration to minimize $D(\mathcal{S}_h, R, T(\boldsymbol{\gamma}))$ w.r.t. $\boldsymbol{\gamma}$

$\widehat{\boldsymbol{\gamma}} \leftarrow$ `minimizeJonFixedOcTree`$(\mathcal{S}_h, R, T, \boldsymbol{\gamma})$

while not converge

- compute $T(\boldsymbol{\gamma})$ and $T_{\boldsymbol{\gamma}}(\boldsymbol{\gamma})$
- approximately solve equation (3.11) for $\delta\boldsymbol{\gamma}$
- perform line search:
  find λ such that $D(\mathcal{S}_h, R, T(\boldsymbol{\gamma} + \lambda\delta\boldsymbol{\gamma})) < D(\mathcal{S}_h, R, T(\boldsymbol{\gamma}))$
- update $\boldsymbol{\gamma} \leftarrow \boldsymbol{\gamma} + \lambda\delta\boldsymbol{\gamma}$

end while
return $\widehat{\boldsymbol{\gamma}} \leftarrow \boldsymbol{\gamma}$

3.2 Generating an OcTree for $\boldsymbol{\gamma}$

In the previous subsection we discussed how to compute a minimizer $\boldsymbol{\gamma}$ for a particular given OcTree. In this subsection we assume that $\boldsymbol{\gamma}$ is given and we discuss how to generate an appropriate OcTree.

The key for the computation of a new OcTree is that the transformed image must be well-represented on the grid, i.e., we need to sufficiently estimate the distance function $\mathcal{D}(R, T(\boldsymbol{\gamma}))$. In order to be more specific, in

this subsection we assume that $\mathcal{D}$ is the sum of squared difference distance measure, cf. Section 2.4.

Since we try to approximate the integral of the difference between $T(\boldsymbol{\gamma})$ and R the OcTree has to be constructed based on both, $T(\boldsymbol{\gamma})$ and R. Therefore, we first evaluate $T(\boldsymbol{\gamma})$ and R on the full grid. We then use Algorithm 1 to generate OcTrees $\mathcal{S}_h^T$ and $\mathcal{S}_h^R$ for $T(\boldsymbol{\gamma})$ and R. Finally, we join the OcTrees. As a result, we obtain a OcTree nested in both, $\mathcal{S}_h^T$ and $\mathcal{S}_h^R$, cf. Section 2.1. The procedure is summarized in Algorithm 4.

Algorithm 4 Generate sparse OcTree representation S (2D)

$S \leftarrow$ `generateJointOcTree`$(T, R, \boldsymbol{\gamma})$

- compute $T(\boldsymbol{\gamma})$ and R on the full grid Ω_h
- compute an OcTree representations S_T and S_R for $T(\boldsymbol{\gamma})$ and R using Algorithm 1
- join S_T and S_R, i.e.,
  - find (i_1, i_2) where $S_R(i_1, i_2) \neq 0$ and $S_T(i_1, i_2) \neq 0$ and set $S(i_1, i_2) \leftarrow \min\{S_R(i_1, i_2), S_T(i_1, i_2)\}$
  - find (i_1, i_2) where $S_R(i_1, i_2) \neq 0$ and $S_T(i_1, i_2) = 0$ and set $S(i_1, i_2) \leftarrow S_R(i_1, i_2)$
  - find (i_1, i_2) where $S_R(i_1, i_2) = 0$ and $S_T(i_1, i_2) \neq 0$ and set $S(i_1, i_2) \leftarrow S_T(i_1, i_2)$

4 Multilevel Continuation

To overcome nonlinearities and accelerate computations we use a multilevel strategy. A multilevel method for image registration is not new [6, 7, 10]. However, modification of the standard algorithm is required to effectively use the OcTree grid. Our multilevel algorithm uses the OcTree structure directly in order to gradually refine the grid only in areas that a refinement is required.

We start on a coarsest OcTree with finest mesh-size $H = 2^L h$, which is a few multiples of the smallest mesh-size h. We then solve the optimization

Algorithm 5 Multilevel solve for $\boldsymbol{\gamma}$

$[\boldsymbol{\gamma}] \leftarrow \texttt{MLSolve}(T, R, \boldsymbol{\gamma}, h, L)$

set $H \leftarrow 2^L h$

initialize $\boldsymbol{\gamma}$ and S_H

while $H > h$

- use Alg. 2 with initial guess $\boldsymbol{\gamma}_{2H}$ to find $\boldsymbol{\gamma}_H$ such that $D(\mathcal{T}_H^0, \boldsymbol{\gamma}_H) = \min$: $\boldsymbol{\gamma}_H \leftarrow \texttt{minimizeJ}(T, R, \boldsymbol{\gamma}_{2H}, H)$
- set $H \leftarrow H/2$

end while

problem using Algorithm 2 to obtain a coarse grid solution $\boldsymbol{\gamma}_H$. We then move to the next finer level. We set the finest mesh-size H to $H/2$ and use the coarse grid solution $\boldsymbol{\gamma}_H$ as an initial guess to the optimization problem for the finest mesh-size $H/2$. We repeat this procedure until we arrive the smallest grid size h. The algorithm is summarized in Algorithm 5.

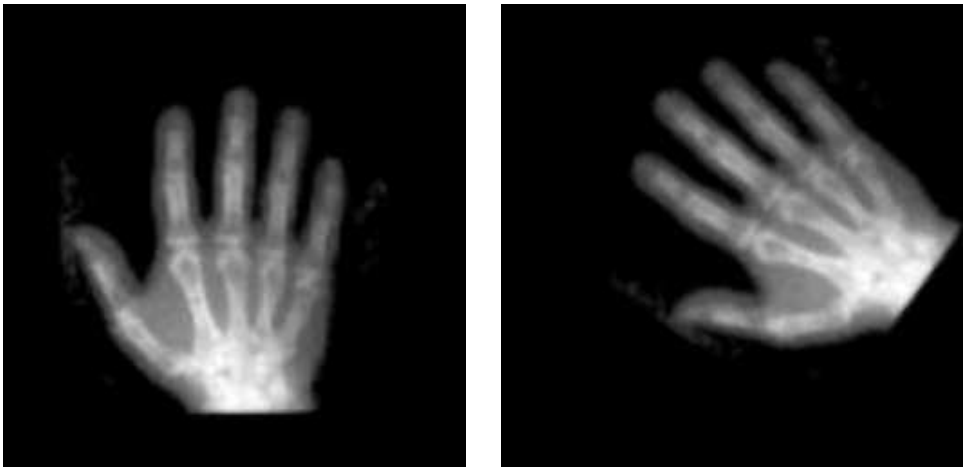
It is important to note that combining our multilevel strategy to the optimization strategy naturally results in only *local grid refinement*. This is because the optimization on every level starts with an *OcTree* of the refined previous previous solution and this automatically refines the grid only in high variability areas.

5 Examples

In this Section we demonstrate the effectiveness of our approach on a few examples. We use a 2D example which can be easily visualize and a 3D example to further demonstrate the computational saving when solving registration problems in 3D. In both examples we use the SSD distance measure and optimize w.r.t. affine transformation.

5.1 2D Registration of Hand CT

Our first numerical experiment is matching 2D images of hands plotted in Figure 4. The multilevel optimization process is summarized in Table 2 and



(a) R (b) T

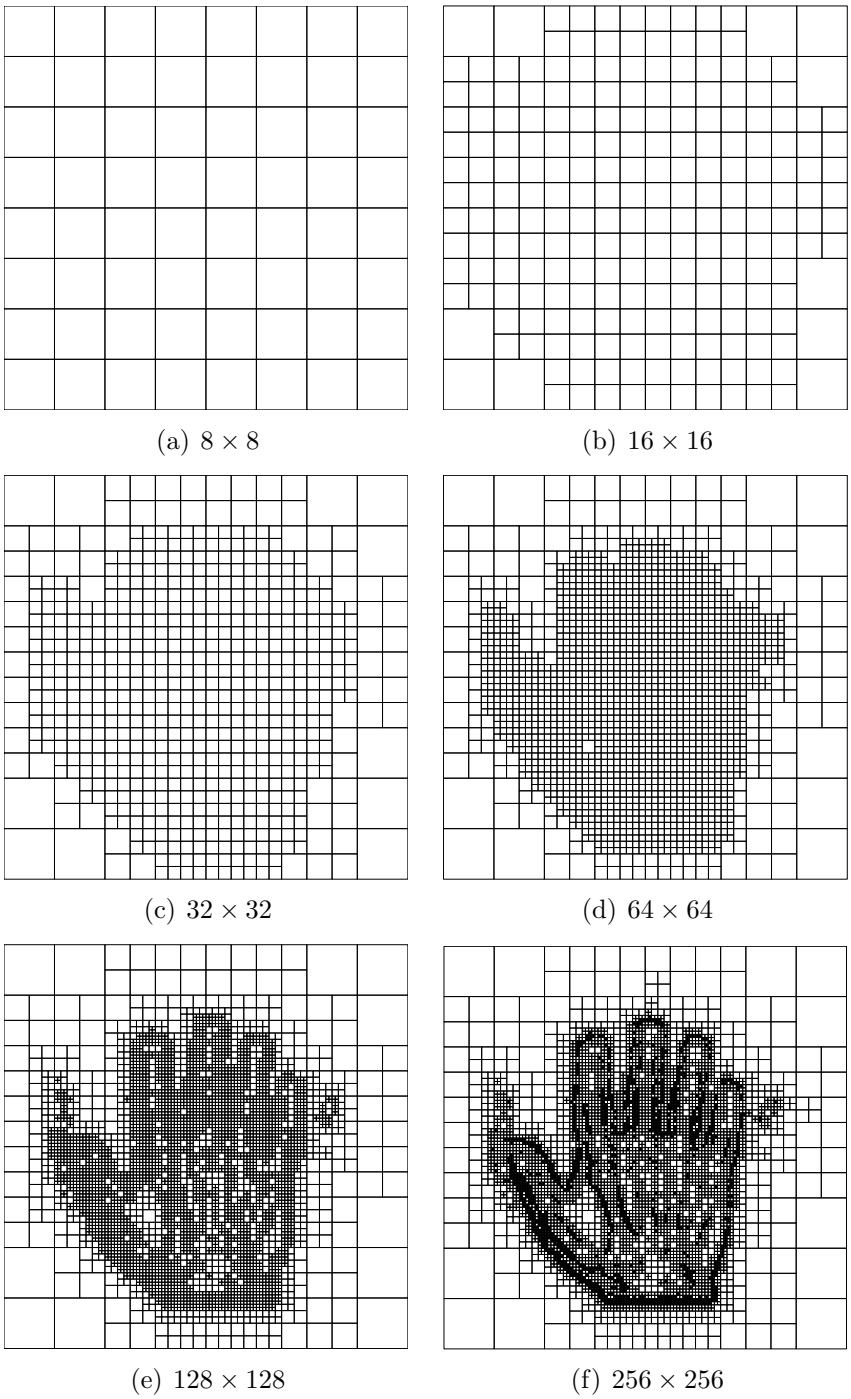
Figure 4: X-ray images used for 2D registration example

Level	$\mathcal{T}_h^0$	Outer Iterations	Inner Iterations
1	8×8	2	15
2	16×16	2	8
3	32×32	2	5
4	64×64	2	4
5	128×128	2	3
6	256×256	2	3

Table 2: Optimization path for the 256×256 2D problem

a few OcTree grids obtained in the process of optimization are plotted in Figure 5.

For this problem, the gray values of the image were in the range $[0, 750]$ and we used a tolerance parameter of 50, which is $\approx$6.5% of the gray value dynamics, to obtain the OcTrees. As a result, the average and maximum error of the images was $\approx$0.4% and $\approx$5%, respectively. On the finest level, the number of cells of the OcTrees were $\approx$13% of a full equispaced discretization. In comparing our code the a non-OcTree multilevel optimization we observed a speed up in wall clock time in about a factor of roughly 5.

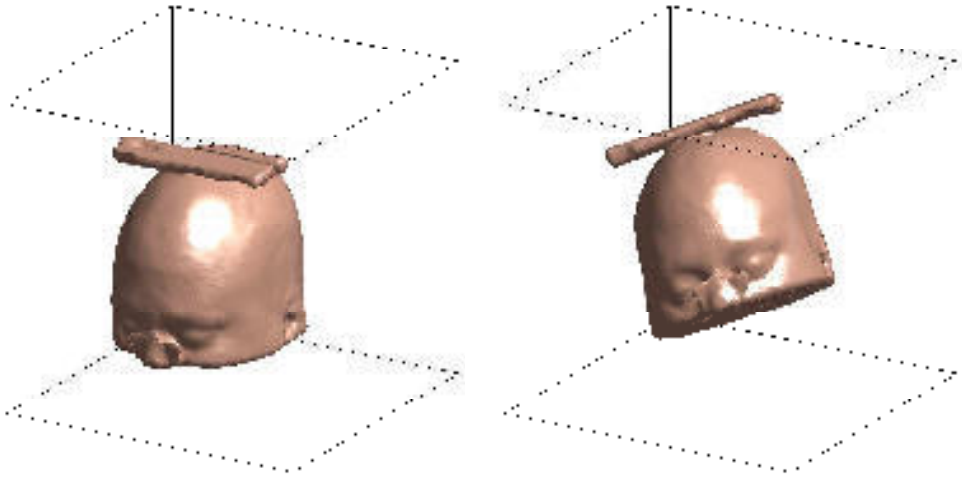


(a) 8×8 (b) 16×16

(c) 32×32 (d) 64×64

(e) 128×128 (f) 256×256

Figure 5: OcTrees used on different levels



(a) R (b) T

Figure 6: MRI images used in the 3D example

Level	$\mathcal{T}_h^0$	Outer Iterations	Inner Iterations
1	$8 \times 8 \times 8$	3	27
2	$16 \times 16 \times 16$	3	17
3	$32 \times 32 \times 32$	2	5
4	$64 \times 64 \times 64$	3	6
5	$128 \times 128 \times 128$	2	4
6	$256 \times 256 \times 256$	2	3

Table 3: Optimization path for the $256 \times 256 \times 256$ 3D problem

5.2 3D Registration of Head MRI

In our second experiment we register 3D MRI images of the head. The images were taken from the Brain-web database and interpolated to a size of $256 \times 256 \times 256$ voxels. As template we rotated the reference by 20° around the center of the image in x-, y-, and z-direction and finally made a diagonal shift by 20 pixels. The reference and template are shown in Figure 6(a) and Figure 6(b), respectively. For the computation we generated OcTree representations of the images with a tolerance of 20 which is ≈8% of the gray value dynamic of the images. Then number of cells of the generated OcTree discretizations were about ≈8% of a full equispaced discretization.

The average errors of the images were ≈0.5 which is ≈0.2% of the gray value dynamic and the maximum errors assumed nearly the tolerance of value 20 .

6 Summary and Final Comments

In this paper we have developed a variational OcTree image registration technique. We have demonstrated that using such a technique we are able to substantially accelerate parametric image registration problems.

While we observe major computational savings for parametric registration we expect to have much better results for non-parametric registration where large sparse linear systems are solved for the displacement.

References

[1] U. Ascher, R. Mattheij, and R. Russell. *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations*. SIAM, Philadelphia, 1995.

[2] Marsha Berger. *Adaptive Mesh Refinement for Time-Dependent Partial Differential Equations*. Ph.d. dissertation, Stanford University, 1982. Computer Science Report No. STAN-CS-82-924.

[3] L.G. Brown. A survey of image registration techniques. *Surveys*, 24(4):325–376, December 1992.

[4] A. Collignon, A. Vandermeulen, P. Suetens, and G. Marchal. multi-modality medical image registration based on information theory. *Kluwer Academic Publishers: Computational Imaging and Vision*, 3:263–274, 1995.

[5] J. E. Dennis and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. SIAM, Philadelphia, 1996.

[6] M. Droske and M. Rumpf. A variational approach to non-rigid morphological registration. *SIAM J. Appl. Math.*, 64(2):668–687, 2004.

[7] E. Haber and J. Modersitzki. Multilevel methods for image registration. *SIAM J. on Scientific Computing*, To Appear:xxx, 2004.

[8] E. Haber and J. Moderzitski. Intensity gradient based registration and fusion of multi-modal images. *Submitted*, 2004.

[9] W. Hackbusch and A. Reusken. Analysis of a damped nonlinear multilevel method. *Numer. Math.*, 55:225–246, 1989.

[10] S. Henn and K. Witsch. Multimodal image registration using a variational approach. *SIAM J. Sci. Comp.*, 25:1429–1447, 2003.

[11] G. R. Hjaltason and H. Samet. Speeding up construction of quadtrees for spatial indexing. *The VLDB Journal*, 11:109–137, 2002.

[12] S. Kruger and A. Calway. Image registration using multiresolution frequency domain correlation. *British Machine Vision Conference*, September:316–325, 1998.

[13] F. Losasso, R. Fedkiw, and S. Osher. Spatially adaptive techniques for level set methods and incompressible flow. *Computers and Fluids*, 35:457–462, 2006.

[14] F. Losasso, F. Gibou, and R. Fedkiw. Simulating water and smoke with an octree data structure. *SIGGRAPH*, 23:457–462, 2004.

[15] J. Modersitzki. *Numerical methods for image registration*. Oxford, 2004.

[16] S.G. Nash. A multigrid approach to discretized optimization problems. *Optimization Methods and Software*, 14:99–116, 2000.

[17] J. Nocedal and S. Wright. *Numerical Optimization*. New York: Springer, 1999.

[18] L.R. Petzold. Observations on an adaptive moving grid method for one-dimensional systems of partial differential equations. *Applied Numerical Mathematics*, 3:347–360, 1987.

[19] J.P.W. Pluim, J.B.A. Maintz, and M.A. Viergever. Mutual-information-based registration of medical images: a survey. *IEEE Transactions on Medical Imaging*, 22:986–1004, 1999.

[20] R. Szeliski and H.Y. Shum. Motion estimation with quadtree splines. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18, Issue 12:1199 – 1210, 1996.

[21] P. Thevenaz and M. Unser. Optimization of mutual information for multiresolution image registration. *IEEE Trans. Image Processing*, v 9:2083–2089, 2000.

[22] Paul A. Viola. *Alignment by maximization of mutual information*. PhD thesis, Massachusetts Institute of Technology, 1995.