Classification Via Image Deformation: Theory and Implementation Details

David P. Williams[1] and Vincent L. Myers

Work performed at
NATO Undersea Research Centre
La Spezia, Italy

August 2006

Abstract

This document is an excerpted version of the full document titled "Automatic Target Recognition for Sonar Imagery Via Image Deformation." It contains the relevant theory from the full paper regarding an image deformation algorithm and Gaussian processes.

I. Image Deformation

A. Introduction

Significant work has been conducted in the medical imaging community on image registration [5], [8], [13], [19] and image segmentation [10], [12] via non-rigid image deformation. Many studies have dealt with aligning body structures in images generated from different modalities — such as magnetic resonance (MR), computed tomography (CT), positron emission tomography (PET), and ultrasound images [13]. Other studies have focused on aligning body structures — and in particular the head or brain — in a patient's images with model template images, or *atlases*, to aid in detecting abnormalities [13]. To the best of our knowledge, the work we present here is the first to apply image deformation techniques to sonar image classification tasks.

An algorithm that permits large-scale deformations will be necessary for our application because the deformation will be applied to pairs of sonar images that may be highly dissimilar. Elastic material models [17], in contrast, are restricted to small deformations because severe penalties discourage large deformations. These elastic models may be sufficient in applications such as brain imaging in which only minor perturbations are required, but they are not appropriate for our purposes. Other approaches that linearize nonlinear models [2] effectively assume that the deformation is small, which again conflicts with our intent. Additionally, since our goal is to develop a fully autonomous algorithm, approaches that require a human to specify landmark points or to label regions in the image must also be avoided. The viscous fluid model we employ in this work can account for large-scale deformations while preserving the topology of structures in the original image. Moreover, no human assistance is required.

B. Theory

In [3], a simplified Navier-Poisson viscous fluid model that governs a non-rigid image deformation process is derived. Specifically, the model explains how particles — which in this case are the image pixels — "flow" in the image during a deformation process that attempts to transform a *template* image, $\mathbf{T}$, into a *study* image, $\mathbf{S}$. That is, the *velocity*, $\boldsymbol{v}$, of each pixel and its resulting *displacement*, $\boldsymbol{u}$, (from its initial location in the original template image) are determined via this fluid model. This flow of pixels is driven by a set of *body forces*, $\boldsymbol{b}$, acting on the image pixels.

The interplay of these quantities to deform a template image into a study image is expressed via a modified Navier-Stokes equation,

$$\mu\nabla^2\boldsymbol{v}(x,y) + (\lambda+\mu)\nabla(\nabla\cdot\boldsymbol{v}(x,y)) + \boldsymbol{b}(x,y) = \mathbf{0}, \tag{1}$$

where (x,y) indicates a specific pixel location in the image, and $\nabla^2 = \nabla^T\nabla$ is the Laplacian operator. In (1), μ and λ are the Lamé constants that control the relative amount of constant-volume viscous flow and the propensity of a volume to grow or shrink. The velocity at each (x,y) pixel location in the image are the quantities for which (1) must be solved.

[1]Author to whom correspondence should be addressed, at DavidWilliams.PhD@gmail.com.

Explicitly separating the x and y direction components from the partial differential equation in (1), one obtains

$$\mu\left[\frac{\partial^2 v_x(x,y)}{\partial x^2}+\frac{\partial^2 v_x(x,y)}{\partial y^2}\right]+(\lambda+\mu)\left[\frac{\partial^2 v_x(x,y)}{\partial x^2}+\frac{\partial^2 v_y(x,y)}{\partial x\partial y}\right]+b_x(x,y)=0 \tag{2}$$

$$\mu\left[\frac{\partial^2 v_y(x,y)}{\partial x^2}+\frac{\partial^2 v_y(x,y)}{\partial y^2}\right]+(\lambda+\mu)\left[\frac{\partial^2 v_x(x,y)}{\partial x\partial y}+\frac{\partial^2 v_y(x,y)}{\partial y^2}\right]+b_y(x,y)=0. \tag{3}$$

In fluid mechanics, two different representations can be used to describe how particles flow [14]. In a Lagrangian representation, the trajectories of specific fluid particles are observed. In an Eulerian representation, the fluid velocity at fixed positions are observed. As a result of employing the Eulerian framework in this work, the velocity $\boldsymbol{v}$ is defined in terms of a convective derivative as

$$\boldsymbol{v}(x,y)=\frac{\partial \boldsymbol{u}(x,y)}{\partial t}+\left(\left[\boldsymbol{v}(x,y)\right]^T\nabla\right)\boldsymbol{u}(x,y). \tag{4}$$

The second term on the right-hand side of (4) accounts for nonlinearities of the displacement $\boldsymbol{u}$, which in turn permit large-scale deformations of the image.

The body forces drive the deformation process that deforms the template image into the study image. The body force at the image location (x,y) at time t is defined to be

$$\boldsymbol{b}^{(t)}(x,y)=-\alpha\left(\mathbf{T}(x',y')-\mathbf{S}(x,y)\right)\left(\nabla\mathbf{T}(x',y')\right) \tag{5}$$

where

$$(x',y')=(x,y)-(u_x^{(t)}(x,y),u_y^{(t)}(x,y)) \tag{6}$$

accounts for the deformations that have transpired up until time t, and α is a scalar that controls the relative magnitudes of the body forces. It can readily be seen from (5) that the body forces act to encourage the template and study images to match. It should also be noted that the body forces have a nonlinear dependence on the displacement $\boldsymbol{u}$.

In summary, body forces manifested by a difference between the template and study images drive a system of partial differential equations that is solved to obtain the velocities at which each pixel in the image deforms. This process is conducted iteratively until the deformed template image matches the study image.

C. Implementation

This section provides implementation details for the image deformation algorithm in Section I-B. It should be noted that the partial differential equations in (2) and (3) are nonlinear with respect to the displacement $\boldsymbol{u}$, but linear with respect to the velocity $\boldsymbol{v}$. Therefore, we solve the equations for the velocity $\boldsymbol{v}$, and then subsequently solve for the displacement $\boldsymbol{u}$ using the relationship in (4).

1) Finite Difference Method: To solve the partial differential equations in (2) and (3), the spatial derivatives are first discretized using a second-order central difference finite difference method [16]. The result of this discretization, with equal spatial step-sizes, h, in the x and y directions, is

$$\begin{aligned}&\left(\frac{\mu}{h^2}\right)\left[v_x(x,y+h)-2v_x(x,y)+v_x(x,y-h)\right]\\&\quad+\left(\frac{\lambda+2\mu}{h^2}\right)\left[v_x(x+h,y)-2v_x(x,y)+v_x(x-h,y)\right]\\&\quad+\left(\frac{\lambda+\mu}{4h^2}\right)\left[v_y(x+h,y+h)-v_y(x-h,y+h)-v_y(x+h,y-h)\right.\\&\quad\left.+v_y(x-h,y-h)\right]+b_x(x,y)=0\end{aligned} \tag{7}$$

$$\begin{aligned}&\left(\frac{\mu}{h^2}\right)\left[v_y(x+h,y)-2v_y(x,y)+v_y(x-h,y)\right]\\&\quad+\left(\frac{\lambda+2\mu}{h^2}\right)\left[v_y(x,y+h)-2v_y(x,y)+v_y(x,y-h)\right]\\&\quad+\left(\frac{\lambda+\mu}{4h^2}\right)\left[v_x(x+h,y+h)-v_x(x-h,y+h)-v_x(x+h,y-h)\right.\\&\quad\left.+v_x(x-h,y-h)\right]+b_y(x,y)=0.\end{aligned} \tag{8}$$

(In this work, we assume $h=1$ pixel. All subsequent discussion presupposes this fact.)

The boundary conditions for the partial differential equations (2) and (3) are that the velocity $\boldsymbol{v}$ is zero along the boundary, Ω, of the image:

$$\boldsymbol{v}(x,y)=\mathbf{0}\quad\forall(x,y)\in\Omega. \tag{9}$$

These boundary conditions in turn also imply that the displacement is zero along the boundary (*cf.* (4)).

2) Matrix Equation: It should be noted that the difference equations (7) and (8) must be solved at *every* pixel location (*i.e.,* every (x, y) pair) in the image under consideration. Thus, for an image composed of N_r rows and N_c columns, a system of $2N_r'N_c'$ coupled difference equations must be solved, where $N_c' = N_c - 2$ and $N_r' = N_r - 2$.[2] These equations can subsequently be expressed compactly in matrix form as

$$\mathbf{\Lambda}\boldsymbol{\nu} = \boldsymbol{\psi} \tag{10}$$

where

$$\begin{aligned}
\boldsymbol{\nu} &= \left[\begin{array}{cc} \boldsymbol{v}_x & \boldsymbol{v}_y \end{array}\right]^T && (11)\\
\boldsymbol{v}_x &= \left[\begin{array}{cccc} \boldsymbol{v}_x^2 & \boldsymbol{v}_x^3 & \cdots & \boldsymbol{v}_x^{N_r-1} \end{array}\right] && (12)\\
\boldsymbol{v}_x^i &= \left[\begin{array}{cccc} v_x(i,2) & v_x(i,3) & \cdots & v_x(i,N_c-1) \end{array}\right] && (13)\\
\boldsymbol{v}_y &= \left[\begin{array}{cccc} \boldsymbol{v}_y^2 & \boldsymbol{v}_y^3 & \cdots & \boldsymbol{v}_y^{N_r-1} \end{array}\right] && (14)\\
\boldsymbol{v}_y^i &= \left[\begin{array}{cccc} v_y(i,2) & v_y(i,3) & \cdots & v_y(i,N_c-1) \end{array}\right] && (15)\\
\boldsymbol{\psi} &= \left[\begin{array}{cc} \boldsymbol{b}_x & \boldsymbol{b}_y \end{array}\right]^T && (16)\\
\boldsymbol{b}_x &= \left[\begin{array}{cccc} \boldsymbol{b}_x^2 & \boldsymbol{b}_x^3 & \cdots & \boldsymbol{b}_x^{N_r-1} \end{array}\right] && (17)\\
\boldsymbol{b}_x^i &= \left[\begin{array}{cccc} b_x(i,2) & b_x(i,3) & \cdots & b_x(i,N_c-1) \end{array}\right] && (18)\\
\boldsymbol{b}_y &= \left[\begin{array}{cccc} \boldsymbol{b}_y^2 & \boldsymbol{b}_y^3 & \cdots & \boldsymbol{b}_y^{N_r-1} \end{array}\right] && (19)\\
\boldsymbol{b}_y^i &= \left[\begin{array}{cccc} b_y(i,2) & b_y(i,3) & \cdots & b_y(i,N_c-1) \end{array}\right] && (20)\\
\mathbf{\Lambda} &= \left[\begin{array}{cc} \mathbf{\Lambda}_{11} & \mathbf{\Lambda}_{12} \\ \mathbf{\Lambda}_{21} & \mathbf{\Lambda}_{22} \end{array}\right] && (21)
\end{aligned}$$

with

$$\begin{aligned}
\mathbf{\Lambda}_{11} = &-\mu\left[\mathbf{\Upsilon}(-\mathbf{1}_{N_c'}, 2\mathbf{1}_{N_c'}, -\mathbf{1}_{N_c'}) \otimes \mathbf{I}_{N_r'}\right] \\
&-\left(\mu + \frac{\lambda+\mu}{4}\right)\left[\mathbf{I}_{N_r'} \otimes \mathbf{\Upsilon}(-\mathbf{1}_{N_c'}, 2\mathbf{1}_{N_c'}, -\mathbf{1}_{N_c'})\right] && (22)
\end{aligned}$$

$$\mathbf{\Lambda}_{12} = \mathbf{\Lambda}_{21} = (\lambda+\mu)\left[\mathbf{\Upsilon}(-\mathbf{1}_{N_c'}, \mathbf{0}_{N_c'}, \mathbf{1}_{N_c'}) \otimes \mathbf{\Upsilon}(\mathbf{1}_{N_r'}, \mathbf{0}_{N_r'}, -\mathbf{1}_{N_r'})\right] \tag{23}$$

$$\begin{aligned}
\mathbf{\Lambda}_{22} = &-\mu\left(\left[\mathbf{\Upsilon}(-\mathbf{1}_{N_c'}, 2\mathbf{1}_{N_c'}, -\mathbf{1}_{N_c'}) \otimes \mathbf{I}_{N_r}\right] + \left[\mathbf{I}_{N_r'} \otimes \mathbf{\Upsilon}(-\mathbf{1}_{N_c'}, 2\mathbf{1}_{N_c'}, -\mathbf{1}_{N_c'})\right]\right) \\
&-\left(\frac{\lambda+\mu}{4}\right)\left(\left[\mathbf{\Upsilon}(\mathbf{1}_{N_c'}, \mathbf{0}_{N_c'}, \mathbf{1}_{N_c'}) \otimes -\mathbf{I}_{N_r'}\right] + \left[-2\mathbf{I}_{N_c'} \otimes -\mathbf{I}_{N_r'}\right]\right), && (24)
\end{aligned}$$

where $\otimes$ is a Kronecker product; $\mathbf{0}_n$ is an n-dimensional column vector of zeros; $\mathbf{1}_n$ is an n-dimensional column vector of ones; $\mathbf{I}_n$ is the $n \times n$ identity matrix; and $\mathbf{\Upsilon}(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma})$ is a tridiagonal matrix with $\boldsymbol{\alpha}$ below the main diagonal, $\boldsymbol{\beta}$ on the main diagonal, $\boldsymbol{\gamma}$ above the main diagonal, and every other matrix element equal to zero.

3) Successive Overrelaxation: The matrix equation in (10) must then be solved for the velocities $\boldsymbol{\nu}$. Here, the method of successive overrelaxation (SOR) [4] is employed to solve the equation at each iteration (*i.e.,* time step) of the deformation process. To solve (10) for $\boldsymbol{\nu}$ using SOR, $\mathbf{\Lambda}$ is first decomposed into three matrices: a diagonal matrix $\mathbf{D}$, a lower triangular matrix $\mathbf{L}$, and an upper triangular matrix $\mathbf{U}$, such that $\mathbf{\Lambda} = \mathbf{D} - \mathbf{L} - \mathbf{U}$. Beginning with an initial guess for the solution $\boldsymbol{\nu}$, one obtains improved solutions for $\boldsymbol{\nu}$ iteratively using

$$\boldsymbol{\nu}^{(n+1)} = (\mathbf{D} - \omega\mathbf{L})^{-1}\left[(\omega\mathbf{U} + (1-\omega)\mathbf{D})\,\boldsymbol{\nu}^{(n)} + \omega\boldsymbol{\psi}\right] \tag{25}$$

until the error ($\epsilon = ||\mathbf{\Lambda}\boldsymbol{\nu} - \boldsymbol{\psi}||$) is below a predefined threshold. In (25), $\omega \in (0, 2)$ is a scalar factor that helps accelerate the rate of convergence beyond that of the standard Gauss-Seidel method [4].

4) Forward Euler Method: After solving (10) for the velocities $\boldsymbol{\nu}$ via SOR, forward Euler integration is applied to (4) by discretizing the temporal derivative in the relationship between the velocity and the displacement. As a result, (4) can be written as

$$\boldsymbol{v}^{(t)}(x,y) = \frac{\boldsymbol{u}^{(t+\tau)}(x,y) - \boldsymbol{u}^{(t)}(x,y)}{\tau} + \left(\left[\boldsymbol{v}^{(t)}(x,y)\right]^T \nabla\right)\boldsymbol{u}^{(t)}(x,y). \tag{26}$$

Upon rearranging (26), the displacement for each pixel location at time $t+\tau$ is calculated using

$$\boldsymbol{u}^{(t+\tau)}(x,y) = \boldsymbol{u}^{(t)}(x,y) + \tau\boldsymbol{v}^{(t)}(x,y) - \tau\left(\left[\boldsymbol{v}^{(t)}(x,y)\right]^T \nabla\right)\boldsymbol{u}^{(t)}(x,y). \tag{27}$$

[2]The boundary conditions eliminate the need to solve for the two velocity components of pixels along each edge of the image. Hence, there are $2N_r'N_c'$ equations rather than $2N_rN_c$ equations.

Care must be taken when choosing the temporal step-size, τ, to ensure that a stable solution is obtained. To this end, the condition required for stability is

$$\tau \leq \frac{h^2}{2} \tag{28}$$

where τ is the temporal step-size and h is the spatial grid spacing [16]. This limit on the magnitude of τ effectively prevents excessively large deformations from occurring in any *single* time step (*i.e.,* deformation iteration).

After calculating the displacement for each pixel location at time $t+\tau$ using (27), the entire process is repeated for the next time step: the body forces are recomputed using (5), the matrix difference equation in (10) is solved for the velocities using SOR, and the Euler integration is performed again to obtain the displacement. This process is repeated until the deforming template image sufficiently matches the study image, indicating convergence.

5) Template Re-gridding: To preserve the topology of the image, the Jacobian of the deformation transformation must remain positive definite. A singular Jacobian would imply that grid points could intersect each other as a result of a transformation, which should be physically impossible.

The image deformation transformation at location (x, y) is defined as

$$\boldsymbol{f}(x,y) = \begin{bmatrix} x & y \end{bmatrix}^T - \left[\boldsymbol{u}(x,y) + \tau \boldsymbol{v}(x,y) - \tau \left(\left[\boldsymbol{v}(x,y)\right]^T \nabla \right) \boldsymbol{u}(x,y)\right], \tag{29}$$

which can be decomposed into

$$f_x(x,y) = x - u_x(x,y) - \tau \left[v_x(x,y) - v_x(x,y)\frac{\partial u_x(x,y)}{\partial x} - v_y(x,y)\frac{\partial u_x(x,y)}{\partial y}\right] \tag{30}$$

$$f_y(x,y) = y - u_y(x,y) - \tau \left[v_y(x,y) - v_x(x,y)\frac{\partial u_y(x,y)}{\partial x} - v_y(x,y)\frac{\partial u_y(x,y)}{\partial y}\right], \tag{31}$$

where the time-dependence of the quantities has been omitted for clarity.

The Jacobian of the transformation is then the determinant

$$J = \begin{vmatrix} \frac{\partial f_x}{\partial x} & \frac{\partial f_x}{\partial y} \\ \frac{\partial f_y}{\partial x} & \frac{\partial f_y}{\partial y} \end{vmatrix} \tag{32}$$

where

$$\frac{\partial f_x}{\partial x} = 1 - \frac{\partial u_x}{\partial x} - \tau \left[\frac{\partial v_x}{\partial x} - \frac{\partial v_x}{\partial x}\frac{\partial u_x}{\partial x} - v_x \frac{\partial^2 u_x}{\partial x^2} - \frac{\partial v_y}{\partial x}\frac{\partial u_x}{\partial y} - v_y \frac{\partial^2 u_x}{\partial x \partial y}\right] \tag{33}$$

$$\frac{\partial f_x}{\partial y} = -\frac{\partial u_x}{\partial y} - \tau \left[\frac{\partial v_x}{\partial y} - \frac{\partial v_x}{\partial y}\frac{\partial u_x}{\partial x} - v_x \frac{\partial^2 u_x}{\partial x \partial y} - \frac{\partial v_y}{\partial y}\frac{\partial u_x}{\partial y} - v_y \frac{\partial^2 u_x}{\partial y^2}\right] \tag{34}$$

$$\frac{\partial f_y}{\partial x} = -\frac{\partial u_y}{\partial x} - \tau \left[\frac{\partial v_y}{\partial x} - \frac{\partial v_x}{\partial x}\frac{\partial u_y}{\partial x} - v_x \frac{\partial^2 u_y}{\partial x^2} - \frac{\partial v_y}{\partial x}\frac{\partial u_y}{\partial y} - v_y \frac{\partial^2 u_y}{\partial x \partial y}\right] \tag{35}$$

$$\frac{\partial f_y}{\partial y} = 1 - \frac{\partial u_y}{\partial y} - \tau \left[\frac{\partial v_y}{\partial y} - \frac{\partial v_x}{\partial y}\frac{\partial u_y}{\partial x} - v_x \frac{\partial^2 u_y}{\partial x \partial y} + \frac{\partial v_y}{\partial y}\frac{\partial u_y}{\partial y} - v_y \frac{\partial^2 u_y}{\partial y^2}\right] \tag{36}$$

and the dependence on the location (x, y) has been omitted for clarity. It must be noted, however, that this Jacobian is calculated for every pixel location (x, y).

To ensure that the Jacobian does not become singular, a method of template re-gridding is performed as follows. If the Jacobian at *all* pixel locations is above a threshold σ, the displacement fields $\boldsymbol{u}$ are updated in the usual manner according to (27). If however, at time t, the Jacobian at *any* of the pixel locations is below the threshold σ, template re-gridding is performed. Specifically, a copy of the deformed template image is created, as if it were an original undeformed template image with no displacement applied:

$$\widetilde{\mathbf{T}}^{[i+1]}(x,y) = \widetilde{\mathbf{T}}^{[i]}(x',y') \tag{37}$$

where

$$(x',y') = (x,y) - (u_x^{(t)}(x,y;[i]), u_y^{(t)}(x,y;[i])). \tag{38}$$

In (38), $u_x^{(t)}(x,y;[i])$ indicates the x-direction displacement at location (x, y) at time t, corresponding to the i-th propagated template. When a template is propagated, the time is reset (to $t = 0$), and the displacement fields for the new template are initialized ($\boldsymbol{u}^{(t)}(x,y;[i+1]) = \mathbf{0}$).

6) Bilinear Interpolation: At each iteration of the deformation algorithm described above, a displacement is calculated for each pixel location. This displacement indicates the new location to which each pixel flows. Initially, each pixel location is a set grid point of the rectangular image. After a displacement is applied, however, any given pixel will no longer be located at any of the original image grid points. Because the processing at each iteration is performed at the original image grid points, though, pixel values must be mapped back to the image grid points. This mapping is accomplished by applying bilinear interpolation, where the value of the deformed template pixel value is a weighted average of the pixel values at the four fixed grid points that surround the location of the deformed pixel location. Specifically, the pixel value of the $(i+1)$-th deformed template $\widetilde{\mathbf{T}}^{[i+1]}$ at the *grid* location (x, y) is the pixel value of the i-th deformed template $\widetilde{\mathbf{T}}^{[i]}$ at the (non-grid) location (x', y') defined in (38), which is

$$\begin{aligned}\widetilde{\mathbf{T}}^{[i+1]}(x,y) &= \widetilde{\mathbf{T}}^{[i]}(x',y') \\ &= d_0^{-1}\Big\{d(\lceil x'\rceil,\lceil y'\rceil)\widetilde{\mathbf{T}}^{[i]}(\lceil x'\rceil,\lceil y'\rceil) + d(\lfloor x'\rfloor,\lceil y'\rceil)\widetilde{\mathbf{T}}^{[i]}(\lfloor x'\rfloor,\lceil y'\rceil) \\ &\quad + d(\lfloor x'\rfloor,\lfloor y'\rfloor)\widetilde{\mathbf{T}}^{[i]}(\lfloor x'\rfloor,\lfloor y'\rfloor) + d(\lceil x'\rceil,\lfloor y'\rfloor)\widetilde{\mathbf{T}}^{[i]}(\lceil x'\rceil,\lfloor y'\rfloor)\Big\}. \end{aligned} \tag{39}$$

In (39) $\lceil z\rceil$ is the ceiling of z (*i.e.,* rounding up to the nearest integer), $\lfloor z\rfloor$ is the floor of z (*i.e.,* rounding down to the nearest integer), and the *distance-weights* are given by

$$d(f_1(x), f_2(y)) = \left[(f_1(x)-x)^2 + (f_2(y)-y)^2\right]^{-1/2}, \tag{40}$$

with

$$d_0 = d(\lceil x'\rceil,\lceil y'\rceil) + d(\lfloor x'\rfloor,\lceil y'\rceil) + d(\lfloor x'\rfloor,\lfloor y'\rfloor) + d(\lceil x'\rceil,\lfloor y'\rfloor). \tag{41}$$

Naturally, the closer a grid point is to the non-grid location in question, the more significantly the pixel value of that grid point contributes to the pixel value at the non-grid location.

This interpolation must be done recursively through each propagated template until the original undeformed template image ($\widetilde{\mathbf{T}}^{[0]} = \mathbf{T}$) is reached. In this way, the pixel value of the deformed template at a non-grid location can be determined from the original undeformed template image.[3]

D. Multi-Resolution Extension

The main drawback to the image deformation algorithm is its computational intensity. To mollify this drawback, we propose the following multi-resolution extension to the image deformation algorithm. The idea is motivated by image decoding using a wavelet transform [1], where coarse features are transmitted and decoded first, with fine details of the image transmitted and decoded later. Time constraints in real applications may necessitate abruptly stopping the deformation algorithm prematurely. Therefore, a method that can provide a "coarse" deformation quickly may be desirable.

Essentially, the proposed extension downsamples images to smaller sizes, and applies the deformation algorithm on the smaller images. Then the final displacement result from using the smaller images is used as the initial displacement when applying the deformation algorithm on larger versions of the images. This approach exploits the fact that it is quicker to deform smaller images.

Consider deforming a template image, $\mathbf{T}$, of size $N_r \times N_c$ pixels. Let the image be square, with $N = N_r = N_c$. Moreover, let N be a power of two, such that $N = 2^C$.[4]

Define $\mathbf{T}_{(C)}$ to be the original template image $\mathbf{T}$. For $c = \{1, 2, 3, \ldots, C-1\}$, define the image $\mathbf{T}_{(c)}$ to be the downsampled image of size $2^c \times 2^c$ such that

$$\begin{aligned}\mathbf{T}_{(c)}(x,y) = \frac{1}{4}\big[&\mathbf{T}_{(c+1)}(2^x-1, 2^y-1) + \mathbf{T}_{(c+1)}(2^x-1, 2^y) \\ &+\mathbf{T}_{(c+1)}(2^x, 2^y-1) + \mathbf{T}_{(c+1)}(2^x, 2^y)\big]\end{aligned} \tag{42}$$

for the pixel located at (x, y). That is, each pixel in the smaller image is an average of four pixels in the larger image. Apply the same downsampling procedure on the study image $\mathbf{S}$ as well.

Let $\boldsymbol{u}_{(c)}^{(t)}(x, y)$ be the displacement at location (x, y), at time t, when the template image is $\mathbf{T}_{(c)}$. Beginning with the smallest image, $\mathbf{T}_{(1)}$, apply the deformation algorithm. When the deformation algorithm converges, at, say, time $t = t_f$, each pixel location will have a corresponding displacement $\boldsymbol{u}_{(1)}^{(t_f)}(x, y)$ associated with it. Next initialize the displacement for the deformation of the next largest image, $\mathbf{T}_{(2)}$, such that

$$\boldsymbol{u}_{(2)}^{(0)}(2^x-1, 2^y-1) = \boldsymbol{u}_{(2)}^{(0)}(2^x, 2^y-1) = \boldsymbol{u}_{(2)}^{(0)}(2^x-1, 2^y) = \boldsymbol{u}_{(2)}^{(0)}(2^x, 2^y) = \boldsymbol{u}_{(1)}^{(t_f)}(x, y). \tag{43}$$

[3]This requisite "back-propagation" of the displacement fields is the most computationally (time-) intensive aspect of the deformation algorithm. The structure of the calculation, however, is ideal for a parallel computing architecture. Specifically, the back-propagation calculation for each individual pixel can be assigned to a different processor, effectively removing the calculation's dependence on the image size.

[4]In the event that the image is not square or that N is not a power of two, zero-padding can be applied to satisfy these two conditions.

Then apply the deformation algorithm again, which gives $\boldsymbol{u}_{(2)}^{(t_f)}(x,y)$ at convergence. Repeat this process iteratively, using increasingly larger images at each iteration, until the deformation algorithm is applied to the original image, $\mathbf{T}_{(C)}$.

II. Gaussian Processes

A. Introduction

Assume we have a set of L labeled data points

$$\mathcal{D} = \{\boldsymbol{x}_i, y_i\}_{i=1}^{L} \tag{44}$$

where $\boldsymbol{x}_i \in \mathbb{R}^d$ is the i-th vector, labeled as $y_i \in \{-1, 1\}$. In probit regression [9], the probability of label y_i given $\boldsymbol{x}_i$ is

$$p\left(y_i|\boldsymbol{x}_i\right) = \Phi(y_i f(\boldsymbol{x}_i)) = \Phi(y_i \boldsymbol{w}^T \boldsymbol{x}_i), \tag{45}$$

where

$$\Phi(a) = \int_{-\infty}^{a} \mathcal{N}(z; 0, 1) dz \tag{46}$$

is the cumulative density function of the standard normal distribution, and $\boldsymbol{w}$ constitutes a classifier. In a *parametric* binary classification task, one wishes to learn the function $f(\boldsymbol{x}_i) = \boldsymbol{w}^T \boldsymbol{x}_i$ — or equivalently the classifier $\boldsymbol{w}$ — that will correctly classify new unlabeled data points.

Alternatively, a non-parametric approach would avoid the explicit parameterization of f, and hence would avoid obtaining a classifier $\boldsymbol{w}$ explicitly. Because the form of the classifier is not limited to a specific parametric form (such as the simple inner product in (45)), non-parametric techniques are more general than parametric approaches. One such non-parametric technique, Gaussian processes [15], places a prior on the space of latent functions f directly. The Gaussian process prior can then be readily incorporated into a Bayesian framework for classification [15], [18].

B. Theory

The final objective of the classification problem is to predict the class label y_* for a test input $\boldsymbol{x}_*$. When solving a regression problem with Gaussian processes, an analytic solution can be obtained; intractable integrals in the classification case preclude an analytic solution without approximations. The classification problem using Gaussian processes can be solved as follows [6].

The joint likelihood of the class labels of the labeled data points, $\boldsymbol{y} = [y_1, y_2, \ldots, y_L]$, is

$$p(\boldsymbol{y}|\boldsymbol{f}) = \prod_{i=1}^{L} p(y_i|f_i) = \prod_{i=1}^{L} \Phi(y_i f_i), \tag{47}$$

where Φ is from using the probit model of (46). Similarly, for a new unlabeled data point,

$$p(y_*|f_*) = \Phi(y_* f_*). \tag{48}$$

A Gaussian process (GP) is a collection of random variables, any finite number of which has a joint Gaussian distribution. A Gaussian process is fully specified by a mean function and a covariance function. In classification problems, the mean function is typically taken to be the zero function, which we will also assume hereafter. The (kernel) covariance function K_{ij}, which expresses the covariance between the value of the function f at the points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, is usually chosen to be a Gaussian covariance function of the form

$$K_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left\{-\frac{1}{2} g(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{\theta})\right\} + \alpha\delta(i, j) \tag{49}$$

with

$$g(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{\theta}) = \sum_{m=1}^{d} \theta_m (x_i^m - x_j^m)^2 \tag{50}$$

and where x_i^m is the m-th feature of $\boldsymbol{x}_i$ and $\boldsymbol{\theta}$ is a vector of length-scale hyperparameters. The kernel function essentially provides a measure of similarity between pairs of data points; the hyperparameters $\boldsymbol{\theta}$ weight the relative importance and contribution of each feature in measuring this similarity. In (49), α is a scalar and $\delta(i, j)$ is a delta function that makes computations better conditioned by "loading the diagonal." The hyperparameters $\boldsymbol{\theta}$ effectively weight the relative importance and contribution of each feature in measuring the similarity between pairs of data points via the kernel function.

Given L (labeled) data points, $\mathbf{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_L]$, the prior probability of the latent function values $\boldsymbol{f} = [f(\boldsymbol{x}_1), f(\boldsymbol{x}_2), \ldots, f(\boldsymbol{x}_L)]$ is the multivariate Gaussian

$$p(\boldsymbol{f}|\mathbf{X}, \boldsymbol{\theta}) = (2\pi)^{-L/2} |\mathbf{K}|^{-1/2} \exp\left\{-\frac{1}{2} \boldsymbol{f}^T \mathbf{K}^{-1} \boldsymbol{f}\right\} = \mathcal{N}\left(\boldsymbol{f}; \mathbf{0}, \mathbf{K}\right) \tag{51}$$

where the ij-th element of the covariance matrix $\mathbf{K}$ is given by (49).

Suppose we also have an additional (unlabeled) data point, $\boldsymbol{x}_*$. The joint distribution of the latent function values of the labeled and unlabeled data points is still a multivariate Gaussian,

$$p\left(\left[\begin{array}{c}\boldsymbol{f}\\ f_*\end{array}\right]\Big|\,\mathbf{X},\boldsymbol{\theta},\boldsymbol{x}_*\right)=\mathcal{N}\left(\left[\begin{array}{c}\boldsymbol{f}\\ f_*\end{array}\right];\left[\begin{array}{c}\mathbf{0}\\ 0\end{array}\right],\left[\begin{array}{cc}\mathbf{K} & \boldsymbol{k}_*\\ \boldsymbol{k}_*^T & k_{**}\end{array}\right]\right), \tag{52}$$

where $\boldsymbol{k}_*$ is a vector of covariances between the latent function values of the unlabeled data point and each of the L labeled data points, and k_{**} is the variance of the latent function value of the unlabeled data point. Conditioning (52) on the latent function values of the labeled data points will result in another Gaussian distribution, namely

$$p(f_*|\boldsymbol{f},\mathbf{X},\boldsymbol{\theta},\boldsymbol{x}_*)=\mathcal{N}\left(f_*;\boldsymbol{k}_*^T\mathbf{K}^{-1}\boldsymbol{f},k_{**}-\boldsymbol{k}_*^T\mathbf{K}^{-1}\boldsymbol{k}_*\right). \tag{53}$$

The predictive distribution of the class label for the unlabeled data point can be obtained from

$$p(y_*|\mathcal{D},\boldsymbol{\theta},\boldsymbol{x}_*)=\int p(y_*|f_*)p(f_*|\mathcal{D},\boldsymbol{\theta},\boldsymbol{x}_*)df_* \tag{54}$$

where the second factor within the integral, the distribution of the latent function values f_*, can be computed by marginalization:

$$p(f_*|\mathcal{D},\boldsymbol{\theta},\boldsymbol{x}_*)=\int p(f_*|\boldsymbol{f},\mathbf{X},\boldsymbol{\theta},\boldsymbol{x}_*)p(\boldsymbol{f}|\mathcal{D})d\boldsymbol{f}. \tag{55}$$

In (54), $p(y_*|f_*)$ is given by the probit model in (48).

In (55), $p(f_*|\boldsymbol{f},\mathbf{X},\boldsymbol{\theta},\boldsymbol{x}_*)$ is obtained by conditioning the joint Gaussian process prior on the latent function values of the labeled data points, which is given by (53). Furthermore, the posterior distribution of the latent function values $\boldsymbol{f}$, $p(\boldsymbol{f}|\mathcal{D},\boldsymbol{\theta})$, is obtained by using Bayes' rule,

$$p(\boldsymbol{f}|\mathcal{D},\boldsymbol{\theta})=\frac{p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f}|\mathbf{X},\boldsymbol{\theta})}{p(\mathcal{D}|\boldsymbol{\theta})}=\frac{\mathcal{N}(\boldsymbol{f};\mathbf{0},\mathbf{K})\prod_{i=1}^{L}\Phi(y_if_i)}{p(\mathcal{D}|\boldsymbol{\theta})}. \tag{56}$$

The use of the probit model prevents the posterior in (56) from being Gaussian.

Because (56) is not Gaussian, the integral in (55) is intractable, which in turn prevents the solution for (54) from being obtained analytically. Instead, the posterior distribution $p(\boldsymbol{f}|\mathcal{D},\boldsymbol{\theta})$ is usually approximated as a Gaussian with mean $\widetilde{\boldsymbol{\mu}}$ and covariance $\widetilde{\boldsymbol{\Sigma}}$ via either a Laplace approximation [7] or an Expectation-Propagation (EP) [11] approximation.

With $p(\boldsymbol{f}|\mathcal{D},\boldsymbol{\theta})\approx q(\boldsymbol{f}|\mathcal{D},\boldsymbol{\theta})=\mathcal{N}(\boldsymbol{f};\widetilde{\boldsymbol{\mu}},\widetilde{\boldsymbol{\Sigma}})$, the *approximate* posterior $q(f_*|\boldsymbol{f},\mathbf{X},\boldsymbol{\theta},\boldsymbol{x}_*)$ in (55) is then a Gaussian,

$$q(f_*|\boldsymbol{f},\mathbf{X},\boldsymbol{\theta},\boldsymbol{x}_*)=\mathcal{N}(f_*;\mu_*,\sigma_*^2), \tag{57}$$

where

$$\mu_* = \boldsymbol{k}_*^T\mathbf{K}^{-1}\widetilde{\boldsymbol{\mu}} \tag{58}$$

$$\sigma_*^2 = k_{**}-\boldsymbol{k}_*^T(\mathbf{K}^{-1}-\mathbf{K}^{-1}\widetilde{\boldsymbol{\Sigma}}\mathbf{K}^{-1})\boldsymbol{k}_*. \tag{59}$$

In turn, the *approximate* predictive probability $q(y_*|\mathcal{D},\boldsymbol{\theta},\boldsymbol{x}_*)$ for (54) can then be computed analytically — when the probit model is employed — as

$$q(y_*|\mathcal{D},\boldsymbol{\theta},\boldsymbol{x}_*)=\int\Phi(y_*f_*)\mathcal{N}(f_*;\mu_*,\sigma_*^2)df_*=\Phi\left(\frac{y_*\mu_*}{\sqrt{1+\sigma_*^2}}\right). \tag{60}$$

Thus, (60) can be used to obtain the probability that any new unlabeled data point belongs to each class.

C. Implementation

The posterior distribution $p(\boldsymbol{f}|\mathcal{D},\boldsymbol{\theta})$ is usually approximated as a Gaussian with mean $\widetilde{\boldsymbol{\mu}}$ and covariance $\widetilde{\boldsymbol{\Sigma}}$ via either a Laplace approximation or an Expectation-Propagation (EP) approximation. The Laplace approximation [7] is a Taylor approximation of the un-normalized log-posterior, whose mean $\widetilde{\boldsymbol{\mu}}$ is placed at the mode (which coincides with the maximum *a posteriori* estimate), and whose covariance $\widetilde{\boldsymbol{\Sigma}}$ is equal to the negative inverse Hessian of the log-posterior density at $\widetilde{\boldsymbol{\mu}}$. The EP approach [11] instead matches the approximate marginal moments of $p(f_i|\mathcal{D})$ by the marginal distributions of the approximation $\mathcal{N}(f_i;\widetilde{\mu}_i,\widetilde{\Sigma}_{ii})$. It has been demonstrated experimentally that the EP approximation is more accurate than the Laplace approximation [6], so we employ the EP approach in our work.

EP is an iterative algorithm that can be applied to approximate the posterior $p(\boldsymbol{f}|\mathcal{D},\boldsymbol{\theta})$ as $q(\boldsymbol{f}|\mathcal{D},\boldsymbol{\theta})$. In particular, the non-Gaussian likelihood is approximated as a product of Gaussian *site functions*:

$$\prod_{i=1}^{L}p(y_i|f_i)\approx\prod_{i=1}^{L}s(f_i,\mu_i,\sigma_i^2,Z_i)=\prod_{i=1}^{L}Z_i\mathcal{N}(f_i;\mu_i,\sigma_i^2). \tag{61}$$

The *approximate* posterior distribution of $\boldsymbol{f}$ is therefore a Gaussian,

$$q(\boldsymbol{f}|\mathcal{D},\boldsymbol{\theta}) = \frac{p(\boldsymbol{f}|\mathbf{X},\boldsymbol{\theta})}{q(\mathcal{D}|\boldsymbol{\theta})}\prod_{i=1}^{L} s(f_i,\mu_i,\sigma_i^2,Z_i) = \mathcal{N}(\boldsymbol{f};\widetilde{\boldsymbol{\mu}},\widetilde{\boldsymbol{\Sigma}}), \tag{62}$$

where

$$\begin{aligned}
\widetilde{\boldsymbol{\mu}} &= \widetilde{\boldsymbol{\Sigma}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} && (63)\\
\widetilde{\boldsymbol{\Sigma}} &= (\mathbf{K}^{-1}+\boldsymbol{\Sigma}^{-1})^{-1} && (64)\\
\boldsymbol{\mu} &= [\mu_1,\ldots,\mu_L]^T && (65)\\
\boldsymbol{\Sigma} &= \text{diag}\left(\left[\sigma_1^2,\ldots,\sigma_L^2\right]\right). && (66)
\end{aligned}$$

The EP algorithm iteratively adjusts the site function parameters μ_i, σ_i^2, and Z_i to match moments of an approximation to the posterior marginal distributions. The k-th moment of f_i under the posterior is

$$\langle f_i^k\rangle = \frac{1}{p(\mathcal{D}|\boldsymbol{\theta})}\int f_i^k p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f}|\mathbf{X},\boldsymbol{\theta})d\boldsymbol{f} = \frac{1}{p(\mathcal{D}|\boldsymbol{\theta})}\int f_i^k p(y_i|f_i)p_{\backslash i}(f_i)df_i \tag{67}$$

where

$$p_{\backslash i}(f_i) = \int\prod_{j\neq i} p(y_j|f_j)p(\boldsymbol{f}|\mathbf{X},\boldsymbol{\theta})d\boldsymbol{f}_{\backslash i} \tag{68}$$

is a *cavity distribution*, and $\boldsymbol{f}_{\backslash i}$ denotes $\boldsymbol{f}$ without f_i. Approximating the likelihood with the site functions from (61) permits an approximation to the cavity distribution to be expressed as an unnormalized Gaussian [6]

$$q_{\backslash i}(f_i) = \int\prod_{j\neq i} s(f_j,\mu_j,\sigma_j^2,Z_j)p(\boldsymbol{f}|\mathbf{X},\boldsymbol{\theta})d\boldsymbol{f}_{\backslash i} \propto \mathcal{N}(f_i;\mu_{\backslash i},\sigma_{\backslash i}^2) \tag{69}$$

where

$$\begin{aligned}
\mu_{\backslash i} &= \sigma_{\backslash i}^2\left(\frac{\widetilde{\mu}_i}{\widetilde{\boldsymbol{\Sigma}}_{ii}} - \frac{\mu_i}{\sigma_i^2}\right) && (70)\\
\sigma_{\backslash i}^2 &= \left((\widetilde{\boldsymbol{\Sigma}}_{ii})^{-1} - \sigma_i^{-2}\right)^{-1}. && (71)
\end{aligned}$$

For the probit likelihood, $p(y_i|f_i) = \Phi(y_i f_i)$, the requisite moments of the approximate posterior marginal distribution can be computed analytically:

$$\begin{aligned}
m_0 &= \Phi(a_i) && (72)\\
m_1 &= \mu_{\backslash i} + \frac{\sigma_{\backslash i}^2\mathcal{N}(a_i;0,1)}{y_i\Phi(a_i)\sqrt{1+\sigma_{\backslash i}^2}} && (73)\\
m_2 &= 2\mu_{\backslash i}m_1 - \mu_{\backslash i}^2 + \sigma_{\backslash i}^2 - \frac{a_i\sigma_{\backslash i}^4\mathcal{N}(a_i;0,1)}{\Phi(a_i)\left(1+\sigma_{\backslash i}^2\right)} && (74)
\end{aligned}$$

where

$$a_i = \frac{y_i\mu_{\backslash i}}{\sqrt{1+\sigma_{\backslash i}^2}}. \tag{75}$$

The result of the moment-matching gives the update equations for the site parameters:

$$\begin{aligned}
\mu_i &= \sigma_i^2\left(m_1\left(\sigma_{\backslash i}^{-2}+\sigma_i^{-2}\right) - \frac{\mu_{\backslash i}}{\sigma_{\backslash i}^2}\right) && (76)\\
\sigma_i^2 &= \left(\left(m_2-m_1^2\right)^{-1} - \sigma_{\backslash i}^{-2}\right)^{-1} && (77)\\
Z_i &= m_0\sqrt{2\pi\left(\sigma_{\backslash i}^2+\sigma_i^2\right)}\exp\left\{\frac{\left(\mu_i-\mu_{\backslash i}\right)^2}{2\left(\sigma_{\backslash i}^2+\sigma_i^2\right)}\right\}. && (78)
\end{aligned}$$

Then $\widetilde{\boldsymbol{\mu}}$ and $\widetilde{\boldsymbol{\Sigma}}$ can be updated using (63) and (64), respectively. This process is performed iteratively until the site parameters μ_i, σ_i^2, and Z_i have converged.

When employing Gaussian processes for classification, the only "classifier training" that must be performed is to learn appropriate values for the hyperparameters, $\boldsymbol{\theta}$, of the covariance function in (49). This task can be accomplished by maximizing the evidence, which is the marginal likelihood in (56),

$$p(\mathcal{D}|\boldsymbol{\theta}) = \int p(\boldsymbol{y}|\boldsymbol{f}) p(\boldsymbol{f}|\mathbf{X}, \boldsymbol{\theta}) d\boldsymbol{f}. \tag{79}$$

The EP formulation provides an approximation to the (log) marginal likelihood (79) as a byproduct, from which $\boldsymbol{\theta}$ can be optimized. Alternatively, for low-dimensional problems (*i.e.,* small d), an exhaustive search over the values of $\boldsymbol{\theta}$ can easily be conducted; the value of $\boldsymbol{\theta}$ that maximizes the (log) marginal likelihood is then chosen.

In the EP approach, the *approximation* to the log marginal likelihood, $\log p(\mathcal{D}|\boldsymbol{\theta})$, is

$$\log q(\mathcal{D}|\boldsymbol{\theta}) = \log \int q(\boldsymbol{f}|\mathbf{X}, \boldsymbol{\theta}) \prod_{i=1}^{L} s(f_i, \mu_i, \sigma_i^2, Z_i) d\boldsymbol{f} \tag{80}$$

$$= \sum_{i=1}^{L} \log Z_i - \frac{1}{2} \log |\mathbf{K} + \boldsymbol{\Sigma}| - \frac{1}{2} \boldsymbol{\mu}^T (\mathbf{K} + \boldsymbol{\Sigma})^{-1} \boldsymbol{\mu} - \frac{L}{2} \log(2\pi), \tag{81}$$

which can be seen to depend heavily on the site parameters.

APPENDIX

For completeness, we list the settings of the various parameters used in the deformation algorithm, which is allowed to run for 100 iterations. The Lamé constants in (1) are $\lambda = 0$ and $\mu = 1$. In the body force equation, (5), $\alpha = 100$. The spatial step-size in (7) and (8) is $h = 1$ pixel, and the temporal step-size in (26) is $\tau = 0.25$. The number of rows and columns of the images are N_r and N_c, respectively. In (25), the SOR parameter is $\omega = 2(1 + \sin(\pi/(\sqrt{N_r N_c} + 1)))^{-1}$. The SOR error threshold is $\epsilon = 10^{-4}$. The Jacobian threshold to propagate the deformed template is $\sigma = 0.5$.

REFERENCES

[1] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies. Image coding using wavelet transform. *IEEE Transactions on Image Processing*, 1(2):205–220, 1992.

[2] R. Bajcsy and S. Kovačič. Multiresolution elastic matching. *Computer Vision, Graphics and Pattern Recognition*, 46:1–21, 1989.

[3] G. Christensen, R. Rabbitt, and M. Miller. Deformable templates using large deformation kinematics. *IEEE Transactions on Image Processing*, 5(10):1435–1447, 1996.

[4] G. Golub and C. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 1983.

[5] D. Hill, P. Batchelor, M. Holden, and D. Hawkes. Medical image registration. *Physics in Medicine and Biology*, 46:R1–R45, 2001.

[6] M. Kuss and C. Rasmussen. Assessing approximate inference for binary Gaussian process classification. *Journal of Machine Learning Research*, 6:1679–1704, 2005.

[7] D. MacKay. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 2003.

[8] J. Maintz and M. Viergever. A survey of medical image registration. *Medical Image Analysis*, 2(1):1–37, 1998.

[9] P. McCullagh and J. Nelder. *Generalized Linear Models, 2nd Edition*. Chapman & Hall, 1989.

[10] T. McInerney and D. Terzopoulos. Deformable models in medical image analysis. *Medical Image Analysis*, 1(2):91–108, 1996.

[11] T. Minka. *A Family of Algorithms for Approximate Bayesian Inference*. PhD thesis, MIT, 2001.

[12] D. Pham, C. Xu, and J. Prince. Current methods in medical image segmentation. *Annu. Rev. Biomedical Eng.*, 2:315–337, 2000.

[13] J. Pluim, J. Maintz, and M. Viergever. Mutual-information-based registration of medical images: A survey. *IEEE Transasctions on Medical Imaging*, 22(8):986–1004, 2003.

[14] J. Price. Lagrangian and Eulerian representations of fluid flow: Kinematics and the equations of motion. Technical report, Woods Hole Oceanographic Institute, Woods Hole, MA, 2006.

[15] C. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[16] G. Smith. *Numerical Solution of Partial Differential Equations: Finite Difference Methods*. Oxford University Press, third edition, 1985.

[17] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformable models. *Computer Graphics*, 21(4):205–214, 1987.

[18] C. Williams and D. Barber. Bayesian classification with Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12), 1997.

[19] B. Zitová and J. Flusser. Image registration methods: A survey. *Image and Vision Computing*, 21:977–1000, 2003.