A. Supplementary material

This supplementary material contains additional information describing our approach. §A.1 discusses the theoretical properties of our model and proves that the resulting spatial transformations are diffeomorphic in the continuum. Possible undesirable effects of the numerical discretization are not studied or addressed in this work. §B provides some critical implementation details for the CNN regressing the local pre-weights of the multi-Gaussian regularizer based on an input image. Lastly, §C provides details on how the synthetic data for our synthetic experiments was created.

A.1. Localized multi-Gaussian kernels

Starting from a sum of kernels $\sum_{i=0}^{N-1} w_i G_i$, we let the coefficient w_i be spatially varying. In order to ensure the diffeomorphic property of deformations, we set the weights $w_i(x) = G_{\sigma_{\text{small}}} \star \omega_i(x) + \varepsilon_i$, where $\omega_i(x)$ are pre-weights which are convolved with a Gaussian filter with small standard deviation and ε_i is a small positive real that acts as a constant offset parameter[6]. We have

$$\begin{aligned}&\text{Reg}_{\text{vSVF}} = \lambda\langle m_0, v_0\rangle + \lambda_{\text{OMT}} \int \text{OMT}(w(x))\,\mathrm{d}x + \\ &\lambda_{\text{TV}} \sqrt{\sum_{i=0}^{N-1}\left(\int \gamma(\|\nabla I_0(x)\|)\|\nabla\omega_i(x)\|_2\,\mathrm{d}x\right)^2}\,, \qquad \text{(A.1)}\end{aligned}$$

where m_0 and v_0 are the initial momentum and vector field, respectively. Note that the partition of unity defining the metric, intervenes in the L^2 scalar product $\langle m_0, v_0\rangle$ since, with $\varepsilon_i > 0$ a positive offset,

$$\begin{aligned}&v_0(x) = (K(w) \star m_0)(x) \\ &= \sum_{i=0}^{N-1} \sqrt{w_i(x)} \int_y G_i(|x-y|)\sqrt{w_i(y)} m_0(y)\,\mathrm{d}y\,, \qquad \text{(A.2)}\end{aligned}$$

whose spatial smoothness is enough to guarantee the deformation to be diffeomorphic. Due to the convolution of the pre-weights, the vector field v_0 has a bounded norm in the space of C^1 vector fields which implies that its flow is a diffeomorphism at every time. In fact, we have:

Proposition 1. *The minimization of the objective functional* (A.1) *over a collection of image pairs provides diffeomorphic deformations for every pair of images. At every stage of the optimization procedure, the deformations are guaranteed to be diffeomorphic.*

Proof. We have the existence of a constant K such that

$$\|f\|_{C^1} \leq K\|f\|_{H_i} \leq K\|f\|_{H_N}\,, \qquad \text{(A.3)}$$

for every $f \in H_N$.

Denote by $\Phi : (I, m) \mapsto \omega$ the nonlinear map learnt by the neural network. At every step of the optimization, and at convergence (for a finite sample of pairs of images, each pair is denoted by the index j), the functional (A.1) is finite, which implies that $\Phi(I_j, m_j)$ is pointwisely bounded on the domain and is in TV, therefore, $G_{\text{small}} \star w_i$ has a bounded C^1 norm, as well as $\sqrt{w_i}$ since $w_i > \varepsilon_i > 0$. In addition, $E_j = \langle m_j, K(w)m_j\rangle$ is also finite and gives an upper bound for $\|G_N \star (w_i m_j)\|_{H_N}$. Thus, we have

$$\begin{aligned}&\|\sum_{i=0}^{N-1} \sqrt{w_i(x)} G_i(|x-y|)\sqrt{w_i(y)} \star m_j\|_{C^1} \\ &\quad \leq KN \sup_{i=1,\dots,N}(\|\sqrt{w_i}\|_{C^1}\|\sqrt{w_i} m_j\|_{H_N})\,. \qquad \text{(A.4)}\end{aligned}$$

Therefore, the norm of the velocity field $v(x) = \sqrt{w_i(x)} G_i(|x-y|) \star \sqrt{w_i} m_j$ is bounded in C^1 and its flow is a diffeomorphism. □

Also, there is a corresponding variational derivation of the spatially varying kernel with the square root which is presented next.

A.1.1 Variational derivation

Let us detail the variational definition of the spatially varying kernel used in Equation (A.2). Consider

$$\|v\|_H^2 = \inf\left\{\sum_{i=0}^{N-1}\|v_i\|_{H_i}^2 \,\middle|\, \sum_{i=0}^{N-1}\sqrt{w_i} v_i = v\right\}. \qquad \text{(A.5)}$$

Using Lagrange multipliers, we get critical points of the functional

$$\sum_{i=0}^{N-1}\frac{1}{2}\|v_i\|_{H_i}^2 + \langle p, \sum_{i=0}^{N-1}\sqrt{w_i} v_i - v\rangle\,, \qquad \text{(A.6)}$$

therefore we get

$$L_i v_i + w_i p = 0 \quad \forall i = 0, \dots, N-1\,, \qquad \text{(A.7)}$$

where L_i is the inverse of the kernel G_i. Hence, there exists p such that

$$\|v\|_H^2 = \sum_{i=0}^{N-1}\langle G_i \sqrt{w_i} p, \sqrt{w_i} p\rangle$$

for the norm. Moreover, since $v_i = G_i\sqrt{w_i}p$, we have

$$v = \sum_{i=0}^{N-1}\sqrt{w_i} G_i(\sqrt{w_i} p)\,. \qquad \text{(A.8)}$$

[6] We enforce this small positive constant by clamping the pre-weights to $[\epsilon, 1]$. One could also directly integrate this into the weighted linear softmax definition by clamping to $[\epsilon, 1]$ instead of $[0, 1]$.

B. Implementation details

CNN initialization/penalty. Directly using the CNN as described in §3.1.1 does, in our experience, not lead to stable estimation results for the weights. Proper initialization and penalizing undesirable weights is therefore essential. Specifically, we use the following approaches:

1) *Initialization:* We initialize all bias terms to zero and use the initialization scheme from [19] for the convolutional weights. For the last batch normalization layer we initialize the slope to a small value (0.025) to avoid massive weight changes at the beginning as the registration is very sensitive to such changes.

2) *Weighted linear softmax input penalty:* As the weighted linear softmax function clamps inputs, values within the clamping range will no longer produce gradients. In our experiments this was a highly problematic behavior as it appeared to lead to cases where one could not easily recover from poor locations in the input space to the weighted linear softmax[7]. Hence, we penalize the inputs when they are outside the $[0, 1]$ range as follows:

$$\mathrm{rp}(z) = \sum_{i=0}^{N-1} \left(w_i + z_i - \overline{z} - \mathrm{clamp}_{\epsilon,1}(w_i + z_i - \overline{z})\right)^2 . \tag{B.1}$$

Here, $\mathrm{clamp}_{\epsilon,1}$ clamps values to the interval $[\epsilon, 1]$. An $\epsilon > 0$ is required as the square root is not differentiable at zero. This penalty is integrated over all of space and added to the overall registration energy, *i.e.*,

$$\mathrm{RP}(z(x)) = \int \mathrm{rp}(z(x))\, \mathrm{d}x \ . \tag{B.2}$$

We did not experiment with weightings of this term and simply added it as is. In practice this appeared to be fine (but may warrant further investigation) as the term results in zero penalty when the input values to the weighted linear softmax are not clamped and it is operating in its linear regime.

3) *Weight decay:* We use a small weight decay [17] (set to 1e-5) applied to all the network weights. However, we did not extensively experiment with this parameter. Hence, its practical necessity is not clear to us at the moment. We added it to mitigate possible drift in the estimated parameters (*e.g.*, very large weights of the convolutional filters).

C. Generation of synthetic data

To be able to validate with respect to a known ground truth we construct synthetic data as follows:

1) We generate concentric circular regions with random radii and associate different multi-Gaussian weights to these regions. We associate a fixed multi-Gaussian weight to the background.

2) We randomly create vector momenta at the borders of the concentric circles. Specifically, we randomly create 10 different sectors and, within each sector, we randomly create either all positive or negative momenta orthogonal to the circle boundaries. These momenta are smoothed afterwards.

3) Based on 2), we create a deformation.

4) We randomly create a noisy image of the same dimension as the image of the concentric circles and smooth it. We add this smoothed noise image to the concentric circle image and deform it and its associated weights given the deformation from 3). The resulting image is our synthetic source image. We also transform the image without noise.

5) We repeat steps 2) to 4), starting from the synthetic source image without noise. The resulting deformation is applied to the (noisy) synthetic source image to create the synthetic target image.

These steps are repeated to obtain a desired set of image pairs.

[7] Similarly, if one uses a standard softmax function then exponential terms may result in very small gradients.