1 LDDMM Algorithm

Algorithm 1. Let $\Omega \in \mathbb{R}^3$ be the background space. Let $I_0 : \Omega \to \mathbb{R}$ be a template image which will be deformed to match target image $I_1 : \Omega \to \mathbb{R}$. Let $\varphi_{t0} = \int_t^0 v(t, \varphi_{t0})dt$ be the forward mapping, $\varphi_{t1} = \int_t^1 v(t, \varphi_{t1})dt$ be the reverse maping where $v(t) : \Omega \to \mathbb{R}^3$ denotes the velocity of the flow. Thus $I(t) = I_0 \circ \varphi_{t0}$ is the forward deformed template image. Furthermore let $\boldsymbol{K_V}$ be the kernel used to smooth velocity fields. We discretize time domain $[0, 1]$ so that $t_0 = 0$ and $t_{J-1} = 1$ where J is the number of time steps. The optimal flow v^* is estimated using a *gradient descent* algorithm below.

```
v = 0, v_old = 0, E_old = ∞, ε = SMALL, ε_min = SMALLER
while k < MAX_ITERATIONS

    // Calculate energy gradient.
    foreach j ∈ {0, ..., J − 1}
        p(t_j) = −(1/σ²) (I_0 ∘ φ_{t_j 0} − I_1 ∘ φ_{t_j 1}) |Dφ_{t_j 1}|
        I(t_j) = I_0 ∘ φ_{t_j 0}
    ∇_v E = v + K_V[p∇I]

    // Update velocity using energy gradient.
    while True
        v = v − ε∇_v E // Take a step.
        E(v) = (1/2) ∫_0^1 ‖v(t)‖_V² dt + (1/(2σ²)) ‖I(1) − I_1‖_{L_2}²
        if E(v) < E_old // If energy decreased...
            ε = ε ∗ 1.05 // ...increase step size...
            (v_old, E_old) = (v, E(v)) // ...and save old values.
            BREAK
        else // If energy increased...
            v = v_old // ...restore old velocity...
            ε = ε ∗ 0.5 // ...and decrease step size...
            if ε < ε_min // ...but stop if min step size reached.
                v* = v
                STOP
```