A Supplementary Material

We provide additional information on our RobOT layer and the two derived methods:

- **S-RobOT**: feature matching with RobOT, followed by a **smoothing** that is implemented in closed form or relies on an optimization loop – as detailed in Sec. 3.1.
- **D-RobOT**: S-RobOT as a pre- and post-processing tool for a **deep** deformation estimator – as detailed in Sec. 3.2.

More specifically:

1. Sec. A.1 provides details on **PVT1010**, our dataset of pulmonary vascular trees.
2. Sec. A.2 describes our **synthetic deformations** for augmenting PVT1010.
3. Sec. A.3 contains more information on global feature matching with **S-RobOT**.
4. Sec. A.4 provides details on our deep deformation prediction approach **D-RobOT**, with additional results on the PVT1010 and Kitti datasets.
5. Sec. A.5 discusses the differences between **RobOT and nearest neighbor projection**.
6. Sec. A.6 details our **computational resources**.
7. Finally, Sec. A.7 discusses the **societal impact** of our work.

A.1 Pulmonary vascular tree dataset

Introducing the PVT1010 dataset. We introduce a new pulmonary vascular tree dataset for point cloud registration. The PVT1010 dataset includes 1,010 pairs of inhale/exhale lung vascular trees extracted from 3D computed tomography (CT) images; 10 of these correspond to the 10 cases of the public DirLab-COPDGene [19] dataset which includes, for each pair, 300 expert annotated landmarks that are in correspondence with each other and that we use to validate our results. We extracted the lung vascular geometry in both inspiratory and expiratory CT scans using a scale-space particle system [62, 34] that is implemented in the Teem library [61]. We used the pipeline that is defined in the chest imaging platform [83, 102].

Legal and regulatory information. The vascular tree reconstructions that are used in this study were part of the COPDGene study (NCT00608764). This study has been IRB approved and participants have provided their consent. The investigators from the Brigham and Women's Hospital (Harvard Medical School) only had access to de-identified CT images to perform vascular reconstructions. Since we are performing secondary analysis using de-identified data, the work under consideration is not considered human subjects research and did not imply additional risks to the participants. Risks related to ionizing radiation exposure were described in the primary IRB-approved study. Study identifiers were re-coded for our release of PVT1010 to preserve anonymity. PVT1010 is released under the *Creative Commons Attribution-NonCommercial-ShareAlike 3.0 License*.

Point cloud representation. We now detail how we extracted the lung vascular trees as high-resolution 3D point clouds from the raw CT images. To perform this geometric segmentation task, we rely on a system of 4D particles that are defined by three spatial coordinates plus one scale parameter that corresponds to the local radius of the lung vessel – these radii are used by RobOT as point weights α_i and β_j in Eq. (2). Our segmentation method starts from a point cloud that is initialized using the Frangi filter [42]. Then, we fit this point cloud representation to our 3D CT volumes by minimizing iteratively a system energy that is expressed as the sum of:

- An **inter-particle regularization** energy that ensures convenient sampling properties. We rely on a sum of quartic polynomials of the pairwise point distances, with a tunable potential well that is chosen to induce regular sampling at a fixed distance between the points.
- A **particle-image data fidelity** term, which is computed using the image Hessian at the current particles' locations.

As a final result, we obtain points that are approximately equally distributed along the vessel centerlines. The vessel radii are first approximated as the image scales at which the middle eigenvalues of the Hessian are locally minimized, and then refined using the generative approach of [83].

The resulting dataset has the following properties, which result in a challenging registration task:

1. The vascular trees have a **complex structure** and exhibit **complex, large motions** between inhalation and exhalation.
2. To capture the complex anatomical structure of the lungs at millimeter scale, registration methods need to focus on branching points or rely on **high-resolution** point clouds.
3. Due to the fixed image resolution of the raw CT volume and to acquisition differences between the inhale and exhale scans, the extracted inhale and exhale vascular trees are **not fully consistent** with each other. This is especially true for tiny structures at the lung boundary, that may not be visible on the smaller lung images at exhalation time.

Sampling density, resolution. The original CT images from which we extract our point clouds are acquired with a uniform resolution on the x, y and z axes: depending on the patients, the side length of our voxels varies between 0.60 mm and 0.65 mm. Using the processing above, we turn these volumetric images into 3D point clouds with 60k samples per lung vascular tree: depending on the subject, the average sampling distance (from each point to its nearest neighbor in the 3D point cloud) ranges between 0.6 mm and 1.0 mm.

We note that for our 10 test cases, the Dirlab annotations were performed on a down-sampled volume with a resolution of 2.5 mm on the z axis. To guarantee a fair comparison between image-based and point-based methods, we follow standard practice for this dataset (`https://www.dir-lab.com/Results.html`) and report our results in Tab. 1 with a "snap-to-voxel" post-processing: we quantize our 3D lung registrations on the original grids (with spacing $\sim 0.625\,\text{mm} \times 0.625\,\text{mm} \times 2.5\,\text{mm}$) before computing the average errors and percentiles. In practice, we note that this quantization slightly lowers the 25% percentile of the registration errors (as we "snap" many displacements to the correct voxel or slice) but increases the 50% and 75% percentiles (some landmarks get "snapped" to the wrong slice). Please note that in the Supplementary Material, we do not include this quantization step for e.g. ablation studies: this allows us to study more precisely the impact of each layer in our architecture.

Volume vs point cloud representation. We stress that our point clouds contain much less information than the original 3D volumes from which they have been sampled. We discard all the intensity (grayscale) values and only retain the sparse geometric support of the lung vascular tree – 60k points out of 100M+ voxels. As a consequence, our registration task on 3D point clouds is significantly harder than the original DirLab benchmark (`https://www.dir-lab.com/Results.html`). Whereas optimization-based methods on the full CT **volumes** reach a nearly perfect accuracy of 0.60 mm to 1.00 mm with run times on the order of the **minute** [19, 98, 90, 18, 56, 55, 89, 119, 99, 120], our **point** neural networks reach an average accuracy of 2 mm to 4 mm in one or two **seconds**.

The main purpose of our work on the PVT1010 dataset is to show that fast and accurate registration is now at hand, even on very degraded anatomical data. This is of significant interest for clinical practice: our method is suitable for **real-time processing** and has **built-in robustness** to changes of the CT acquisition parameters that may affect the intensities of the raw image volumes. Going forward, as detailed in the conclusion of our manuscript, we intend to work on improving the accuracy of our method with a better sampling strategy and image-based features. Packaging our method as an accessible Python toolbox will also open the door to a genuine multi-center evaluation of our trained models.

A.2 Augmentation of the training dataset for vascular tree registration

We now describe how to augment the PVT1010 dataset with synthetic deformations in order to create a large training set with **dense ground truth annotations** for lung registration. We proceed in four steps: voxel-grid sampling; local deformation; global deformation; local property distortion and degradation using an inconsistent sub-sampling. Our efficient implementation lets us generate synthetic training pairs online – just like a standard data augmentation layer. We showcase our local and global deformations in Fig. 6.

1. Voxel-grid sampling. To start, we use a voxel-grid strategy to re-sample the raw point clouds with a standard sampling density. First, we subdivide the volume space into coarse 3D blocks with spacing $s_{\text{voxelgrid}} = 0.03\,\text{mm}$. Second, we sort all points into these cubic bins according to their (x, y, z) coordinates. Third, we compute one barycenter per cubic cell to down-sample the original point cloud. Since we work with *weighted* point clouds, these local centroids are associated to the sums of the weights of all points in the corresponding cells.

2. Local deformation. We then sample control points from the vessel trees and generate random displacements that we smooth using an anisotropic spline model:

1. We first sample $\mathrm{C} = 1,000$ spline control points x_c uniformly at random from the point cloud $(x_1, \ldots, x_\mathrm{N})$.
2. Second, we compute local covariance matrices for the distribution of points x_i in a neighborhood of each control point x_c, using an isotropic Gaussian kernel window.
3. We compute the three eigenvalues (e_c^1, e_c^2, e_c^3) and unit eigenvectors (v_c^1, v_c^2, v_c^3) of each local covariance matrix to determine the main direction of the lung vessel. To avoid rank deficiency, we use a lower threshold of 0.2 on the eigenvalues e_c^k. For each control point x_c, we then normalize the vector of three eigenvalues (e_c^1, e_c^2, e_c^3) as $(e_c^1, e_c^2, e_c^3)/\sqrt{(e_c^1)^2 + (e_c^2)^2 + (e_c^3)^2}$.
4. For each control point x_c, we create a numerically stable anisotropic covariance matrix $\Sigma_c = \sum_{k=1}^3 (s_\text{local} e_c^k)^2 \, v_c^k {v_c^k}^\top$, where $s_\text{local} = 4\,\text{mm}$ is a positive scaling factor.
5. To obtain a robust estimation of the local covariance structure of our point cloud, we re-run steps 2-4 with neighborhoods that are defined using an *anisotropic* Gaussian kernel window of covariance Σ_c.
6. For every control point x_c, we generate a random displacement vector $\Delta x_c \in \mathbb{R}^3$ such that $\|\Delta x_c\| \leqslant d_\text{local}$, drawn uniformly in the ball of center 0 and radius $d_\text{local} = 3\,\text{mm}$.
7. We smooth and interpolate the displacement vector field Δx_c from the control points x_c to the full point cloud $\{x_i\}$ using an anisotropic Nadaraya–Watson kernel interpolator:

$$x_i \;\leftarrow\; x_i \;+\; \frac{\sum_{c=1}^\mathrm{C} k_{\Sigma_c}(x_c, x_i) \Delta x_c}{\sum_{c=1}^\mathrm{C} k_{\Sigma_c}(x_c, x_i)} \;, \tag{9}$$

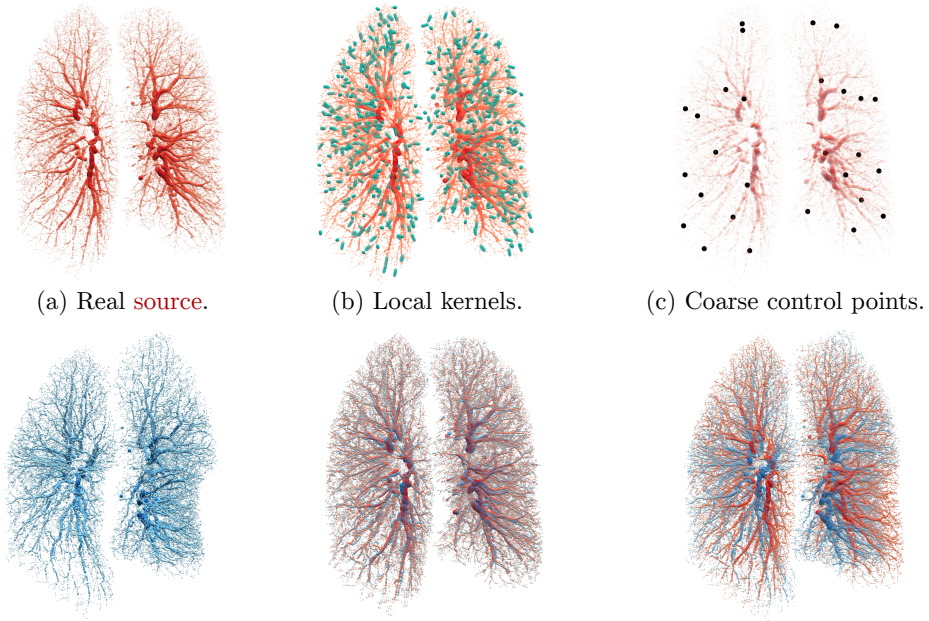
   where $k_{\Sigma_c}(x_c, \cdot)$ is a Gaussian kernel with local covariance Σ_c. This anisotropic formula ensures that the **local connectivity structure** of the lung vessel tree is preserved: the displacement of points that belong to the same vessel are strongly correlated with each other.

3. Global deformation. The step above simulates local relative displacements between lung vessels. To take large-scale breathing movements into account, we apply a second spline deformation which is smoother but has a larger magnitude. In practice, we use a voxel-grid sampling with spacing resolution $s_\text{global} = 90\,\text{mm}$ to obtain control points. We then generate random displacements of magnitude at most d_global for every control point, and interpolate them to the full point cloud using a Nadaraya–Watson estimator, parameterized by an isotropic Gaussian kernel with standard deviation σ_global.

4.a. Radius distortion. Having altered the 3D coordinates of our points using the deformations above, we add random noise to the local estimates α_i of the vessel radii – which are encoded as additional point features as detailed in Sec. A.1 and used in our RobOT layer as point weights. This additive noise is scaled by a positive parameter $s_\text{radius} = 0.1$ and drawn at random in $[-s_\text{radius}\alpha_i, s_\text{radius}\alpha_i]$.

4.b. Inconsistent sampling. By construction, the steps above let us create an arbitrary number of pairs of (real, simulated) lungs with known pairwise correspondence between all points. In order to simulate acquisition artifacts and introduce challenging inconsistencies, we sample $\mathrm{N} = \mathrm{M} = 60\text{k}$ points at random from the source and the synthetic (target) point clouds as a last generation step. We note that for training, the ground-truth flow is computed based on the source sampling and is not affected by this last degradation: it may or may not point to a sample in the synthetic target.

Augmentation of the the target and source point clouds. In our experiments, we generate our target point clouds using $d_\text{global} = 25\,\text{mm}$, $\sigma_\text{global} = 25\,\text{mm}$. Additionally, we also perform data augmentation for the source point cloud itself using the smaller values of $d_\text{global} = 8\,\text{mm}$ and $\sigma_\text{global} = 15\,\text{mm}$.



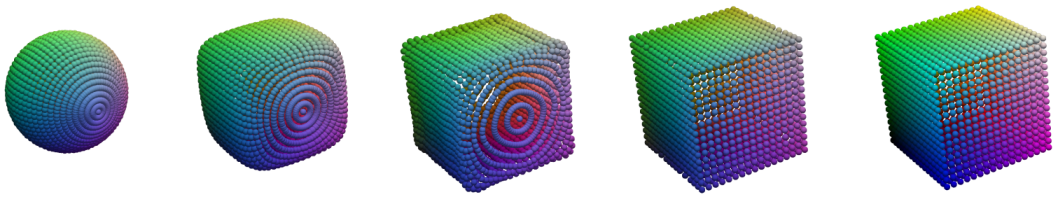
(a) Real source. (b) Local kernels. (c) Coarse control points.

(d) Synthetic target. (e) Local perturbation, source. (f) Global deformation, source.

Figure 6: **Local and global deformations that we use to generate our synthetic training dataset.** (a) Original (real) vascular tree. (b) Ellipsoids that represent anisotropic kernels of size 2 mm that we use to generate vessel-preserving local deformations. Note that in our experiments, we use larger kernels of size 4 mm that induce a stronger regularization but are harder to display cleanly. (c) Spline control points that we sample using a voxel-grid scheme with 90 mm spacing and use to generate a global deformation. (d) Synthetic vascular tree, the output of the process that we use as a target for training. (e) Source point cloud after local deformation. (f) After global deformation.

(a) Source. (b) Target. (c) FilterReg, target. (d) S-RobOT, target. (e) S-RobOT weights.

Figure 7: **Partial rigid registration** of the Stanford scans based on the matching of FPFH features – as discussed in Sec. 3.1 and Suppl. A.3.1. The feature-based CPD model FilterReg falls in a local minimum while the rigid S-RobOT of Eq. (5) results in a successful registration. In the last column, we display the attention weights w_i that are derived from unbalanced OT.



(a) Source. (b) Spline. (c) LDDMM. (d) RobOT. (e) Target.

Figure 8: **Smooth RobOT (S-RobOT) on a toy registration problem,** the deformation of a sphere (left) onto a cube (right). From left to right, we display: (a) the source shape A $= (x_1, \dots, x_N)$; (b) the output of Spline RobOT from Eq. (7); (c) the output of LDDMM RobOT, solution of the optimization problems of Eq. (8,10); (d) the output of the "raw" RobOT matching $x_i \mapsto x_i + v_i$ from Eq. (4); (e) the target shape B $= (y_1, \dots, y_M)$.

		Rotation			Translation		
	Method	**MSE** degrees^2 ↓	**RMSE** degrees ↓	**MAE** degrees ↓	**MSE** 3D units^2 ↓	**RMSE** 3D units ↓	**MAE** 3D units ↓
Matching	ICP	1134.552	33.683	25.045	0.0856	0.293	0.250
	FGR [134]	126.288	11.238	2.832	0.0009	0.030	0.008
	Go-ICP [130]	195.985	13.999	3.165	0.0011	0.033	0.012
	S-RobOT (rigid)	8.939	2.989	1.313	0.0002	0.014	0.009
End-to-end	PointNetLK [2]	280.044	16.735	7.550	0.0020	0.045	0.025
	DCP(v2) [121]	45.005	6.709	4.448	0.0007	0.027	0.020
	PRNet [122]	10.235	3.199	1.454	0.0003	0.016	0.010
	Partial-OT [28]	**0.107**	**0.328**	**0.052**	**3.384e-06**	**0.00183**	**0.0003**

Table 2: **Partial-to-Partial registration with unseen objects on ModelNet40.**

		Rotation			Translation		
	Method	**MSE** degrees^2 ↓	**RMSE** degrees ↓	**MAE** degrees ↓	**MSE** 3D units^2 ↓	**RMSE** 3D units ↓	**MAE** 3D units ↓
Matching	ICP	1217.618	34.894	25.455	0.086	0.293	0.251
	FGR [134]	98.635	9.932	1.952	0.0014	0.038	0.007
	Go-ICP [130]	157.072	12.533	2.940	0.0009	0.031	0.010
	S-RobOT(rigid)	50.997	7.142	4.012	0.0005	0.022	0.013
End-to-end	PointNetLK [2]	526.401	22.943	9.655	0.0037	0.061	0.033
	DCP(v2) [121]	95.431	9.769	6.954	0.0010	0.034	0.025
	PRNet [122]	24.857	4.986	2.329	0.0004	0.021	0.015
	Partial-OT [28]	**0.127**	**0.357**	**0.069**	**3.953e-06**	**0.002**	**0.0004**

Table 3: **Partial-to-Partial registration with unseen categories on ModelNet40.**

A.3 Additional material on S-RobOT and global feature matching

A.3.1 Partial registration

Experiment 1: Stanford scans. We now discuss a toy example, illustrated in Fig. 7, that showcases the benefits of unbalanced optimal transport theory for partial rigid registration. We normalize and resample the partial Stanford scans using a voxel-grid sampling with 0.005 spacing. We then rotate every source shape by (120°,10°,10°) degrees to create a target shape. We illustrate two methods:

- As a baseline competitor, we use the FilterReg [44] implementation of `https://github.com/neka-nat/probreg` (a feature-based CPD method [81]) with noise ratio set to 0.7 and an automatic annealing strategy based on the Expectation-Maximization (EM) algorithm.
- For our rigid S-RobOT registration, we first compute a 33-dimensional FPFH feature vector [100] for each point using a radius of 0.02 for the normal search and a radius of 0.05 for the feature search. We then rely on Eqs. (4-5) to compute the weighted RobOT matching and project it onto the space of rigid transformations. For our robust OT problem, we set the *blur* parameter to 0.1 and the *reach* parameter to 2 units in $\mathbb{R}^{33}$.

Results. Given standard FPFH features [100], our RobOT approach successfully registers the low-overlap pair while FilterReg [44] fails. The RobOT confidence weights w_i of Eq. (4) naturally act as an attention mechanism.

Experiment 2: ModelNet40. Going further, we evaluate our partial registration strategy S-RobOT on the standard dataset ModelNet-40 [127] with comparisons to state-of-the-art methods. Instead of relying on FPFH features, we use a self-supervised deep feature learning strategy to increase the robustness and accuracy of the registration. The ModelNet40 dataset contains 12,311 CAD (Computer-Aided Design) models, from 40 object categories. We follow the same experimental setup as in [122, 28]:

1. We normalize the point clouds to fit in the cube $[-1, 1]^3$.
2. To generate a registration pair, we first sample a random source shape using 1,024 points. We then apply a random rigid transformation along each axis, with rotation angle in $[0°, 45°]$ and translation in $[-0.5, +0.5]$.
3. To simulate a partial acquisition, we sample one point at random in both of the source and target shapes. In each shape, we then keep the 768 nearest neighbors of these points to define the observed regions.

Evaluation metrics. To assess the quality of our rigid registrations, we evaluate the rotation and translation errors separately. On each component, we compute the mean squared error (MSE), the root mean squared error (RMSE) and the mean absolute error (MAE).

S-RobOT setup. For feature learning, we proceed in two steps:

1. Given a source point cloud, we synthesize a target point cloud by applying a random rigid transform. For convenience, we re-use the augmentation strategy detailed above: along each axis, we sample a rotation in $[0°, 45°]$ and a translation in $[-0.5, +0.5]$.
2. We train a **self-supervised deep feature extractor** using the loss function of Sec. A.3.3. We use a PointNet++ [93] with 10,654 parameters that learns a 30-dimensional feature vector for each point.

For the S-RobOT matching, we rely on the rigid projection formula of Eq. (5); we set the *blur* parameter to $\sigma = 0.01$ and the *reach* parameter to $\tau = 10$ units in $\mathbb{R}^{30}$.

Partial-to-Partial registration with unseen objects. We follow [122, 28] and split the dataset of 12,311 point clouds into a training set with 9,843 shapes and a testing set with 2,468 shapes. We first train on **all** 9,843 shapes from **all 40 categories** in the training set and test on **all** 2,468 unseen shapes in the test set. In Table 2, we compare S-RoboT with feature matching methods (ICP, FGR [134], Go-ICP [130]) and end-to-end deep learning models (PointNetLK [2], DCP(v2) [121], PRNet [122] and Partial-OT [28]).

Partial-to-Partial registration with unseen categories. We then follow [122] and test the generalization ability of our model between object categories: we train on the first 20 categories of ModelNet40 and test on the remaining 20 categories. We report these results in Tab. 3.

Results. Overall, we observe that the unsupervised Rigid S-RobOT method performs much better than traditional approaches and is close to end-to-end methods. Since our main focus is on free-form registration, we do not push these experiments further: we use Affine and Rigid S-RobOT as pre-alignment steps and are satisfied with this level of performance.

We note that the best-performing method Partial-OT also relies on an optimal transport layer: as discussed in [105], **partial** OT is a very close cousin of the theory of **unbalanced** OT that we leverage in our RobOT layer. We understand the Partial-OT method as a supervised and end-to-end version of our S-RobOT baseline. Both [28] and this manuscript thus show that **optimal transport theory is ready to be part of the standard toolbox in our field**, with state-of-the-art performance in varied and complementary application settings.

A.3.2 Diffeomorphic registration

Background on LDDMM. Going beyond rigid and affine transformation models, the Large Deformation Diffeomorphic Metric Mapping (LDDMM) framework captures large, smooth and invertible deformations in a principled way [5]. This fluid-based model is standard in computational anatomy, especially for applications to neuroimaging where the preservation of shape topology is a key registration prior [35].

In the LDDMM model, deformations are encoded via the integration of a time-varying velocity field $v^t = \frac{\mathrm{d}}{\mathrm{d}t} x^t$ over the unit time interval $t \in [0,1]$. Given any two shapes A and B to register with each other, we seek a geodesic path $(v^t)_{t\in[0,1]}$ for a chosen Riemannian metric $\|v\|_K^2$ that is induced by a convolution kernel K_x over the space of vector fields $v : \mathbb{R}^3 \to \mathbb{R}^3$.

Following standard derivations from optimal control theory [5, 54, 35], we know that plausible deformations are fully parameterized by the **momentum** $m^0 : \mathbb{R}^3 \to \mathbb{R}^3$ at time $t = 0$, a vector field that acts as the parameter "θ" of the LDDMM deformation model. In the LDDMM framework, the optimization problem of Eq. (8) reads:

$$\theta^* = m^{0*} = \underset{m^0:\mathbb{R}^3\to\mathbb{R}^3}{\arg\min} \underbrace{\lambda_{\text{reg}} \langle m^0, K_x * m^0 \rangle_{L^2(\mathbb{R}^3,\mathbb{R}^3)}}_{\text{Regularization.}} + \underbrace{\sum_{i=1}^{\mathrm{N}} w_i \|x_i + v_i - \text{Morph}(m^0, x_i)\|^2_{\mathbb{R}^3}}_{\text{Fidelity to the data.}} , \tag{10}$$

where the deformation model Morph : $(m^0, x_i^0) \mapsto x_i^1$ is computed through the integration of the geodesic shooting equation:

$$\tfrac{d}{dt} x^t = +\tfrac{\partial H}{\partial m}\left(x^t, m^t\right), \ \tfrac{d}{dt} m^t = -\tfrac{\partial H}{\partial x}\left(x^t, m^t\right) \tag{11}$$

from time $t = 0$ to time $t = 1$ for the Hamiltonian $H(x, m) = \frac{1}{2}\langle m, K_x * m\rangle_{L^2(\mathbb{R}^3,\mathbb{R}^3)}$. We refer to Chapter 5 of [35] for a presentation and implementation of these equations on point clouds with the PyTorch and KeOps libraries.

Black-box deformation models. Different transformation models may result in different projection results. In Fig. 8, we show several smooth RobOT (S-RobOT) results for spline and LDDMM models on a toy registration task.

Experiment. The source sphere (left) and the target cube (right) are sampled with $\mathrm{N} = 1,922$ and $\mathrm{M} = 1,538$ points respectively. We normalize their coordinates to be contained in $[-1,1]^3$ and normalize the point cloud weights to sum up to one ($\alpha_i = 1/1,922$ and $\beta_j = 1/1,538$ so that $\sum_{i=1}^{1,922} \alpha_i = \sum_{j=1}^{1,538} \beta_j = 1$). For the initial RobOT matching, we set the *blur* parameter to 0.005 units in $\mathbb{R}^3$ and the *reach* parameter to $+\infty$ (balanced OT). For the following spline and LDDMM regularizations, we use a multi-Gaussian-kernel with standard deviations $\{0.05, 0.2, 0.3\}$ and weights $\{0.2, 0.3, 0.5\}$, i.e. a kernel function:

$$\begin{aligned} k(x,y) \ &= \ 0.2 \cdot \exp\big[-\|x-y\|^2_{\mathbb{R}^3}/(2\cdot 0.05^2)\big] \ + \ 0.3 \cdot \exp\big[-\|x-y\|^2_{\mathbb{R}^3}/(2\cdot 0.2^2)\big] \\ &+ \ 0.5 \cdot \exp\big[-\|x-y\|^2_{\mathbb{R}^3}/(2\cdot 0.3^2)\big] \, . \end{aligned} \tag{12}$$

Results. We make two important observations:

- As evidenced by the disappearance of the **polar sampling pattern** in the fourth column, the raw RobOT matching $x_i \mapsto x_i + v_i$ provides a "perfect fit" to the target but does not preserve the topology of the source point cloud. The spline and the LDDMM approximations alleviate this problem: they smooth the weighted RobOT matching to combine accuracy with topology preservation and, in the case of the LDDMM model, guarantees of invertibility.
- The spline and the LDDMM model both result in smooth deformations. But crucially, the LDDMM model is more suited to large deformations and provides a **better fit to the corners of the target cube**. We note that workarounds exist for the over-smoothing of the second (spline) column: at the cost of an increased sensitivity to noise, singular kernels [109] or ridge regression methods [10] allow spline models to get a closer fit to the target. For medical applications, the key benefit of LDDMM is that it provides strong guarantees on the invertibility of the deformation: we refer to Chapter 5 of [35] for a detailed discussion. In the remainder of this work, we benchmark both spline and LDDMM deformation models whenever relevant. We observe that diffeomorphic LDDMM registrations perform better on e.g. lung data, but stress that such comparisons are task-dependent.

A.3.3 Deep feature learning

Feature learning on 3D point clouds. As discussed above, (x, y, z) coordinates or standard FPFH features can be good enough to handle simple shapes and deformations. On challenging settings however, learning task-specific features is often key to high performance.

In Sec. 3.2 and Suppl. A.4, we present our best-performing solution to this problem: the D-RobOT architecture. In this section, we discuss an **alternative approach** that decouples feature learning and matching. In practice, we did not obtain competitive results with this method. Nevertheless, using a separate feature extractor may increase interpretability and ease deployment issues in medical scenarios: we believe that this line of work is worth pursuing and provide full details on our experiments.

Let us process a synthetic pair of point clouds with N points in correspondence with each other – $\mathrm{N} = 60,000$ in our experiments for lung registration. Instead of sampling positive and negative samples, which is common for contrastive loss functions, we choose to rely on $\mathrm{N} \times \mathrm{N}$ correspondence matrices that take **all possible point pairs into account**. We use the KeOps library [37] to manipulate these objects efficiently, with extremely fast run times and without memory overflows.

Specifically, we train our feature extraction network as follows:

1. **Input data.** We assume that we are given two point clouds $(x_1, \dots, x_{\mathrm{N}})$ and $(y_1, \dots, y_{\mathrm{N}})$ that are in pairwise correspondence with each other. In our experiments, these are typically the output of a synthetic "data augmentation" procedure: the Flying3D objects for the Kitti benchmark and our synthetic lung pairs for the Dirlab benchmark.
2. **Feature extraction.** We apply the feature extractor (a trainable point neural network) on both point clouds, independently from each other. We retrieve point features p_i and q_i in $\mathbb{R}^{\mathrm{D}}$ that are respectively associated to the points x_i and y_i.
3. **Feature normalization.** As discussed in Sec. 2.1, we normalize the feature vectors so that $\|p_i\|_{\mathbb{R}^{\mathrm{D}}} = \|q_i\|_{\mathbb{R}^{\mathrm{D}}} = 1$. This prevents the feature extractor from converging to degenerate solutions and ensures that our hyper-parameters for the RobOT problem of Eq. (2) can be interpreted as sensible scales in $\mathbb{R}^{\mathrm{D}}$.
4. **Source self-similarities.** We compute the $\mathrm{N} \times \mathrm{N}$ correspondence matrix for the point positions in the source point cloud:

$$c_{(x_i,x_j)} = \mathrm{softmax}_{j=1}^{\mathrm{N}}(-\|x_i - x_j\|^2_{\mathbb{R}^3}/2\kappa^2) = \frac{\exp(-\|x_i - x_j\|^2_{\mathbb{R}^3}/2\kappa^2)}{\sum_{j=1}^{\mathrm{N}} \exp(-\|x_i - x_j\|^2_{\mathbb{R}^3}/2\kappa^2)} \,, \quad (13)$$

where the Softmax denotes a mirrored exponential followed by a normalization over the rows of the correspondence matrix while $\kappa > 0$ is a scaling factor. Each row of the matrix $c_{(x_i,x_j)}$ then refers to a probability distribution, a position heatmap with a peak at point x_i whose radius is proportional to κ.

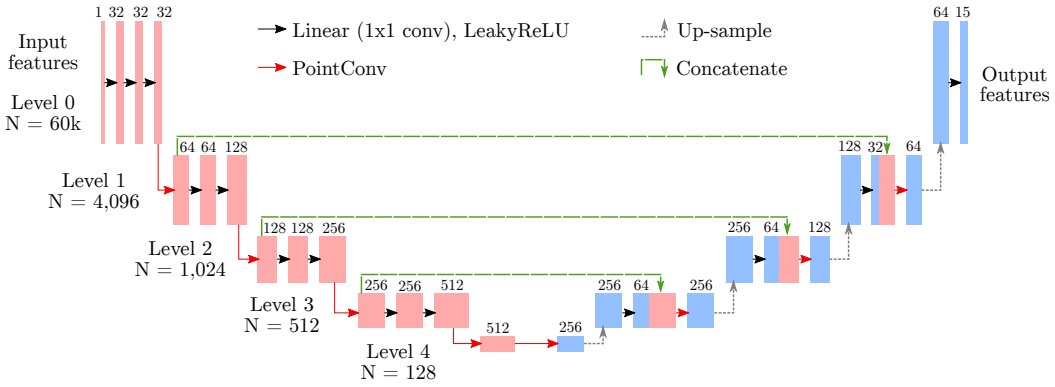


Figure 9: **Deep feature extractor** for our S-RobOT experiments. We adapt this feature pyramid network from PointPWC-Net [126]. We implement a four-scale U-net [95] architecture using: point-wise multi-layer perceptrons on the feature embeddings (with LeakyReLU non-linearities); PointConv [125] for downsampling and gathering information from the neighbors; K-Nearest Neighbor interpolation for upsampling.
Please note that this architecture relies on the point coordinates (x, y, z) to define the PointConv convolution and the upsampling layer. In our experiments for lung registration, we use the local vessel radius as our only input feature: this corresponds to using a single input channel on the left-most layer of the figure above.

5. **Feature correspondences.** Similarly, we compute a correspondence matrix for the source and target features:

$$c_{(p_i,q_j)} = \text{softmax}_{j=1}^{\text{N}}(-\|p_i - q_j\|_{\mathbb{R}^{\text{D}}}^2) \,. \tag{14}$$

   Each row of $c_{(p_i,q_j)}$ is a feature heatmap that indicates how well p_i corresponds to q_j.
6. **Training loss.** Our total loss is the sum over the cross entropies for each row:

$$\text{CE}(c_{(x_i,x_j)}, c_{(p_i,q_j)}) = -\sum_{i=1}^{\text{N}}\sum_{j=1}^{\text{N}} c_{(x_i,x_j)} \log\ c_{(p_i,q_j)} \,. \tag{15}$$

   We optimize it by stochastic gradient descent over the parameters of the feature extraction network (step 2).

Hyper-parameters. For feature learning, we use a U-net structured PointConv [125] architecture with 7M parameters that is illustrated in Fig. 9. At the feature learning stage, we set κ (described above) to $\sqrt{2}$ mm. For each point we learn a 15 dimensional feature vector of unit length. For the RobOT feature matching, we set the *blur* parameter to $\sigma = 0.01$ units in $\mathbb{R}^{15}$ and the *reach* parameter to $\rho = +\infty$ (balanced OT). As discussed in Suppl. A.4.2, we benchmark two types of S-RobOT regularization:

1. The spline smoothing of Eq. (7), using a Gaussian (RBF) kernel with a standard deviation of 5 mm.
2. An LDDMM deformation model that we optimize as in Eq. (8) using an SGD solver. We use a multi-Gaussian-kernel with standard deviations $\{5, 8, 10\}$ mm and weights $\{0.2, 0.3, 0.5\}$.

A.4 Additional material on deep deformation prediction

We now provide full details on our D-RobOT architecture and additional experiments on scene flow estimation and lung registration.

A.4.1 Deep registration module

Architecture of the prediction network. In Sec. 3.2, we rely on a modified PointPWC-Net to act as a predictor $\text{Pred} : (x_i, y_j) \mapsto \theta$. This multiscale point neural network takes as input the source and target point clouds. It returns a high-dimensional parameter θ for the deformation model $\text{Morph} : (\theta, x_i) \mapsto \hat{y}_i$. In this work, the predicted θ is always a 3D vector field that is supported by

the points x_i (for the raw displacements model) or by a collection of control points c_i that have been generated by farthest point sampling (for the spline and LDDMM models).

We describe our modifications to the original PointPWC-Net architecture in Fig. 10. In order to define a network that can predict spline and LDDMM parameters:

1. We replace the flow prediction layer of PointPWC-Net by a suitable prediction layer for registration parameters.
2. We use an asymmetric hierarchical architecture that outputs registration parameters for the control points c_i. Since the number of control points is often much smaller than the number of points in the input point cloud, this reduces the memory footprint of our network architecture.

Losses. We use synthetic data pairs for training: for scene flow estimation on Kitti, we rely on the synthetic Flying3D dataset; for lung registration, we rely on the two-scales simulator of Suppl. A.2. In both cases, we thus have access to ground truth deformations and can rely on a supervised learning strategy. Let us consider a PointPWC-Net with L scales, and denote by $(x_1^l, \dots, x_{\mathrm{N}_l}^l)$ the subsampled input for the l-th scale. For training, we compute a multi-scale similarity loss as:

$$\mathrm{Loss}(\hat{y}_i) \;=\; \sum_{l=0}^{\mathrm{L}-1} W^l \sum_i w_i \cdot \| \, \hat{y}_i^l \;-\; y_i^l \, \|_2^2 \,, \tag{16}$$

where $l \in \{0, \dots, \mathrm{L}\}$ is a scale ($l = 0$ corresponds to the raw point clouds), W^l is a scalar hyper-parameter that we use as a total weight for the l-th scale, $\hat{y}_i^l$ is the output of the network that corresponds to the flowed x_i^l at the l-th scale and y_i^l denotes the ground truth target that corresponds to the same point x_i^l with weight w_i. Note that at the finest scale, we compute $\hat{y}^0$ using a task-specific deformation model, Morph : $(\theta, x_i) \to \hat{y}_i$.

A.4.2 Experiments on lung vascular trees

We now provide more details on our experiments for lung registration, which are illustrated in Fig. 11 with a visualization of landmark errors in Fig. 13. Notably, we discuss both supervised and unsupervised training strategies as well as the influence of the deformation model Morph : $(\theta, x_i) \mapsto \hat{y}_i$ on the D-robot architecture.

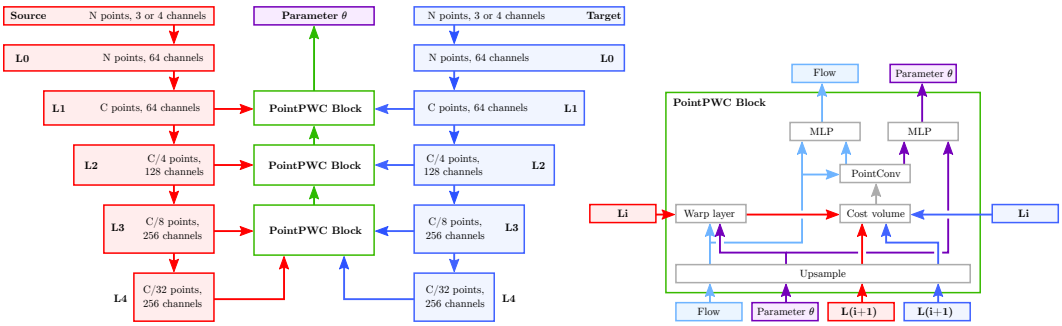
Synthetic data vs unsupervised learning. To overcome the lack of dense 3D annotations on real shape data, we advocate the use of simulated deformations and synthetic training datasets. As detailed in Sec. 3.2, we observe that D-RobOT networks are easy to train to a high level of accuracy: RobOT-based post-processing and fine-tuning help our models to bridge the domain gap between the synthetic and real distributions of shapes.

We would like to stress that our focus on synthetic training datasets to the detriment of e.g. unsupervised approaches results from **careful and extensive experiments**. In the lead-up to the publication of this work, we tried several competing approaches to train our registration networks: supervised learning with dense correspondences on synthetic data; "unsupervised" learning on unannotated point clouds using geometric loss functions between point sets; a mix of both approaches. In practice, supervised learning on synthetic data clearly emerged as the most practical option for challenging registration tasks. Let us briefly explain why.

Unsupervised loss functions. When dense pointwise correspondences are not available, a popular strategy to train registration methods is to rely on permutation-invariant loss functions between point clouds. We refer to Chapters 3 and 4 of [35] for an introduction to the topic.

Since our PVT1010 dataset contains 1,000 pairs of lung vessel trees that are in correspondence with each other (inspiration/expiration) but for which no expert-annotated landmarks are available, we are in a perfect situation to try out these tools. We thus attempted to train our networks using local Laplacian matching [126], Maximum Mean Discrepancies, Gaussian Mixture Models (GMM) as well as Wasserstein distances [35] on (x, y, z) coordinates. Unfortunately, we **never succeeded in converging to a competitive level of accuracy**. We believe that this is due to the **complex geometric structure of the lung vascular trees**, which are significantly more intricate than the clean point clouds and surface meshes on which these methods are usually tested [36, 40].

Combining synthetic (supervised) and real (unsupervised) data. Going further, we also tried to use a **mixed strategy**: in our training dataset, we combined synthetic deformations of real source shapes (that can be handled using a mean square error) with genuine target point clouds (that can be



(a) Multiscale prediction architecture.

(b) Architecture of a PointPWC block.

Figure 10: **Architecture of our deep prediction network** Pred : $(x_i, y_j) \mapsto \theta$, which is adapted from PointPWC-Net [126]. The original PointPWC-Net architecture is a multi-scale network that predicts 3D scene flow (a raw displacement field) in a coarse-to-fine fashion. We modify this state-of-the-art architecture to make it suitable for **registration with a task-specific deformation model** Morph : $(\theta, x_i) \mapsto \hat{y}_i$.
a) First of all, we compute **feature pyramids** at scales L0 to L4 using farthest point sampling and PointConv layers [125] on the source and target shapes (left and right columns) – see Fig. 5 in [126]. In our experiments, we always use $\mathrm{C} = 4,096$ or $\mathrm{C} = 8,192$ **control points**. The key design decision behind the PointPWC-Net architecture is to run through this pyramid from the coarsest scale (L4) to the finest one (L0, the original point clouds) in order to predict the final 3D scene flow: we represent this coarse-to-fine prediction as a stack of **PointPWC blocks** (central column).
b) Those blocks share the same architecture (but not the same neural weights): an **upsampling** layer that interpolates our 3D vector fields from the coarser to the finer scale using an inverse distance spline kernel; another **upsampling** layer that interpolates the pyramid features at the coarsest scale; a **warp layer** that deforms the source shape according to the up-sampled 3D flow, allowing the network to focus on **residual** deformations at each scale; a **cost "volume"** layer, detailed in Fig. 2 of [126], which relies on K-NN neighborhoods and PointConv layers to merge the source and target features into a vector of "patchwise" discrepancies for every point in the (subsampled and warped) source shape; a **prediction** PointConv and Multi-Layer Perceptrons (MLP), detailed in Fig. 6 of [126], that turns this vector of discrepancies into a predicted correction for the up-sampled 3D flow.
Please note that this architecture relies on the point coordinates (x, y, z) in the PointConv and upsampling layers. For our scene flow experiments, we use the (x, y, z) coordinates as input to the network: this corresponds to "Source" and "Target" arrays that have 3 input channels. For our lung registration experiments, we also add the local vessel radius as a fourth feature and thus use 4 input channels. We stress that in our RobOT-based pre-alignment and post-processing (steps 1 and 3 of the D-RobOT architecture), we rely on the (x, y, z) coordinates as point features p_i and q_j in Eq. (2). When available, the vessel radii are only used as point weights α_i and β_j in the RobOT problem.
In all our experiments, the parameter θ is a 3D vector field that has the same memory footprint as a "raw" 3D scene flow supported by our control points: this corresponds to using 3 output channels on the C control points. Our main modification to the original PointPWC-Net architecture is to **handle this registration parameter** (purple) **in parallel with the usual scene flow** (sky blue): the final prediction module of every PointPWC block runs **two estimations in parallel**, which share the same PointConv layer and are equivalent to Fig. 6 in [126].

handled using e.g. the Wasserstein distance). We expected that this combination would narrow the **domain gap** between our synthetic deformations and the real breathing movement. In practice, this strategy produced **indecisive results**:

- On the one hand, assuming that we only have access to a **simple deformation model** (e.g. a single-scale random field), this mixed training strategy improves the accuracy of our registrations. This is especially true when the synthetic deformations are not diverse enough.
- On the other hand, assuming that we have access to the more **realistic and expressive** deformation model of Suppl. A.2, the introduction of real but not annotated pairs in the training loop proves **slightly detrimental** to performance. We observe a small increase of about 0.2mm in landmark Root Mean Squared Error (RMSE).

For the sake of **simplicity**, we thus trained our final D-RobOT network entirely on synthetic deformations. We found that using RobOT as a **final post-processing** layer in the D-RobOT pipeline is enough to address prediction errors effectively and compensate for a small domain gap between synthetic and real deformations.

Raw displacements vs spline and LDDMM regularizations. In Fig. 12, we further compare the D-RobOT results for three different deformation models. Our main observations are that:

- A deep prediction module that returns raw displacements Δx_i with a naive deformation model $\text{Morph} : (\theta = \Delta x_i, x_i) \mapsto \hat{y}_i = x_i + \Delta x_i$ may produce non-smooth registration results. This holds even when the network is trained entirely on smooth, synthetic deformations. On the other hand, the regularizing spline and LDDMM models always result in smooth deformations. This vindicates the use of **explicit regularizing models** in safety-critical applications: we cannot trust our networks to "learn smoothness properties" from the data.
- According to Fig. 12 and Tab. 4, the LDDMM model captures large deformations better than the spline model. However, we obtain similar performance with both spline and LDDMM models after the (affordable) RobOT post-processing step: **fine-tuning with RobOT alleviates the need for complex and expensive deformation models**.

Experiments. We now provide detailed hyperparameters for the D-RobOT architecture in our lung registration experiments. As detailed in Sec. 3.2, the full model applies successively: an affine S-RobOT pre-alignment; a deep non-parametric registration; and a spline S-RobOT post-processing.

1. Affine registration. For affine pre-alignment, we use the S-RobOT model of Eq. (6) with raw (x, y, z) coordinates as input features in $\mathbb{R}^3$. The vessel radii are taken into account as point weights α_i and β_j. In the RobOT problem of Eq. (2), we set the *blur* parameter to $\sigma = 1$ mm and the *reach* parameter to $\tau = +\infty$ (balanced OT with strong constraints on the marginals).

2. Deep non-parametric registration. We evaluate three deformation models $\text{Morph} : (\theta, x_i) \mapsto \hat{y}_i$:

1. **Raw displacements** correspond to the case where the parameter $\theta = \Delta x_i$ is a 3D vector field that is supported by the point x_i and the deformation model is the addition:
$$\text{Morph} \ : \ (\Delta x_i, x_i) \in \mathbb{R}^{\mathrm{N}\times 3} \times \mathbb{R}^{\mathrm{N}\times 3} \ \mapsto \ x_i + \Delta x_i \in \mathbb{R}^{\mathrm{N}\times 3} \, . \tag{17}$$
As a regularization penalty, we use the squared Euclidean norm:
$$\text{Reg}(\Delta x_i) = \tfrac{1}{\mathrm{N}} \textstyle\sum_{i=1}^{\mathrm{N}} \|\Delta x_i\|^2_{\mathbb{R}^3} \, . \tag{18}$$
2. The **spline** model corresponds to the case where the parameter $\theta = \Delta c_l$ is a 3D vector field that is supported by a set of control points $(c_1, \dots, c_{\mathrm{C}})$ in $\mathbb{R}^3$. As a deformation model, we use a simple kernel smoothing:
$$\text{Morph} \ : \ (\Delta c_l, x_i) \in \mathbb{R}^{\mathrm{C}\times 3} \times \mathbb{R}^{\mathrm{N}\times 3} \ \mapsto \ x_i + \textstyle\sum_{l=1}^{\mathrm{C}} k(x_i, c_l) \Delta c_l \in \mathbb{R}^{\mathrm{N}\times 3} \, . \tag{19}$$
As a regularization penalty, we use the squared Euclidean norm:
$$\text{Reg}(\Delta c_l) = \tfrac{1}{\mathrm{C}} \textstyle\sum_{l=1}^{\mathrm{C}} \|\Delta c_l\|^2_{\mathbb{R}^3} \, . \tag{20}$$
For our lung experiments, we always use a multi-Gaussian kernel with standard deviations $\{3, 6, 9\}$ mm and weights $\{0.2, 0.3, 0.5\}$. With the coordinates of points x and y in millimeters, this corresponds to the positive definite kernel function:
$$\begin{aligned} k(x,y) \ = \ & 0.2 \cdot \exp\big[- \|x-y\|^2_{\mathbb{R}^3} / (2 \cdot 3^2) \big] \ + \ 0.3 \cdot \exp\big[- \|x-y\|^2_{\mathbb{R}^3} / (2 \cdot 6^2) \big] \\ & + \ 0.5 \cdot \exp\big[- \|x-y\|^2_{\mathbb{R}^3} / (2 \cdot 9^2) \big] \, . \end{aligned} \tag{21}$$

3. The **diffeomorphic LDDMM** model is equivalent to a spline deformation with continuous time. Notably, we use the same kernel $k(x, y)$ as in the spline model above. As detailed in Suppl. A.3.2 and Chapter 5 of [35], the parameter $\theta = m_l^0$ is a 3D vector field that is supported by a set of control points $(c_1, \dots, c_{\mathrm{C}})$ in $\mathbb{R}^3$. As a deformation model, we use an iterative kernel smoothing and deformation layer that corresponds to a numerical integration scheme for the Ordinary Differential Equation (11). More specifically, we use the differentiable Dopri-5 layer that is provided by the TorchDiffEq library [23, 22], which implements the Dormand-Prince or Runge-Kutta 4(5) integrator [32] with 20 time steps from time $t = 0$ to time $t = 1$. For a simpler implementation, we refer to the Euler integration scheme that is detailed in [35], Algorithm 5.6. As a regularization term, we use the squared kernel norm:

$$\mathrm{Reg}(\Delta c_l) = \tfrac{1}{\mathrm{C}} \sum_{l=1}^{\mathrm{C}} \sum_{s=1}^{\mathrm{C}} k(c_l, c_s) \langle m_l^0, m_s^0 \rangle_{\mathbb{R}^3} \ . \tag{22}$$

For the **deep prediction network** $\mathrm{Pred} : (x_i, y_j) \mapsto \theta$, we use the 4-scale hierarchical architecture that is described in Suppl. A.4.1 and set the scale weights W^l to $\{1.0, 0.8, 0.4, 0.2\}$. We compute the 4-scale decompositions of the point clouds using Farthest Point Sampling. For all models, we promote a close fit to the target shape and weight the regularization term $\mathrm{Reg}(\theta)$ by 1/100 in the training loss:

$$\mathrm{Loss}(x_i, y_i, \theta) \ = \ \tfrac{1}{100}\mathrm{Reg}(\theta) \ + \ \underbrace{\sum_{l=0}^{\mathrm{L}-1} W^l \sum_i w_i \cdot \| \, \hat{y}_i^l(\theta, x_i) \ - \ y_i^l \, \|_{\mathbb{R}^3}^2}_{\text{Multiscale loss of Eq. (16)}} \ . \tag{23}$$

We recall that in the expression above:

1. The weights w_i are attached to the points x_i and are proportional to the vessel radius.
2. The weights W^0, W^1, W^2 and W^3 put a strong emphasis on the finest scales.
3. Each point y_i^l corresponds to the ground truth target for x_i at scale l.
4. Each point $\hat{y}_i^l(\theta, x_i)$ corresponds to the output of our prediction and deformation networks at scale l for x_i. As detailed in Fig. 10, the finest-scale output $\hat{y}_i^0(\theta, x_i) = \mathrm{Morph}(\theta, x_i)$ corresponds to our model-based deformation; the larger-scale predictions $\hat{y}_i^1(\theta, x_i)$, $\hat{y}_i^2(\theta, x_i)$ and $\hat{y}_i^3(\theta, x_i)$ correspond to raw displacements and are only used in Eq. (23) to make the training easier [126].

3. Postprocessing. To fine-tune our registration, we use the spline S-RobOT deformation of Eq. (7) with (x, y, z) coordinates as input features in $\mathbb{R}^3$. We set the *blur* parameter to $\sigma = 0.1$ mm and the *reach* parameter to $\tau = 10$ mm. For smoothing, we use the vessel-preserving anisotropic kernel $k(x, y)$ of Sec. A.2 with a kernel scale $s_{\text{local}} = 8$ mm. In order to get a closer fit to the target, we apply this post-processing step twice for all lung registration results.

Post-processing	D-RobOT (raw) mm ↓	D-RobOT (spline) mm ↓	D-RobOT (LDDMM) mm ↓
Without post-processing	3.46 (3.13)	4.45 (4.10)	3.33 (3.09)
NN projection	3.33 (2.96)	3.32 (3.05)	3.31 (2.96)
RobOT	3.35 (3.04)	3.18 (3.01)	3.19 (3.01)
NN projection + Smoothing	**3.32 (2.95)**	3.08 (2.82)	2.95 (2.64)
RobOT + Smoothing	3.33 (2.99)	**2.94 (2.71)**	**2.83 (2.40)**

Table 4: **Ablation study on the post-processing module.** We assess the influence of the fine-tuning step in the D-RobOT model. As an addendum to Tab. 1, we display the Root Mean Squared Error in millimeters (median of the 10 subjects in parentheses) for expert-annotated landmarks on the 10 Dirlab lung pairs, that we use as a test set for PVT1010. The first row corresponds to the output of the deep registration module, without any fine-tuning. The second row corresponds to a nearest neighbor projection on the target point cloud. The third row corresponds to the "raw" RobOT matching of Eq. 4, extrapolated to the landmarks using a Nadaraya-Watson spline with an isotropic Gaussian kernel of deviation $\sigma = 0.5$ mm. In the fourth and fifth rows, we use a vessel-preserving anisotropic kernel to smooth a nearest neighbor projection (fourth row) and our RobOT matching (fifth row).

Method	Pre-alignment	Deep prediction	Post-processing	Number of points	Number of control points	RMSE mm ↓
Input data		None		60k	—-	23.32
Affine	✓	None		60k	—-	10.31
FlowNet3D		FLOW		30k	4,096	8.20
	✓	FLOW		30k	4,096	7.08
	✓	FLOW	✓	30k	4,096	6.51
D-RobOT	✓	PWC*		30k	4,096	4.10
		PWC*		60k	4,096	4.64
	✓	PWC*		60k	4,096	3.82
	✓	PWC*		60k	8,192	3.33
	✓	PWC*	✓	60k	8,192	**2.83**

Table 5: **Ablation study for the D-RobOT model (with LDDMM deformations) on the lung registration task.** We benchmark the Root Mean Squared Error on the 10×300 pairs of DirLab landmarks for a wide range of configurations. We investigate the influence of affine S-RobOT pre-alignment (column 2); spline S-RobOT fine-tuning (column 4); the sampling rate for the input data (column 5); and the number of control points for the LDDMM deformation model (column 6). As a backbone architecture for the deep prediction network, we use a FlowNet3D (rows 3-5) and the modified PointPWC-Net of Fig. 10 (rows 6-10).

	Method	Prealign	Post	EPE3D cm ↓	Acc3DS % ↑	Acc3DR % ↑	Outliers3D % ↓	EPE2D px ↓	Acc2D % ↑
8192 points	PWC [126]			6.78	78.58	90.30	23.61	2.6484	82.36
	PWC	✓		4.62	**83.24**	94.91	**19.61**	2.1411	86.82
	PWC	✓	✓	4.76	82.16	94.53	20.10	2.0716	**87.59**
	D-RobOT	✓		4.63	79.13	**95.30**	21.43	2.3930	82.93
	D-RobOT	✓	✓	**4.55**	80.27	94.85	20.65	2.1404	85.24
30k points	PWC [126]			7.90	61.33	89.75	29.36	4.0558	65.35
	PWC	✓		5.94	72.62	92.26	25.79	3.1817	72.82
	PWC	✓	✓	3.99	86.44	95.02	18.59	1.9592	86.09
	D-RobOT	✓		2.94	92.86	98.59	16.05	1.5733	91.86
	D-RobOT	✓	✓	**2.15**	**95.74**	**98.96**	**12.93**	**1.1118**	**95.66**

Table 6: **Ablation study on the modules of D-RobOT (with spline deformations)** and the influence of the point sampling rate, performed on the **57 largest scenes from the Kitti dataset**. As detailed in Suppl. A.4.3, these results correspond to average values on the 57 (out of 142) pairs of 3D scenes that are sampled with more than 30k points per frame in the original Kitti dataset. This allows us to focus our evaluation on the most challenging 3D scenes, with under-sampling artifacts: as expected, performance is lower than in the "full" Kitti benchmark of Fig. 4.

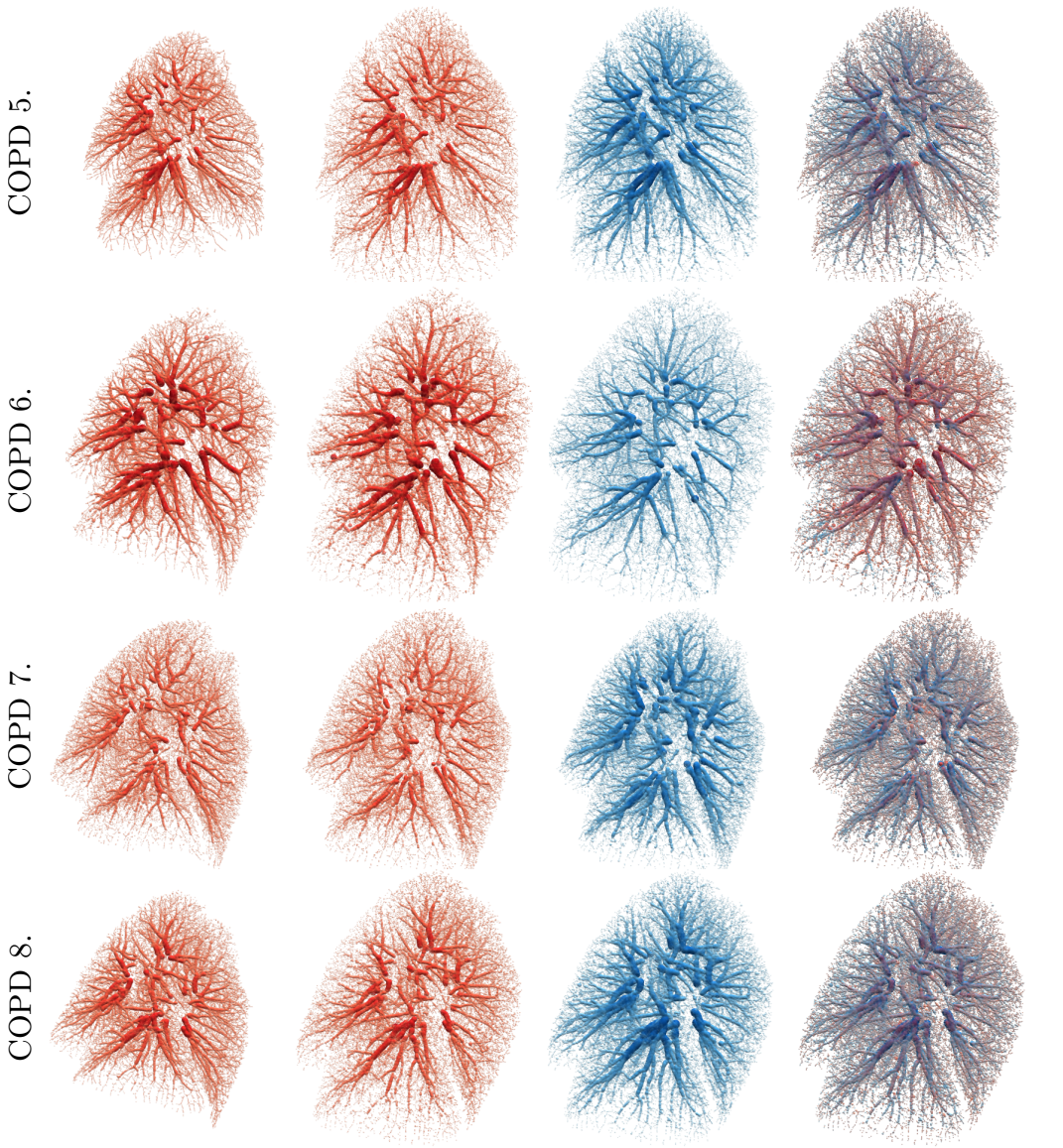
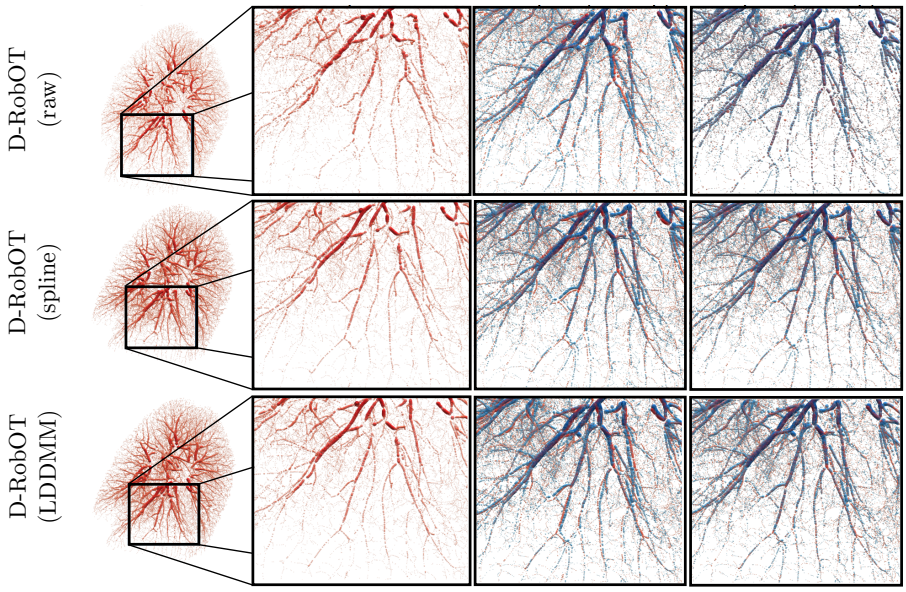
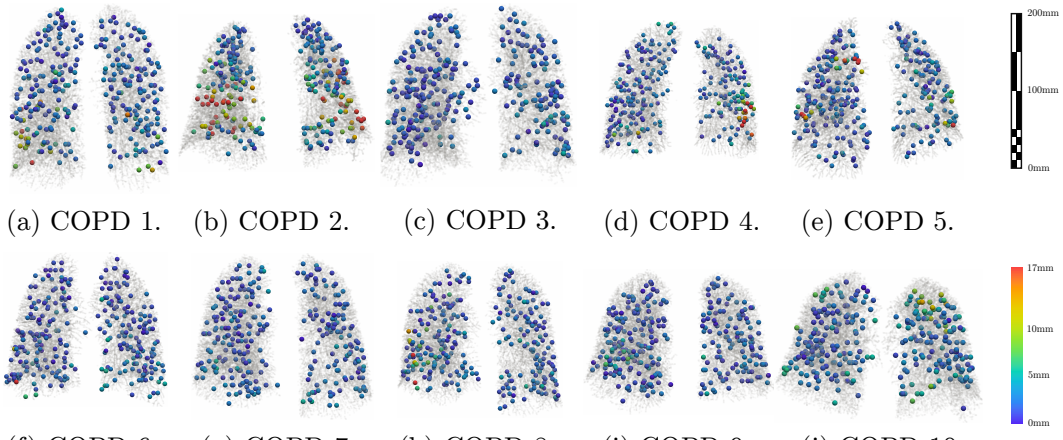


Figure 11: **Additional results for lung vascular trees.** Columns refer to: (a) the source shape (expiration); (b) the registration result from D-RobOT using the LDDMM deformation model; (c) the target shape (inspiration); (d) an overlap between the D-RobOT registration and the target. Each row corresponds to a patient, with names that refer to case IDs in the original DirLab dataset.



(a) Output of the deep registration (step 2). (b) Output of step 2, target. (c) Output of step 3, target.

Figure 12: **D-RobOT results for different deformation models Morph** : $(\theta, x_i) \mapsto \hat{y}_i$**.** (a) The first two columns correspond to the output of our deep registration module (without post-processing); (b) the third column also includes the target shape, displayed as a blue point cloud; (c) the fourth column showcases our final result, with RobOT-based post-processing. As detailed in Suppl. A.4.2, we observe that: (i) the spline and LDDMM models produce smooth registration results; (ii) before the RobOT post-processing step, LDDMM provides a better fit than the spline model; (iii) after post-processing, all three deformation models produce "sharp" results, with smoothness guarantees for the spline and LDDMM models.



(a) COPD 1. (b) COPD 2. (c) COPD 3. (d) COPD 4. (e) COPD 5.

(f) COPD 6. (g) COPD 7. (h) COPD 8. (i) COPD 9. (j) COPD 10.

Figure 13: **Distributions of the DirLab-COPDGene errors for the 300 expert-annotated corresponding landmarks.** Each image refers to a patient, with a title that refers to the case ID in the original DirLab dataset. We use color to display the registration error with our best performing model (D-RobOT with LDDMM deformations) on the source landmarks. The COPD 2 patient is a clear outlier in our experiment as in all previous studies on the original volumetric data (`https://www.dir-lab.com/Results.html`) – this may be due to annotation errors. On the other patients, most of the large errors occur in boundary regions where acquisition artifacts induce inconsistencies between the source and target point clouds: small vessels are harder to catch in the original 3D volume.

A.4.3 Experiments on Kitti

We now provide details for our experiments on scene flow estimation, which are illustrated in Fig. 14. Notably, we discuss the influence of the sampling rate for the source and target point clouds and provide an extensive ablation study for the modules of the D-RobOT model.

Influence of the sampling rate. We follow the pre-processing strategy of PointPWC-Net [126]:

1. We train on points whose depth with respect to the acquisition device is smaller than 35 m.
2. We sample points from both of the source and target frames at random, in a non-corresponding manner.

In the main manuscript, we report results when the source and target point clouds are sampled with either 8,192 or 30k points at a time. We note that in a similar situation, when sampling 32,768 points from each frame, HPLFlowNet [49] reported an average EPE3D value of 10.87 cm over all 142 pairs of the Kitti dataset. In comparison, our D-RobOT model reaches a much higher accuracy (EPE3D $=$ 2.23 cm) with only 30,000 points per frame.

Working with the 57 largest scenes from Kitti. We note that the number of points per 3D frame varies widely in the Kitti dataset. To investigate the potential negative effects from under-sampling, we evaluate our models separately on pairs where both frames contain more than 30k points: 57 out of the 142 Kitti scene pairs meet this requirement. When dealing with those 57 "most populated" frames, sub-sampling the scene to 30k points induces a net loss of geometric information.

We display our results for this subset of Kitti in Tab. 6: unsurprisingly, performance is lower than in Fig. 4, which reports results on the full Kitti dataset and thus includes the 85 pairs of frames for which 30k points are enough to work at full resolution. Nevertheless, our conclusions remain consistent: the D-RobOT method outperforms PointPWC-Net [126] for both sampling rates.

Ablation study. Going further, we analyze the contributions of the pre-alignment and post-processing modules in the D-RobOT model. Our results are summarized in Tab. 6 and can be described as follows:

- The **pre-alignment module** improves results for **both sampling rates**. Since rigid transformations have few degrees of freedom, the estimation of Eq. (5) does not rely on fine details.
- In sharp contrast, the **post-processing module** only improves results for **high-resolution points**. In practice, our RobOT-based post-processing behaves as a fast local fitting to the target point cloud. In the upper half of the table, the target point cloud and the set of "ground truth" final positions for the source points have small overlap due to the small number of sampling points for these complex scenes: a local "pixel-perfect" fine-tuning is detrimental to performance. In the lower half of the table, we increase the sampling rate to 30k points per frame: the target point cloud and the set of ground-truth final positions for the source points have a much higher overlap. This allows the RobOT post-processing to increase the accuracy of the full model.

Experiments. We now provide detailed hyperparameters for the D-RobOT architecture in our Kitti experiments. As detailed in Sec. 3.2, the full model applies successively: a rigid S-RobOT pre-alignment; a deep non-parametric registration; and a spline S-RobOT post-processing.

1. Rigid registration. For rigid pre-alignment, we use the S-RobOT model of Eq. (5) with raw (x, y, z) coordinates as input features in $\mathbb{R}^3$. The vessel radii are taken into account as point weights α_i and β_j. In the RobOT problem of Eq. (2), we set the *blur* parameter to $\sigma = 1$ m and the *reach* parameter to $\tau = +\infty$ (balanced OT with strong constraints on the marginals).

2. Deep non-parametric registration. We use the prediction architecture and deformation models of Suppl. A.4.2. Our spline deformation model is based on a multi-Gaussian-kernel with standard deviations $\{20, 40, 60\}$ cm as and weights $\{0.2, 0.3, 0.5\}$. With the coordinates of points the x and y in centimeters, this corresponds to the positive definite kernel function:

$$\begin{aligned} k(x,y) \;=\; & 0.2 \cdot \exp\big[-\|x-y\|^2_{\mathbb{R}^3}/(2\cdot 20^2)\big] \;+\; 0.3 \cdot \exp\big[-\|x-y\|^2_{\mathbb{R}^3}/(2\cdot 40^2)\big] \\ & +\; 0.5 \cdot \exp\big[-\|x-y\|^2_{\mathbb{R}^3}/(2\cdot 60^2)\big]\,. \end{aligned} \tag{24}$$

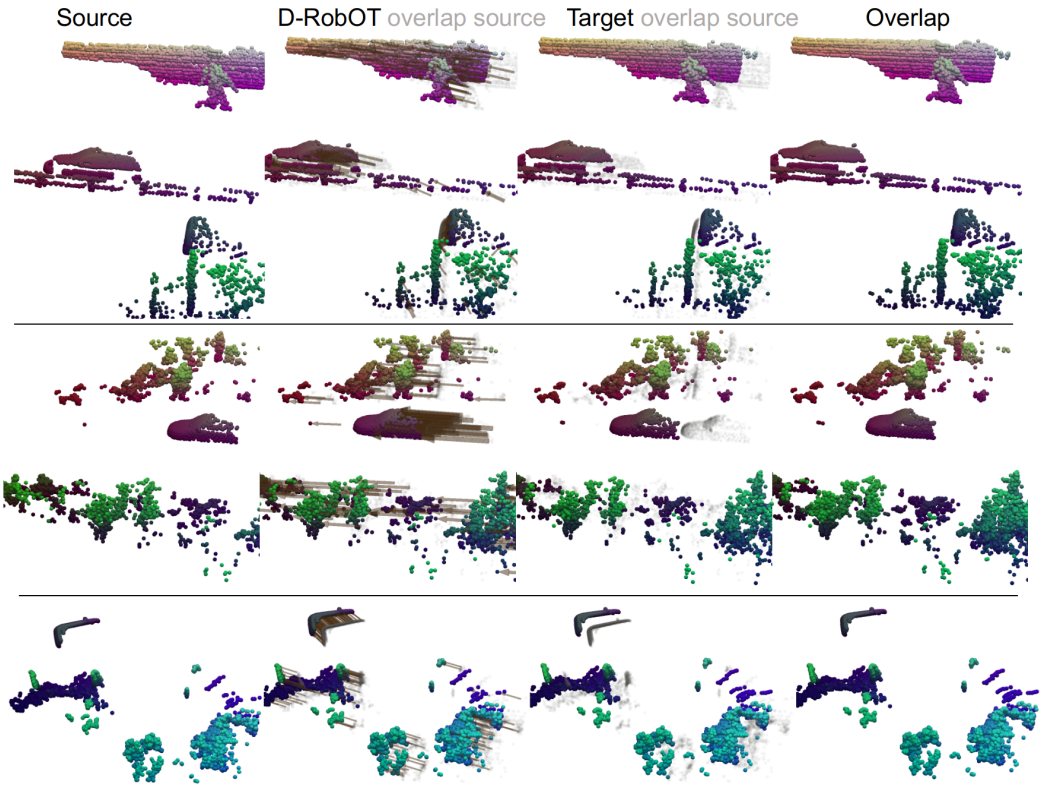


Figure 14: **Additional results for scene flow estimation on Kitti.** From left to right, the columns refer to: the source shape; the registration result for D-RobOT with a spline deformation model (the source point cloud is displayed in gray in the background, flows are displayed as brown arrows); the target shape (the source point cloud is displayed in gray in the background); and the D-RobOT result overlapped with the target. Each row corresponds to a single pair of 3D frames.

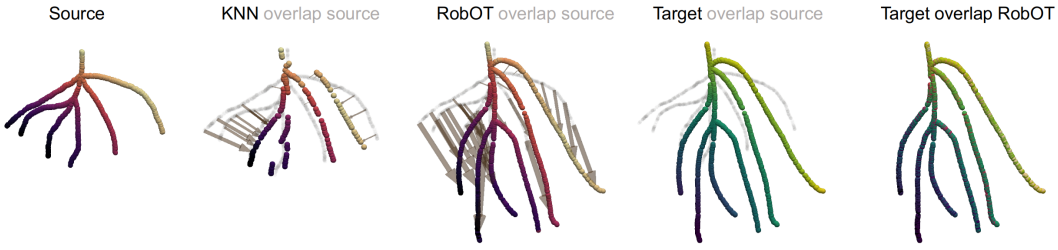


Figure 15: **RobOT matching vs Nearest Neighbor projection on a synthetic local vessel tree.** From left to right, the columns refer to: the source shape; the registration result for NN projection and RobOT followed by a local smoother (the source point cloud is displayed in gray in the background, flows are displayed as gray arrows); the target shape (the source point cloud is displayed in gray in the background); and the D-RobOT result overlapped with the target.

3. Postprocessing. To fine-tune our registration, we use the spline S-RobOT deformation of Eq. (7) with (x, y, z) coordinates as input features in $\mathbb{R}^3$. We set the *blur* parameter to $\sigma = 5$ cm and the *reach* parameter to $\tau = 80$ cm. For smoothing, we use the vessel-preserving anisotropic kernel $k(x, y)$ of Sec. A.2 with a kernel scale $s_{\text{local}} = 80$ cm.

A.5 RobOT matching vs nearest neighbor projection.

As detailed in Sec. 2.2, we use the RobOT matching of Eq. (4) as a plug-in replacement for nearest neighbor (NN) projection. At an affordable computational cost, our RobOT layer enforces a local conservation of the point density: this geometric prior is relevant for many registration tasks and mitigates accumulation artifacts that are commonly found in nearest neighbor projections [40, 35]. In accordance with the theory, we thus observe that RobOT matching outperforms nearest neighbor projection in a wide range of settings.

Local translations. When the point features p_i and q_j correspond to the (x, y, z) coordinates in $\mathbb{R}^3$, a key property of the Monge-Brenier map that is defined by Eq. (4) is that it perfectly retrieves translations and dilations [15, 87, 35]. Scene flow estimation is often close to this best case scenario for RobOT theory: after we remove points that correspond to the ground (a common pre-processing for this task), most of the displacements can be explained as local translations and small rotations of solid objects (e.g. cars or trees) in the frame of reference of the acquisition device.

On the Kitti benchmark of Fig. 4, we observe that a simple RobOT matching already performs very well for 3D scene flow estimation. Remarkably, and without any training, **vanilla RobOT matching** on high-resolution point clouds (30k points per frame) **outperforms most pre-existing deep learning methods** on medium-resolution point clouds (8,192 points per frame) in terms of speed, memory usage and accuracy. This is strong evidence that geometric methods deserve more attention from the computer vision community.

Complex structures. Going further, the mass distribution constraint of Eq. (2) is useful to register **complex shapes with appendices and branches**. This property has been discussed in depth in previous works: we refer to e.g. [36] for the matching of five-fingered hand surfaces. The (soft) constraints on the marginals of the transport plan promote a bijective matching of the fingers, preventing the thumb and the index of the source shape from being both projected on the thumb of the target.

In Fig. 15, we provide a similar example for the local registration of branching vessels. We create a synthetic pair of source and target vessels with $\mathrm{N} = 1,000$ and $\mathrm{M} = 1,200$ points respectively. We normalize their (x, y, z) coordinates to be contained in the unit cube $[0, 1]^3$ and assign uniform weights $\alpha_i = 1$ and $\beta_j = 1$ to each point – we work in a realistic unbalanced scenario. We compute the raw NN matching using a simple nearest neighbor projection from the source onto the target in $\mathbb{R}^3$. For the RobOT matching, we also use raw (x, y, z) coordinates and the squared Euclidean metric; we set the *blur* parameter to 0.0005 units in $\mathbb{R}^3$ and the *reach* parameter to 1 unit (unbalanced OT). As detailed in Suppl. A.2, we regularize both of the NN and RobOT matchings using an anisotropic muti-Gaussian-kernel with standard deviations $\{0.03, 0.05, 0.07\}$ and weights $\{0.2, 0.3, 0.5\}$.

As a post-processing step. As discussed above, RobOT matching is good at handling local translations, dilations and small free-form deformations. We propose to use it as a fast post-processing step in our D-RobOT architecture, presented in Sec. 3.2. In order to validate its utility, we perform an ablation study on the PVT1010/DirLab dataset for lung registration.

We benchmark increasingly suitable fine-tuning layers: no fine-tuning; nearest neighbor projection; RobOT matching; nearest neighbor projection with vessel-preserving smoothing; RobOT matching with vessel-preserving smoothing. Our results are detailed in Tab. 4: when used in combination with a smooth deformation model, these layers result in an increasingly higher accuracy. This confirms the main findings of our work: including **sensible geometric priors** in a modular deep learning architecture is the key to reliable state-of-the-art performance.

A.6 Computational Resources

For the evaluation of time and memory cost in Fig. 4, we compare all models on a Ubuntu server with a single 24GB NVIDIA Quadro RTX 6000 graphics processing unit (GPU) and a 10-core Intel(R) Xeon(R) Silver 4114 CPU @ 2.20GHz. We trained all of our networks on a single 24GB

NVIDIA RTX 3090 GPU, except for the training of PointPWC-Net with 30k sampled points that was performed on a 48GB NVIDIA RTX A6000 GPU.

A.7 Potential for negative societal impact

Point cloud registration is a fundamental low-level task in computer vision and computer graphics. As a consequence, negative societal impact of our work may arise in a wide range of use cases. A first example is that of point cloud registration for autonomous driving: algorithm failure could lead to incorrect driving decisions. Similar concerns may arise when using these approaches to register facial scans for the purpose of person identification. In general, while our manuscript demonstrates improved performance and robustness over competing approaches, the possibility for registration failures should always be considered when applying our models in safety critical environments.