Intraoperative 2D/3D Image Registration via Differentiable X-ray Rendering

Supplementary Material

A. Lie Theory of SE(3)

We present a brief overview of the Lie theory of $\mathbf{SE}(3)$ as it pertains to our method. A camera pose $\mathbf{T} \in \mathbf{SE}(3)$ can be represented as the matrix

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \in \mathbf{SE}(3)\,, \tag{13}$$

where $\mathbf{R} \in \mathbf{SO}(3)$ is a 3×3 rotation matrix and $\mathbf{t} \in \mathbb{R}^3$ is a translation. The Lie group $\mathbf{SE}(3)$ corresponds to the Lie algebra $\mathfrak{se}(3)$, spanned by six basis vectors representing either infinitesimal rotations or translations along a specific axis [9]:

$$\begin{gathered} \mathbf{G}_1 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \mathbf{G}_2 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \mathbf{G}_3 = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \\ \mathbf{G}_4 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \mathbf{G}_5 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \mathbf{G}_6 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{gathered} \tag{14}$$

That is, any transformation $\mathbf{T} \in \mathbf{SE}(3)$ can be represented as a 6-vector $\mathbf{v} = \begin{bmatrix} \boldsymbol{\omega} & \mathbf{u} \end{bmatrix}^T$ corresponding to the matrix

$$\log \mathbf{T} = \omega_1 \mathbf{G}_1 + \omega_2 \mathbf{G}_2 + \omega_3 \mathbf{G}_3 + u_1 \mathbf{G}_4 + u_2 \mathbf{G}_5 + u_3 \mathbf{G}_6 = \begin{bmatrix} 0 & -\omega_3 & \omega_2 & u_1 \\ \omega_3 & 0 & -\omega_1 & u_2 \\ -\omega_2 & \omega_1 & 0 & u_3 \\ 0 & 0 & 0 & 0 \end{bmatrix} \in \mathfrak{se}(3)\,, \tag{15}$$

where $\log : \mathbf{SE}(3) \to \mathfrak{se}(3)$ is the logarithmic map. In the literature, it is common to "vee" operator $(\cdot)^\vee$ to relate the matrix $\log \mathbf{T}$ to its vector representation $\mathbf{v}$. While most authors will write

$$(\log \mathbf{T})^\vee = \begin{bmatrix} 0 & -\omega_3 & \omega_2 & u_1 \\ \omega_3 & 0 & -\omega_1 & u_2 \\ -\omega_2 & \omega_1 & 0 & u_3 \\ 0 & 0 & 0 & 0 \end{bmatrix}^\vee = \begin{bmatrix} \omega_1 & \omega_2 & \omega_3 & u_1 & u_2 & u_3 \end{bmatrix}^T = \mathbf{v}\,, \tag{16}$$

we let $\log(\cdot)$ implicitly denote this vectorization, in a slight abuse of notation. The matrix exponential provides the map $\exp : \mathfrak{se}(3) \to \mathbf{SE}(3)$. While it is common to use the analogous "hat" operator to represent $\mathbf{T} = \exp(\mathbf{v}^\wedge)$, we also treat this as implied. By grouping even and odd powers in their Taylor expansions, the logarithmic and exponential maps can be calculated in closed form [16]. The implementations we use are available in PyTorch3D [43].

An important interpretation appears when observing the equation for the exp map:

$$\mathbf{T} = \exp(\mathbf{v}^\wedge) = \begin{bmatrix} \exp \boldsymbol{\omega}^\wedge & \boldsymbol{\Omega}\mathbf{u} \\ 0 & 1 \end{bmatrix} \quad \text{where} \quad \boldsymbol{\Omega} = \mathbf{I} + \left(\frac{1 - \cos\theta}{\theta^2} \right) \boldsymbol{\omega}^\wedge + \left(\frac{\theta - \sin\theta}{\theta^3} \right) (\boldsymbol{\omega}^\wedge)^2$$

and θ is given by Eq. (6). Note, the translational component in $\mathbf{T}$ is produced through a combination of $\boldsymbol{\omega}$ and $\mathbf{u}$. That is, unlike all other Euclidean parameterizations of $\mathbf{SE}(3)$ considered in Table 3, the translational and rotational components are *not* independent when represented in $\mathfrak{se}(3)$. It is possible that the improved performance of $\mathfrak{se}(3)$ is due to the coupling of rotational and translational components in the representation. Finally, note that the logarithmic map on $\mathbf{SO}(3)$, *i.e.* $\log(\mathbf{R}) = \boldsymbol{\omega}$, is equivalent to the axis-angle representation [16]. Therefore, the parameterization in the second row of Table 3 is the same as $\mathfrak{so}(3) \times \mathbb{R}^3$.

B. Visualizing Loss Landscapes for Multiple Image Similarity Metrics

In Figure 7, we compare the loss landscapes of multiscale NCC (mNCC), local NCC (patch size of 13), and global NCC (*i.e.* the whole image is a single patch). Loss landscapes are generated by measuring the similarity between the ground truth X-ray and synthetic X-rays rendered at perturbations from the ground truth camera pose. Local NCC has a sharp peak at the ground truth camera pose, however, its landscape has many local minima. In contrast, global NCC is much smoother, but has a less defined peak. Averaging local and global NCC yields mNCC, which has both a strong peak and a smooth landscape.

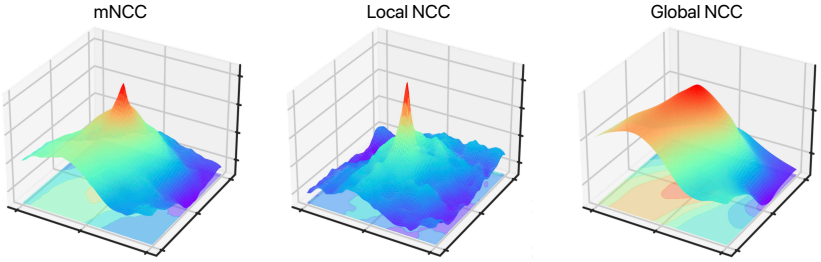

Figure 7. **Visual comparison of mNCC, local NCC, and global NCC.**

In Figure 8, we compare mNCC to the following image similarity metrics: local NCC, global NCC, gradient NCC [23], SSIM, multiscale SSIM (mSSIM), PSNR, negative MAE, and negative MSE. For the 6 d.o.f. in a camera pose, rotational perturbations are jointly sampled from ±1 rad and translational perturbations are jointly sampled from ±100 mm.

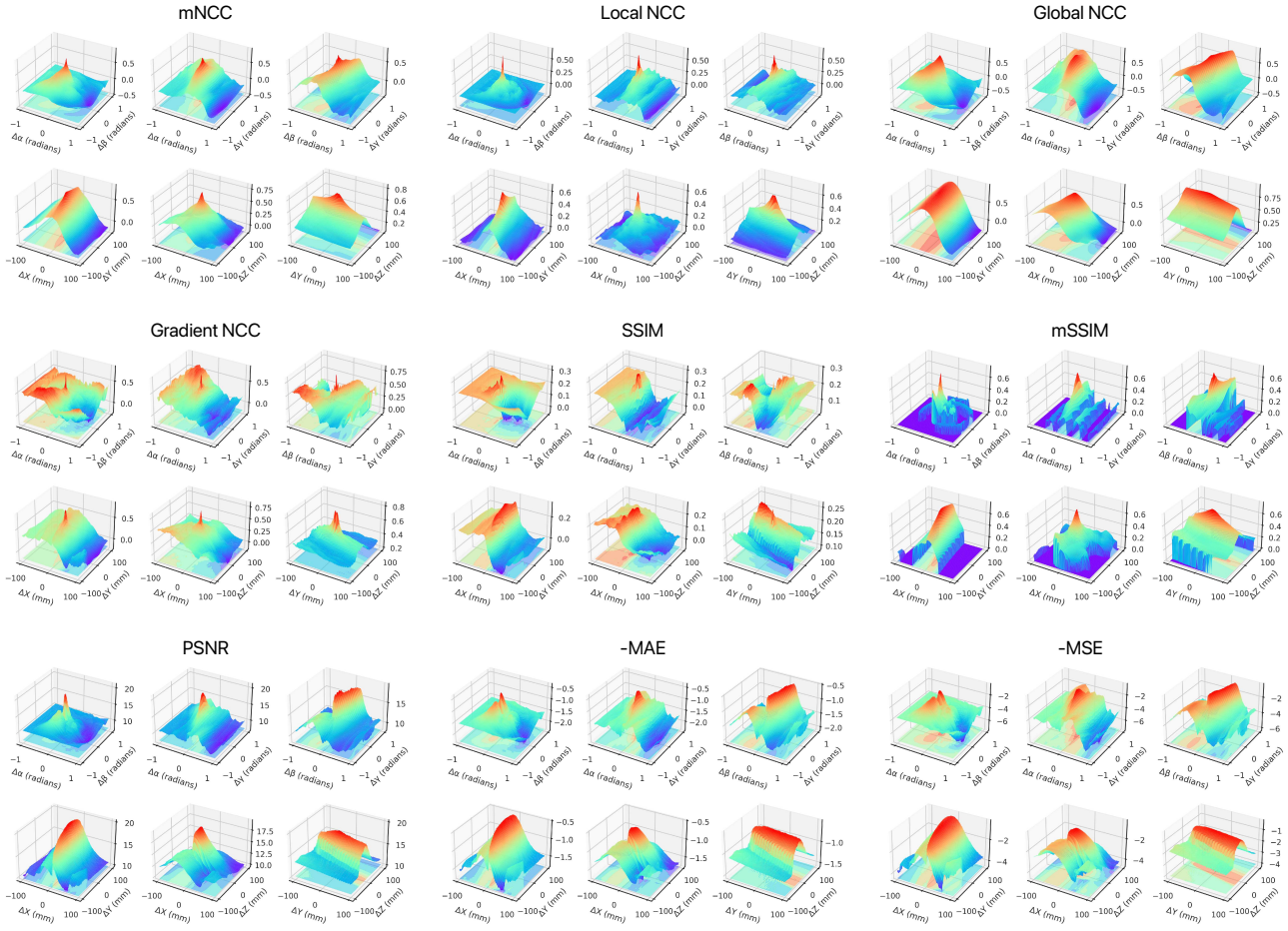

Figure 8. **Visualization of image-based loss landscapes.** mNCC is the most amenable to gradient-based optimization.

C. Visualizing Sparse Multiscale NCC

Figure 9 visualizes the sparse patch-wise differentiable rendering procedure used to compute sparse mNCC. In DiffPose, the camera pose of a real X-ray is estimated using a CNN. Along with the regressed pose, activations at the final convolutional layer are stored. Visualizing these activations shows that the network mostly uses the location of bony structures (*e.g.* spine, pelvis, hips, *etc.*) to estimate the pose. This activation map is resized to match the original X-ray and used to define a probability distribution over the pixels in the image. For each iteration, a fixed number of patch centers are sampled from this distribution. Specifying the patch size defines the sparse subset of pixels in the detector plane that need to be rendered. In practice, we render 100 patches with a patch size of 13. Since we downsample real X-rays to 256×256 pixels, when using sparse rendering, we render at most $(100 \cdot 13^2)/256^2 \approx 25\%$ of the pixels in the image. Note, this is an upper bound because patches can overlap. For ease of visualization, we also render 750 patches with a patch size of 13. Finally, sparse mNCC is computed by averaging the local NCC over all rendered patches and the global NCC computed all rendered pixels. Note that sparse mNCC is a biased estimate of mNCC, and this approach is closely related to prior work that used sparse image patches to estimate mutual information between images [58].

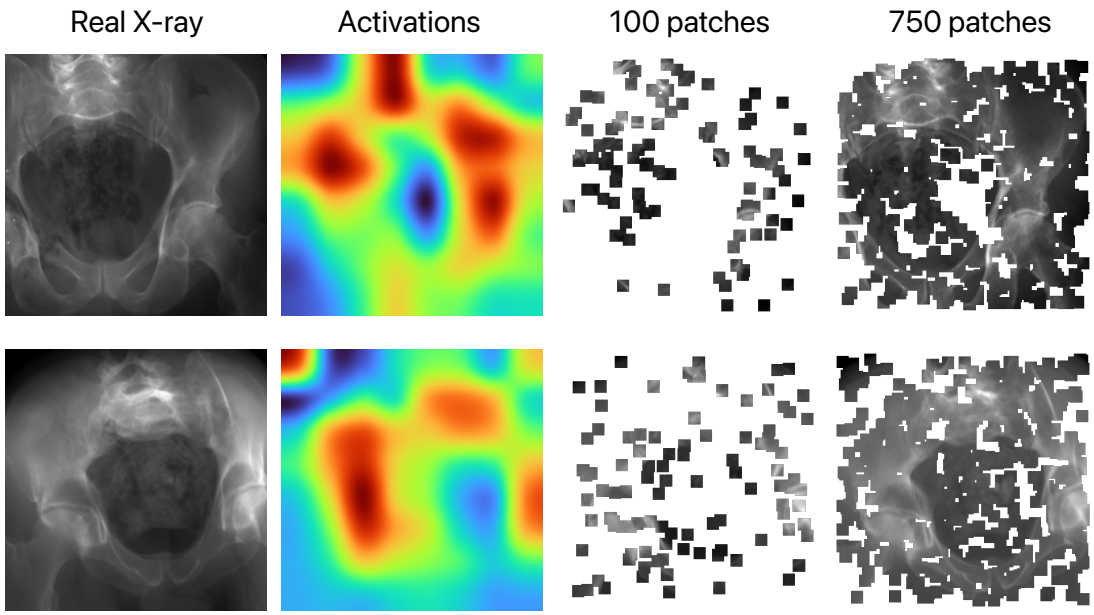

Figure 9. **Visualization of sparse mNCC.** We compute mNCC using sparse image patches rendered around anatomical structures that drive 2D/3D registration. In our experiments, we compute sparse mNCC using 100 patches with a patch size of 13.

The accuracy and speed of sparse mNCC are provided in Table 5. In an alternative formulation of sparse mNCC, we can ignore the distribution defined by the network's activations and instead sample patch centers uniformly at random over the image. We find that this strategy, which we term unbiased sparse mNCC, performs nearly as well as the original sparse mNCC. Finally, we also compare against mNCC. While mNCC was the most accurate image similarity metric and had the highest success rate, it is also the slowest method to compute. On average for one iteration, it is $3.5\times$ faster to render and compute sparse mNCC with 100 patches (13×13) than mNCC over all patches. However, since sparse mNCC is a noisy estimate of mNCC computed over all image patches, it takes more iterations to converge than standard mNCC.

Table 5. Comparison of sparse mNCC to other mNCC variants.

	SMSR ↑	mTRE (mm) ↓	Time (s) ↓
Sparse mNCC	87%	0.9 ± 2.8	2.2 ± 1.2
Unbiased Sparse mNCC	86%	1.0 ± 2.4	**2.1 ± 1.3**
mNCC	**89%**	**0.8 ± 2.1**	3.9 ± 1.6

D. Converting Imaging System Coordinates to DiffDRR Camera Coordinates

Parsing intrinsic matrices. Intrinsic matrices provided in the DeepFluoro and Ljubljana datasets can be decomposed as

$$\begin{bmatrix} f_x & 0 & x_0' \\ 0 & f_y & y_0' \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -1/\Delta X & 0 & W/2 \\ 0 & -1/\Delta Y & H/2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{17}$$

where (f_x, f_y) are the focal lengths in the x- and y-directions (in units of pixels), (x_0', y_0') is the camera's principal point (in units of pixels), (H, W) is the height and width of the detector plane (in units of pixels), and $(\Delta X, \Delta Y)$ are the x- and y-direction pixel spacings (in units of length per pixel). From these known parameters, the focal length of the X-ray scanner (in units of length) can be expressed as

$$f = \frac{f_x \Delta X + f_y \Delta Y}{2}, \tag{18}$$

and the principal point also expressed in units of length is

$$x_0 = \Delta X \left(\frac{W}{2} - x_0' \right) \tag{19}$$

$$y_0 = \Delta Y \left(\frac{H}{2} - y_0' \right). \tag{20}$$

The intrinsic parameters f, ΔX, ΔY, H, W, x_0, and y_0 are needed to define the detector plane in DiffDRR [22].

Parsing extrinsic matrices. Extrinsic matrices in the DeepFluoro and Ljubljana datasets assume that the camera is initialized at the origin and pointing in the negative z-direction. However, DiffDRR initializes the camera at $(f/2, 0, 0)$ pointed towards the negative x-direction. To transform a camera pose $\mathbf{T} \in \mathbf{SE}(3)$ from DeepFluoro/Ljubljana's coordinate system to DiffDRR's coordinate system, we use the following conversion:

$$\tilde{\mathbf{T}} = \mathbf{T}^{-1} \begin{bmatrix} 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -f/2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{21}$$

When passed to DiffDRR along with the scanner's intrinsic parameters, the camera pose $\tilde{\mathbf{T}}$ renders synthetic X-rays using the same geometry as the real-world imaging system (see Figure 3). Transformations of camera poses, along with conversions between multiple parameterizations of $\mathbf{SO}(3)$ and $\mathbf{SE}(3)$, are handled using PyTorch3D [43].

E. Derivation of the Image Formation Model

For completeness, we present a single, detailed derivation of the X-ray image formation model and preprocessing steps described in many places throughout the main text. Let $\mathbf{r}(\alpha) = \mathbf{s} + \alpha(\mathbf{t} - \mathbf{s})$ be a ray originating at the radiation source $\mathbf{s} \in \mathbb{R}^3$ and terminating at the target pixel on the detector plane $\mathbf{t} \in \mathbb{R}^3$. We are interested in modeling the attenuation of $\mathbf{r}$ as it travels through the anatomical volume $\mathbf{V} : \mathbb{R}^3 \to \mathbb{R}$. For every point $\mathbf{x}$ in 3D space, $\mathbf{V}(\mathbf{x})$ returns a *linear attenuation coefficient* that characterizes how much intensity $\mathbf{r}$ loses to $\mathbf{V}$ when it travels through $\mathbf{x}$. A large coefficient denotes that $\mathbf{x}$ comprises a material that greatly attenuates $\mathbf{r}$ by absorbing a large amount of its intensity, while a small coefficient represents a material that is easily penetrated. We model points in empty space as having a linear attenuation coefficient of zero. The attenuated intensity of $\mathbf{r}$ after it has passed through every point on its path, as governed by the Beer-Lambert law [49], is

$$I(\mathbf{p}) = I_0 \exp\left(- \int_{\mathbf{x} \in \mathbf{r}} \mathbf{V}(\mathbf{x}) \, \mathrm{d}\mathbf{x} \right), \tag{22}$$

where I_0 is the initial intensity of every X-ray radiating from $\mathbf{s}$.

Instead of modeling the attenuated intensity of $\mathbf{r}$, it is both equivalent and simpler to model the amount of energy absorbed by $\mathbf{V}$. To this end, we only consider the integral in Eq. (22) and model the absorption $I_\mu(\mathbf{p}) = \log I_0 - \log I(\mathbf{p})$, which is

inversely proportional to $I(\mathbf{p})$. If I_0 is unknown, we can estimate it by computing the maximum value over all pixels in a set of X-rays acquired using a scanner with the consistent parameters. Then, Eq. (22) can be expressed as

$$
\begin{aligned}
I_\mu(\mathbf{p}) &= \log I_0 - \log I(\mathbf{p}) && (23)\\
&= \int_{\mathbf{x}\in\mathbf{r}} \mathbf{V}(\mathbf{x})\,\mathrm{d}\mathbf{x} && (24)\\
&= \int_0^1 \mathbf{V}\big(\mathbf{r}(\alpha)\big)\|\mathbf{r}'(\alpha)\|\,\mathrm{d}\alpha && (25)\\
&= \|\mathbf{p}-\mathbf{s}\| \int_0^1 \mathbf{V}\big(\mathbf{s}+\alpha(\mathbf{p}-\mathbf{s})\big)\,\mathrm{d}\alpha && (26)\\
&\approx \|\mathbf{p}-\mathbf{s}\| \sum_{m=1}^{M-1} \mathbf{V}\left[\mathbf{s}+\frac{\alpha_{m+1}+\alpha_m}{2}(\mathbf{p}-\mathbf{s})\right](\alpha_{m+1}-\alpha_m)\,, && (27)
\end{aligned}
$$

where, in the last step, we approximate $\mathbf{V}$ with a preprocessed CT volume (a voxel grid of linear attenuation coefficients estimated from the original Hounsfield units). This approximation results in a discretization of the line integral over the voxel grid—note the similarities between Eq. (26) and the Eq. (27). The set $\{\alpha_1, \ldots, \alpha_M\}$ parameterizes all the intersections of $\mathbf{r}$ with the orthogonal planes comprising the CT volume, and $\mathbf{V}[\cdot]$ is an indexing operation that returns the linear attenuation coefficient of the voxel within which a 3D point is located. Siddon's method [47] provides a method for efficiently computing the intersections and indexing operations in the rendering equation (27). In DiffDRR [22], Siddon's method is reformulated as a series of vectorized tensor operations, enabling Eq. (27) to be computed in a differentiable manner. Tangentially, one might wonder why the length of the ray $\|\mathbf{p}-\mathbf{s}\|$ appears in Eq. (27). Note that the SI unit of the linear attenuation coefficient is the reciprocal meter (m^{-1}). Therefore, multiplication by term $\|\mathbf{p}-\mathbf{s}\|$ serves to make the absorbance $I_\mu(\mathbf{p})$ unit-less.

F. Additional Implementation Details

Domain randomization. Inspired by previous (non-differentiable) synthetic X-ray renderers [51], we augment the contrast of synthetic X-rays by upweighting the attenuation of voxels in the CT scan corresponding to bone. To isolate the voxels corresponding to bone, we segment the CT by thresholding all voxels with Hounsfield units greater than 350. Multiplying these voxels by a bone attenuation multiplier $c \geq 1$ increases the brightness of bones relative to soft tissue. While pretraining the encoder, we randomly sample $c \sim \mathrm{Uniform}[1, 10]$. This domain randomization improves transfer from simulated to real data by diversifying the appearance of synthetic X-rays. Synthetic X-rays rendered with $c \in [1, 10]$ are shown in Figure 10.

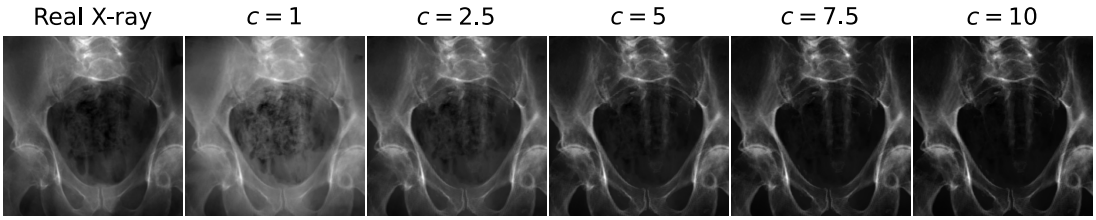

Figure 10. **Examples of domain randomization via X-ray contrast augmentation.**

Architecture. We implemented the pose regression encoder $\mathcal{E}$ using a ResNet18 backbone. As the rendering and pose regression of synthetic X-rays were performed on the same device, we were limited to a maximum batch size of eight 256×256 images. This small batch size induced instability in the running estimates of mean and variance in the batch normalization layers, leading us to replace them with group normalization layers. All encoders were trained from scratch for each patient.

Early stopping criteria. We terminate test-time optimization early if the image similarity metric does not improve by at least 0.05 for 20 iterations in a row.

Hardware. For each patient, pretraining the pose regression encoder was performed on a single NVIDIA RTX A6000 GPU. For each intraoperative X-ray, test-time optimization was performed a single GeForce RTX 2080 Ti GPU.

G. Additional Qualitative Results on the DeepFluoro Dataset

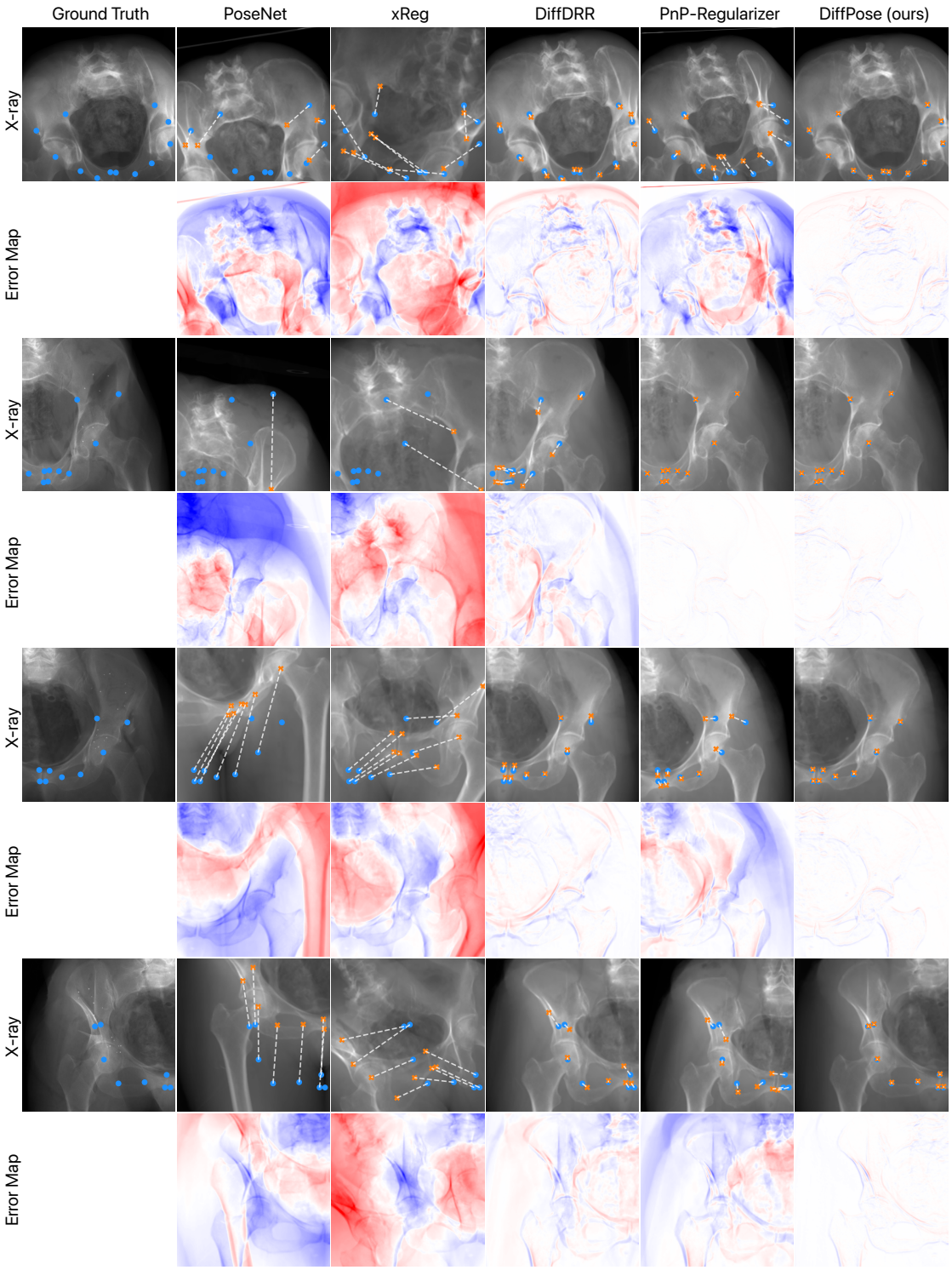

Figure 11. **Visualizations of additional qualitative results on the DeepFluoro dataset.** Fiducials projected at the ground truth camera pose are in blue, while projections at the estimated pose are in orange. White lines are drawn between corresponding fiducials.

H. Additional Qualitative Results on the Ljubljana Dataset

The Ljubljana dataset consists of 2D/3D digital subtraction angiography (DSA) images. In 2D DSA images, a subtraction step is used to remove the outline of the skull, attenuating the vasculature and increasing the signal-to-noise ratio [12]. In 3D DSA images, the signal-to-noise ratio is too low to capture the smallest blood vessels [45]. Despite missing the microvasculature, the trunks of main vasculature are sufficient to drive 2D/3D image registration with DiffPose in most cases.

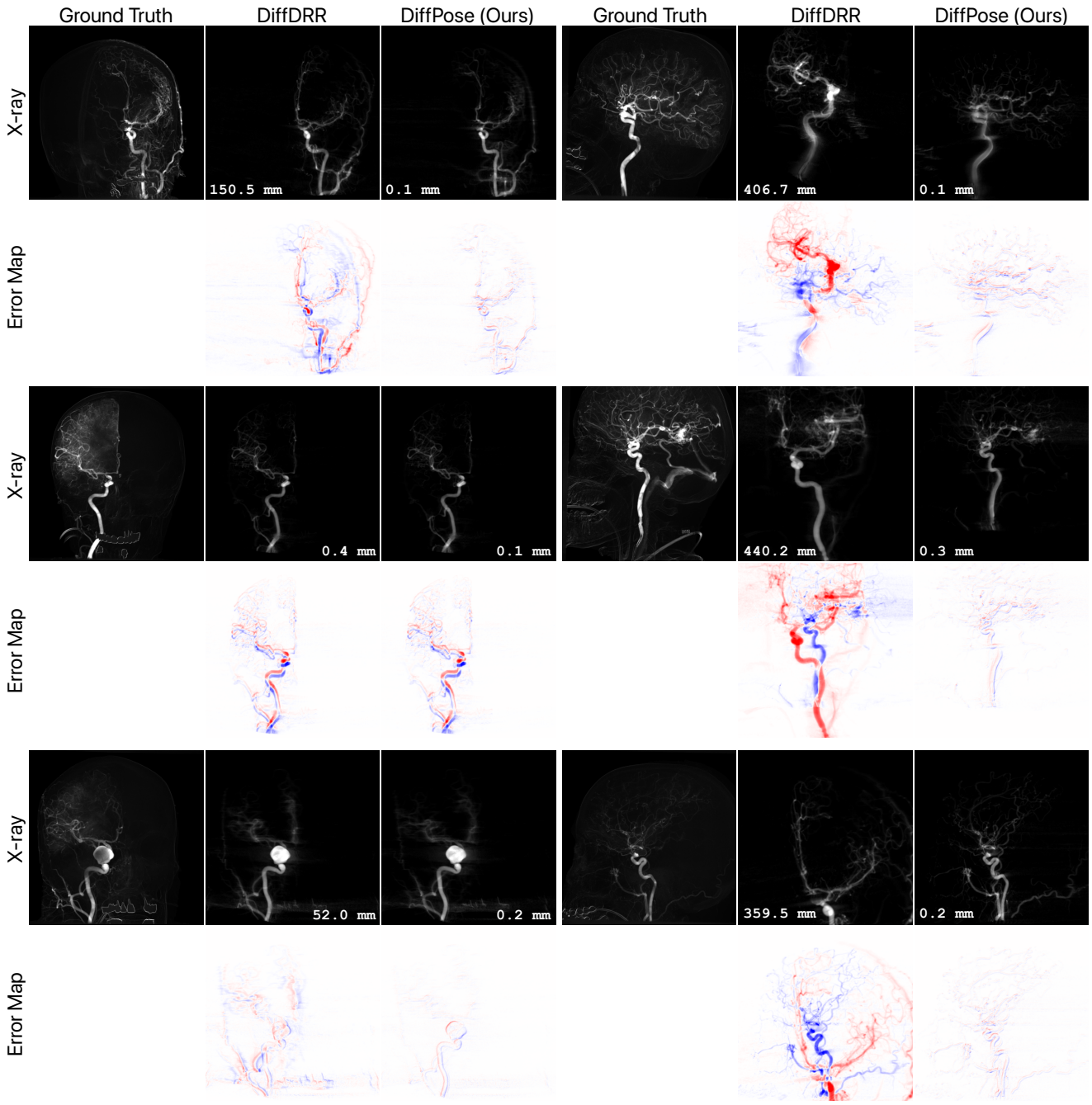

Figure 12. **Visualizations of additional qualitative results on the Ljubljana dataset.** mTRE is reported for each example.