# EDFFDNet: Towards Accurate and Efficient Unsupervised Multi-Grid Image Registration

Supplementary Material

## A. Details of Warp

### A.1. B Spline

The B-spline function $\beta^n$ of order $n$ is obtained by $n$ times convolution of the zeroth-order B-spline function [38], defined as:

$$\beta^n(x) = \beta^0(x) \underbrace{* \cdots * \beta^0(x)}_{n \text{ times}}, \tag{15}$$

where $*$ denotes the convolution operation as follows:

$$(f * h)(x) \stackrel{def}{=} \int_{-\infty}^{+\infty} f(x-t)h(t)dt, \tag{16}$$

and $\beta^0$ is the zeroth-order B-spline function:

$$\beta^0(x) = \begin{cases} 1 & : -0.5 \leq x < 0.5 \\ 0 & : \text{otherwise} \end{cases}. \tag{17}$$

Higher-order B-splines offer better smoothness but come at the cost of more complex polynomial computations. The 3rd-order B-spline (cubic B-spline) is commonly used due to its balance between smoothness and computational efficiency. Consequently, we select it as our baseline for comparison.

### A.2. Thin-Plate Spline

The Thin Plate Spline (TPS) is an interpolation method used to simulate smooth 2D deformations by minimizing a combined energy function that balances alignment accuracy and deformation smoothness. The energy function is defined as:

$$\varepsilon = \varepsilon_{\text{alignment}} + \lambda \varepsilon_{\text{distortion}}, \tag{18}$$

where $\varepsilon_{\text{alignment}}$ is the alignment energy, measuring the error between the deformed control points and their target positions; $\varepsilon_{\text{distortion}}$ is the bending energy, quantifying the smoothness of the deformation; and $\lambda$ is a balancing factor that controls the trade-off between alignment accuracy and deformation smoothness. Let $\mathcal{P} = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_N\}$ be the set of original control points and $\mathcal{P}' = \{\mathbf{p}'_1, \mathbf{p}'_2, \ldots, \mathbf{p}'_N\}$ be the set of target control points ($\mathbf{p}_i, \mathbf{p}'_i \in \mathbb{R}^{2\times 1}$). The alignment energy is formulated as:

$$\varepsilon_{\text{alignment}} = \sum_{i=1}^{N} \|\mathbf{p}'_i - T(\mathbf{p}_i)\|^2, \tag{19}$$

where $T(\cdot)$ is the warp function that maps the original points to the deformed points. The bending energy, which penalizes non-smooth deformations, is defined as:

$$\varepsilon_{\text{distortion}} = \iint_{\mathbb{R}^2} \left(\frac{\partial^2 T}{\partial x^2}\right)^2 + 2\left(\frac{\partial^2 T}{\partial x \partial y}\right)^2 + \left(\frac{\partial^2 T}{\partial y^2}\right)^2 dx\, dy. \tag{20}$$

In our implementation, we set $\lambda = 0$ to prioritize exact alignment of the control points, following the approach of previous work [32]. By minimizing the energy function in Eq. 18, the warp function $T(\mathbf{x})$ can be derived as:

$$T(\mathbf{x}) = \mathbf{C} + \mathbf{M}\mathbf{x} + \sum_{i=1}^{N} \mathbf{w}_i \phi(\|\mathbf{x} - \mathbf{p}_i\|_2), \tag{21}$$

where $\mathbf{x} \in \mathbb{R}^{2\times 1}$ is the input point coordinate, $\mathbf{C} \in \mathbb{R}^{2\times 1}$, $\mathbf{M} \in \mathbb{R}^{2\times 2}$, and $\mathbf{w}_i \in \mathbb{R}^{2\times 1}$ are transformation parameters. $\phi(r) = r^2 \log r^2$ is the radial basis function (RBF). To solve for the parameters, we formulate $N$ data constraints based on the alignment of the control points:

$$T(\mathbf{p}_i) = \mathbf{p}'_i \quad \text{for} \quad i = 1, 2, \ldots, N. \tag{22}$$

Additionally, to ensure the uniqueness and stability of the solution, we impose the following constraints [20]:

$$\sum_{i=1}^{N} \mathbf{w}_i = 0 \quad \text{and} \quad \sum_{i=1}^{N} \mathbf{w}_i \mathbf{p}_i^\top = 0. \tag{23}$$

These constraints ensure that the deformation is globally affine beyond the influence of the control points, and they are necessary to regularize the problem. Let $\mathbf{K} \in \mathbb{R}^{N\times N}$ be the matrix of RBF values between control points, defined as:

$$\mathbf{K}_{i,j} = \phi(\|\mathbf{p}_i - \mathbf{p}_j\|_2), \tag{24}$$

where $\mathbf{p}_i$ and $\mathbf{p}_j$ are the coordinates of the $i$-th and $j$-th control points, respectively. The parameters can be solved as follows:

$$\begin{bmatrix} \mathbf{C} \\ \mathbf{M} \\ \mathbf{W} \end{bmatrix} = \begin{bmatrix} \mathbf{1} & \mathbf{P} & \mathbf{K} \\ \mathbf{0} & \mathbf{0} & \mathbf{1}^\top \\ \mathbf{0} & \mathbf{0} & \mathbf{P}^\top \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{P'} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \tag{25}$$

where $\mathbf{P} \in \mathbb{R}^{N\times 2}$ is the matrix of original control point coordinates, $\mathbf{P'} \in \mathbb{R}^{N\times 2}$ is the matrix of target control point coordinates, $\mathbf{W} \in \mathbb{R}^{N\times 2}$ is the matrix of weight vectors $\mathbf{w}_i$, $\mathbf{1}$ is an $N \times 1$ column vector of ones, and $\mathbf{0}$ represents zero matrices of appropriate dimensions.

### A.3. Discussion on Local Deformation Handling

While TPS provide a powerful framework for interpolating smooth deformations, it exhibits limitations in handling localized deformations compared to B-spline Free Form Deformation (B-spline FFD) and our proposed Exponential-Decay Free Form Deformatio (EDFFD). The primary limitation of TPS lies in its global optimization nature: the warp function is derived by solving a linear system that simultaneously optimizes the parameters ($\mathbf{C}$, $\mathbf{M}$, and $\mathbf{W}$) based on all control points. TPS lacks the ability to directly compute displacements based on local properties, making it less suitable for applications requiring fine-grained local control.

In contrast, B-spline FFD and the proposed EDFFD leverage localized deformation mechanisms. B-spline FFD achieves this through piecewise polynomial basis functions with local support, ensuring that the influence of each control point is confined to a limited region. Similarly, the EDFFD, while using a globally supported basis function, introduces a rapid decay property that effectively limits the influence of each control point to its immediate neighborhood. Both methods allow for direct computation of displacements based on local control points, enabling precise and flexible manipulation of localized deformations.

## B. Details of the Network Architecture

The network architecture of EDFFDNet-2 is detailed in Table 10 and Table 11, excluding the multi-scale feature extractor which has been described in the manuscript. EDFFDNet follows the same architecture as EDFFDNet-2 but without the FFD Estimator-2 component.

## C. More Results

### C.1. Evaluation of Challenging Cases

To further demonstrate the robustness of our model, we evaluate its performance on challenging cases from the UDIS-D dataset [31], as illustrated in Fig. 6. The arrows in the figure indicate specific challenging regions where existing methods exhibit inaccurate alignment. In contrast, our approach consistently achieves precise alignment in these challenging areas, demonstrating its superior performance.

### C.2. Evaluation of Cross-Dataset Generalization

To further validate cross-dataset generalization capabilities, we provide additional qualitative results on cross-dataset scenarios [24, 25], as illustrated in Fig. 7 Fig. 8, Fig. 9, and Fig. 10. The results demonstrate that our method achieves significantly better generalization performance without requiring any fine-tuning, outperforming previous deep learning-based methods MGDH [30] and UDIS++ [32].

Table 10. Network Architecture of EDFFDNet-2. The details of the Group Linear Layer have been provided in the manuscript, while the components of the ConvLayer are presented in Table 11. $M_i$ and $N_i$ represent the control point grid dimensions in local refinement stage $i$.

| **Module** | **Layer** | **Input channels** | **Output channels** |
|---|---|---|---|
| Homography Estimator | ConvLayer | 2 | 64 |
| | ConvLayer | 64 | 128 |
| | ConvLayer | 128 | 256 |
| | Group Linear Layer | 4096 | 2048 |
| | Group Linear Layer | 2048 | 1024 |
| | Linear | 1024 | 8 |
| FFD Estimator-1 | ConvLayer | 81 | 64 |
| | ConvLayer | 64 | 128 |
| | ConvLayer | 128 | 256 |
| | ConvLayer | 256 | 512 |
| | Group Linear Layer | 8192 | 4096 |
| | Group Linear Layer | 4096 | 2048 |
| | Linear | 2048 | $(M_1 + 1) \times (N_1 + 1) \times 2$ |
| FFD Estimator-2 | ConvLayer | 81 | 32 |
| | ConvLayer | 32 | 64 |
| | ConvLayer | 64 | 128 |
| | ConvLayer | 128 | 256 |
| | ConvLayer | 256 | 512 |
| | Group Linear Layer | 8192 | 4096 |
| | Group Linear Layer | 4096 | 2048 |
| | Linear | 2048 | $(M_2 + 1) \times (N_2 + 1) \times 2$ |

Table 11. Components of ConvLayer(Input channels, Output channels). Note: For all Conv2d layers, kernel sizes are $3 \times 3$, strides are 1, and paddings are 1. The max-pooling layer has a kernel size of $2 \times 2$ and stride of 2.

| **Layer** |
|---|
| Conv2d(Input channels, Output channels) |
| ReLU |
| Conv2d(Output channels, Output channels) |
| ReLU |
| MaxPool2d |


Figure 6. Qualitative results on the UDIS-D dataset [31]. The yellow arrows in the image highlight regions where other methods fail to achieve accurate alignment.

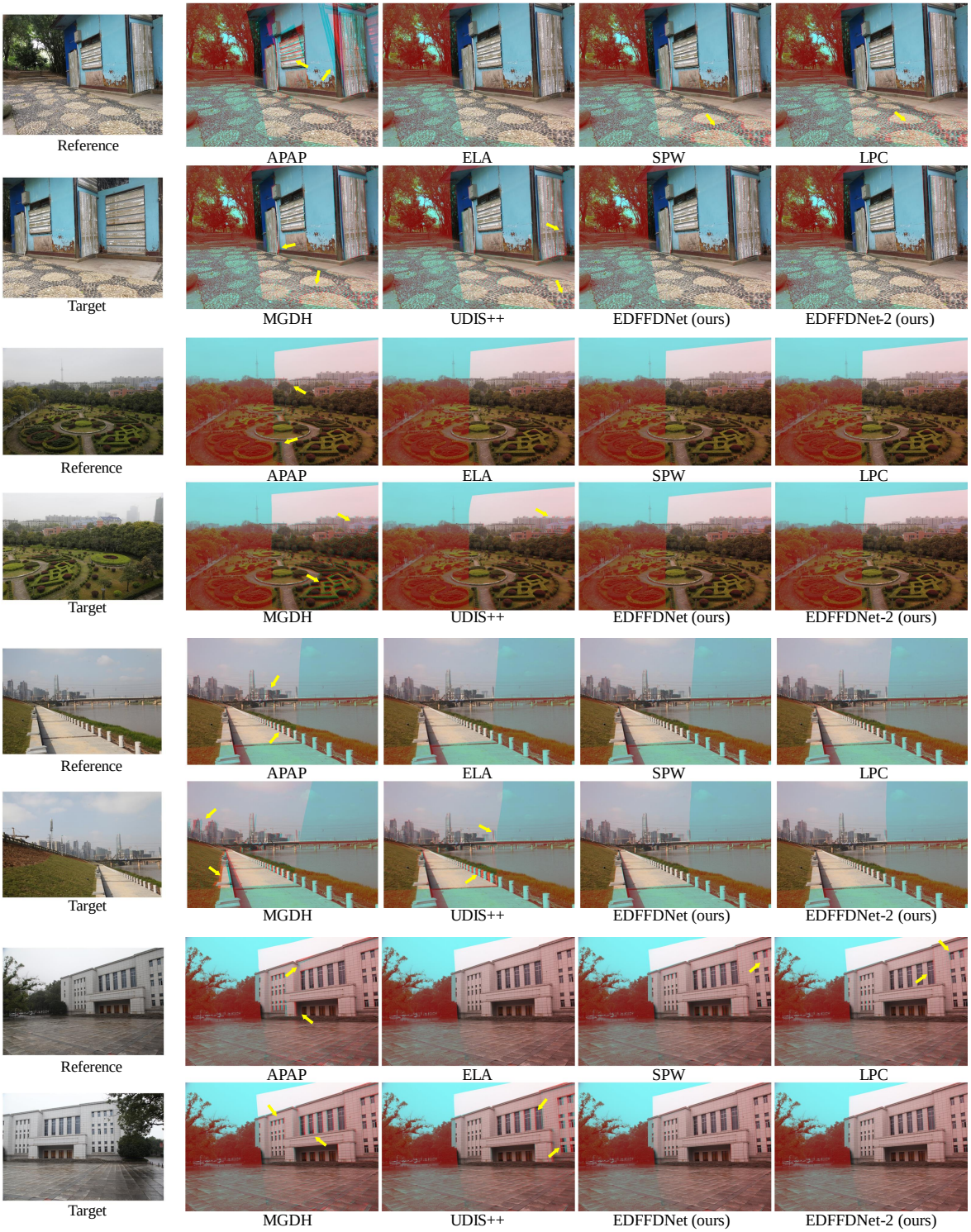


Figure 7. Qualitative results on cross-dataset cases [24]. The arrows in the image highlight regions where other methods fail to achieve accurate alignment.

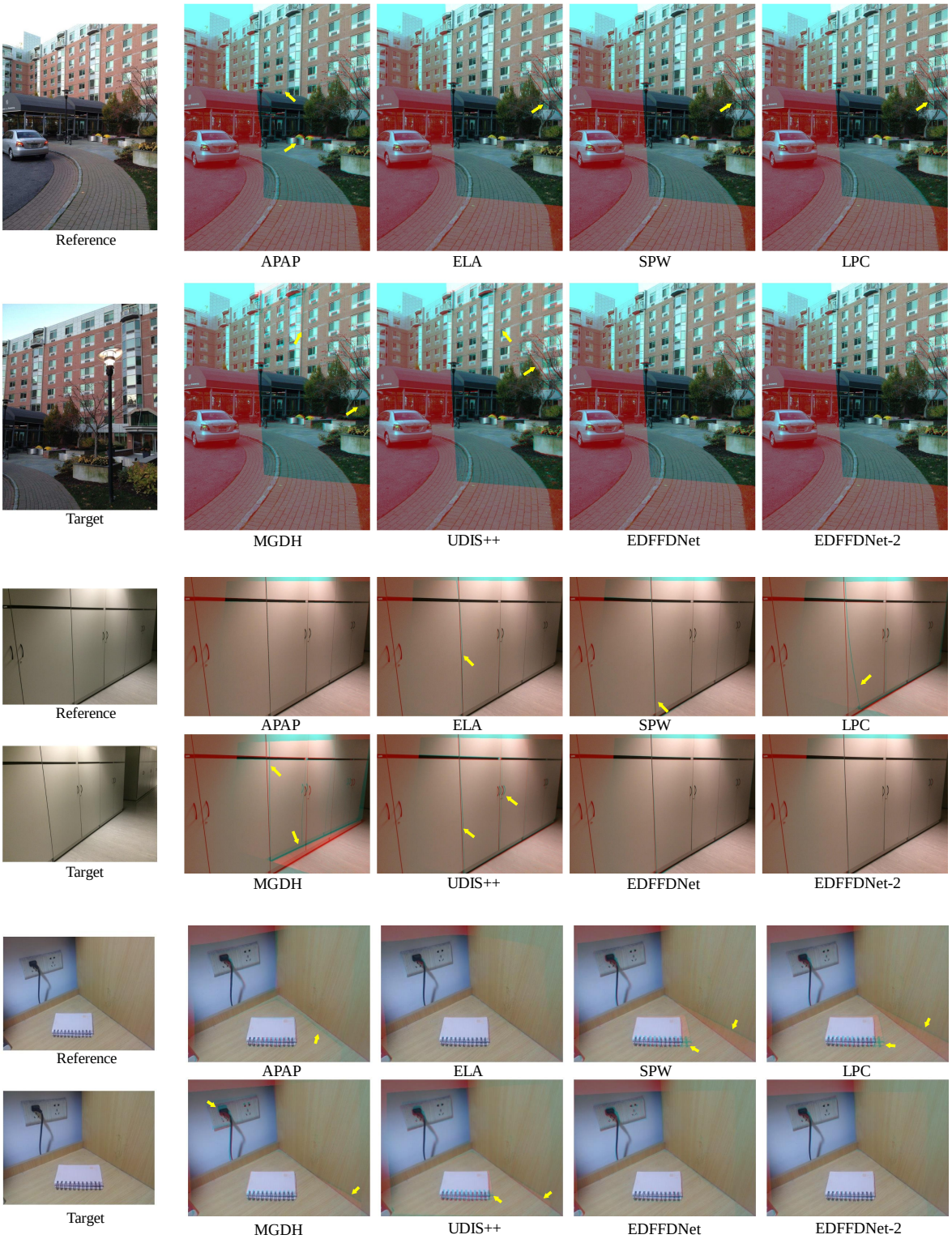


Figure 8. Qualitative results on cross-dataset cases [24]. The arrows in the image highlight regions where other methods fail to achieve accurate alignment.

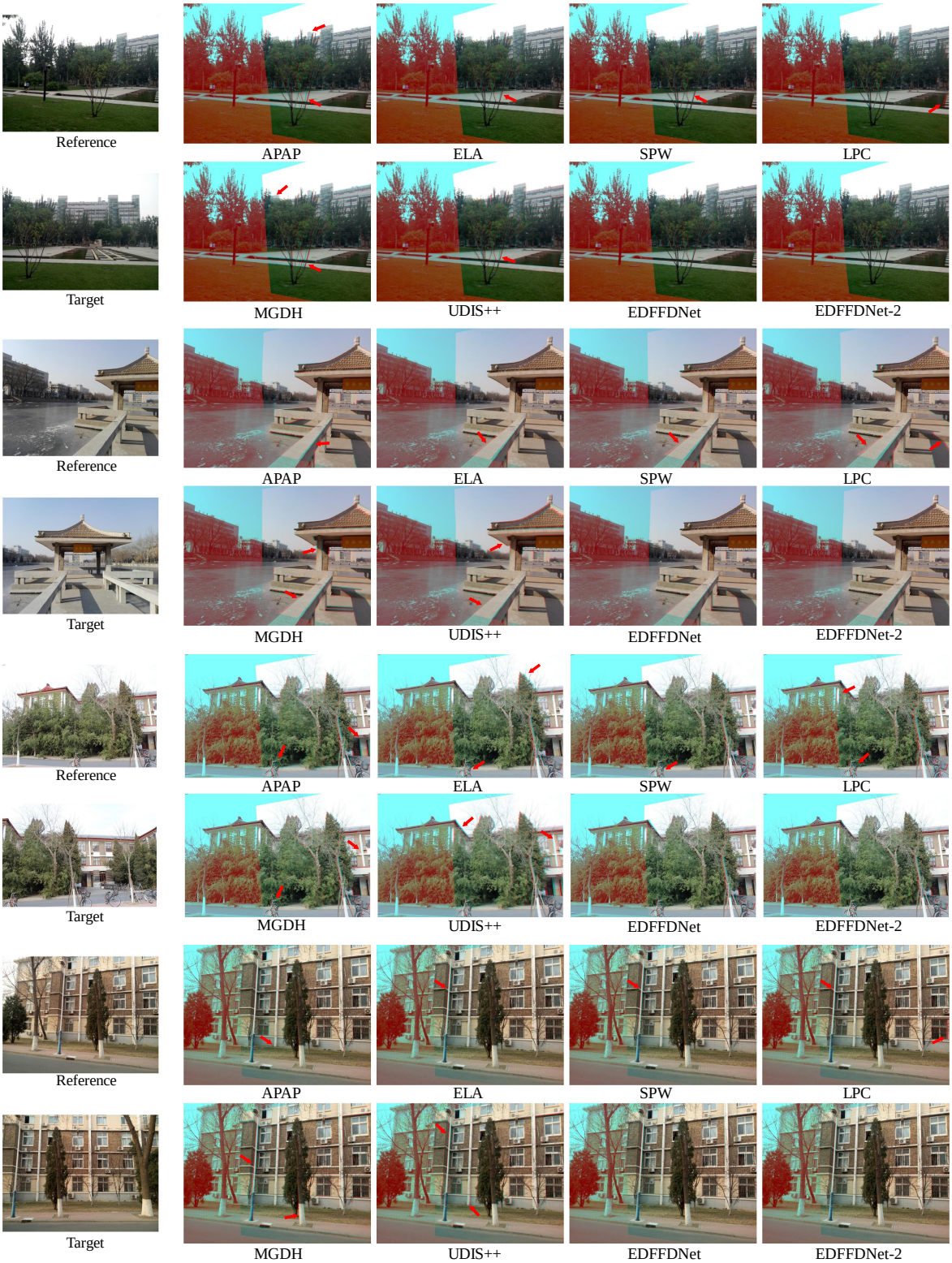


Figure 9. Qualitative results on cross-dataset cases in [25]. The arrows in the image highlight regions where other methods fail to achieve accurate alignment.

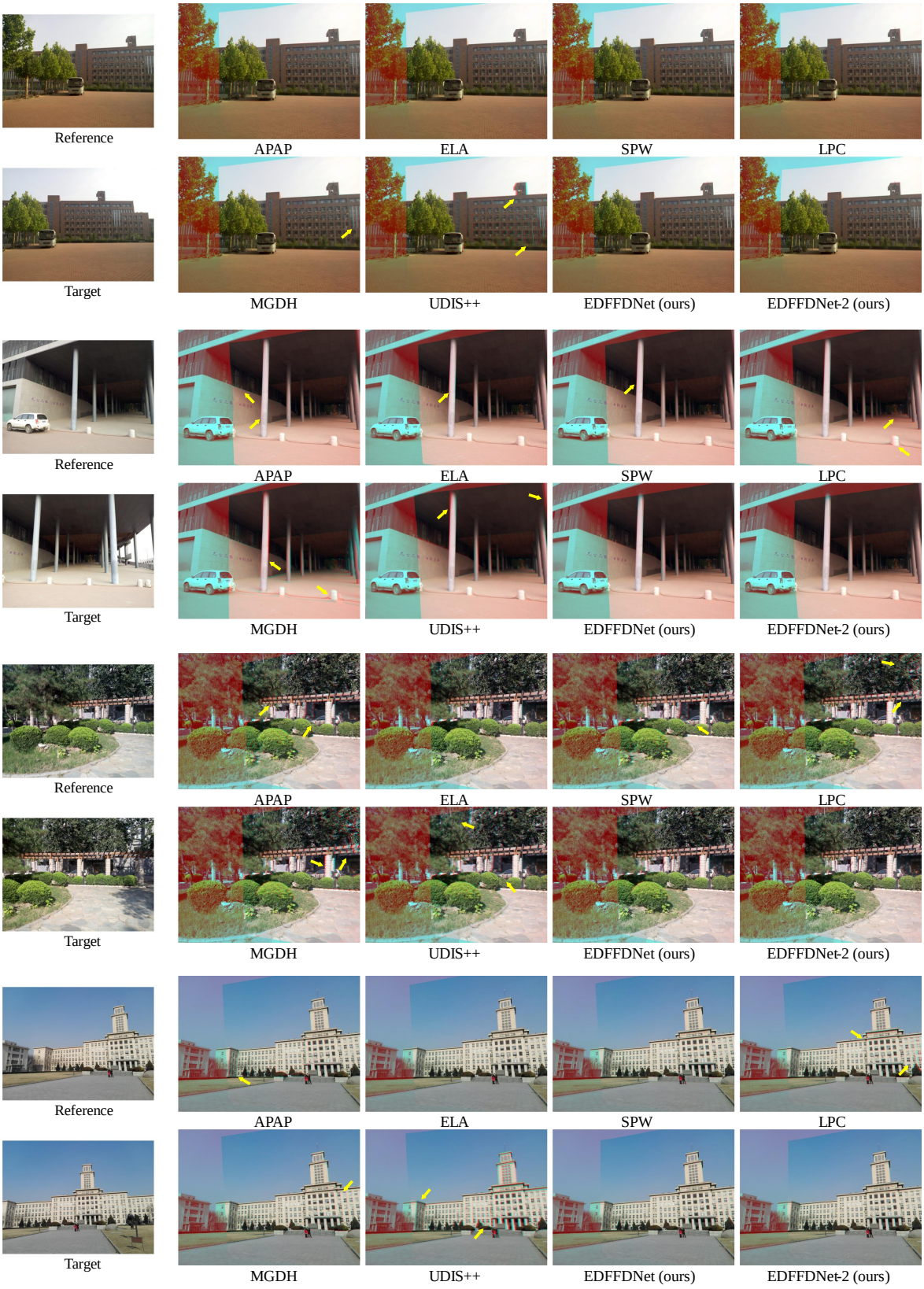


Figure 10. Qualitative results on cross-dataset cases in [25]. The arrows in the image highlight regions where other methods fail to achieve accurate alignment.