ReFineR: Single image to dense point cloud registration at pixel level with point splatting

Ugo Insalaco[1,2]
ugo.insalaco@minesparis.psl.eu
Jean-Emmanuel Deschaud[1]
jean-emmanuel.deschaud@minesparis.psl.eu
Pierre Dellenbach[2]
pdellenbach@chapsvision.com

[1] Mines Paris, PSL University
Centre for robotics (CAOR)
75006 Paris, France
[2] ChapsVision
4 Rue du Port aux Vins,
92150 Suresnes, France

Abstract

Cameras and LiDARs (Light Detection and Ranging) provide complementary observations of the same scene: cameras capture rich appearance cues while LiDAR sensors provide accurate geometric measurements. In this work, we address the problem of fine image-to-point cloud registration from a single image and a roughly estimated camera pose. On robotic platforms, residual reprojection errors of several pixels are commonly observed even after calibration due to temporal desynchronization, mechanical vibrations, or inaccuracies in point cloud reconstruction. Such errors can lead to decimeter-level geometric discrepancies at moderate distances and significantly affect multiple downstream tasks, such as dense cross-modal supervision, texture mapping, and semantic label transfer. To address this challenge, we introduce ReFineR: an offline framework to finely register individual images to pre-reconstructed point clouds. ReFineR builds upon differentiable point splatting rasterization and a smooth Gradient Orientation Measure (GOM) to iteratively refine the camera pose through analysis-by-synthesis optimization. We evaluated the method on a curated dataset of manually calibrated images collected from heterogeneous data sources. Furthermore, we show that an automatic point cloud rescale factor can be estimated prior to optimization, enabling robust refinement across a wide range of scenarios using a single scale-invariant formulation. The project page can be found at https://refiner-project.github.io/

1 Introduction

Accurate camera-LiDAR alignment is often treated as a solved pre-processing step, assuming a fixed extrinsic calibration between sensors across an entire sequence. However, in practice, even well-calibrated multi-sensor platforms exhibit residual reprojection errors that vary from frame to frame due to temporal misalignment, mechanical vibrations, rolling-shutter effects, and SLAM drift. In large-scale autonomous driving datasets such as nuScenes [6], reprojection discrepancies of 7–10 pixels are commonly observed despite official calibration parameters (details in Supplementary Section B.8). Although such errors may appear negligible, they can lead to significant metric misalignments: a 10-pixel error can induce

© 2026. The copyright of this document resides with its authors.
It may be distributed unchanged freely in print or electronic forms.

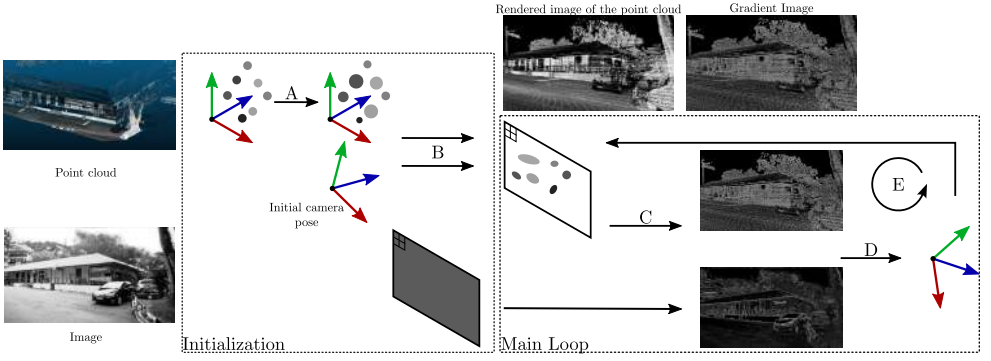


Figure 1: ReFineR optimization loop. A. During initialization, points are spatially extended to prepare the differentiable rendering step. B. Differentiable point splatting creates a view of the point cloud at the initial camera position (Section 4.2). C. Gradients are computed using a Sobel filter. D. Both modalities are compared using the GOM loss and selected comparison channels (Section 4.3 and Section 4.4). E. The camera pose is optimized by backpropagation through the differentiable image, and the new pose iteratively renders new views (Section 4.5).

up to 40 cm of lateral misalignment at moderate distances. This magnitude critically affects downstream applications like dense cross-modal supervision, semantic label transfer from image to LiDAR, HD map construction, and fine-grained scene understanding, where centimeter-level consistency is required. Importantly, these errors are not constant over time and cannot be corrected by a single global calibration step, making per-image fine alignment essential. Therefore, beyond coarse calibration, sub-pixel single-camera-LiDAR registration is a fundamental problem for high-precision multi-sensor perception.

Advances in performant SLAM-LiDAR algorithms [12] allow the reconstruction of dense and precise point clouds from the aggregation of multiple LiDAR scans. When used for rasterizing new images, these dense point clouds create high-quality images containing rich visual information that can be leveraged for image-to-point cloud registration. Previous work [9, 10, 19, 30, 52, 59] has shown that analysis-by-synthesis – the principle of iteratively rendering the point cloud in artificial images and comparing each render with the real image in a loop – performs significantly well for the task of fine-level registration. While earlier methods used non-differentiable algorithms for optimization [19, 52], our work takes advantage of the recent development of differentiable rendering techniques [28, 32, 56, 61] to propose a fully differentiable pipeline, from the camera pose parameters to the loss value.

This paper presents a threefold contribution. **1)** As pixel-level registration is in fact absent from many LiDAR-image datasets [6, 7, 51], we propose an open dataset, created from six heterogeneous sources of point cloud and image data (ETH3D [50], CARLA [14], nuScenes [6], Tanks and Temples [29], RGBDScenes [31], and data from a Livox sensor [62]) with manually calibrated images on reconstructed dense point clouds for three of these sets. **2)** We propose an automatic rescaling method, derived from first-order pinhole camera model analysis, to provide a principled solution to current scene-dependent hyperparameter tuning. **3)** We introduce ReFineR for **Re**gistration at **Fine** level using differentiable **R**endering, between single images and any pre-reconstructed point clouds. ReFineR is a training-free method that handles heterogeneous point channels at once. Prioritizing reg-

istration accuracy over speed, ReFineR was created for offline processing (HD mapping, semantic transfer...) rather than being a localization brick in real-time systems.

2 Related work

Knowing correspondences between 3D points and 2D pixels, closed-form solutions exist to estimate the camera's 6 degrees of freedom [33]. For more robustness to matching noise, most common approaches use either the EPnP algorithm [33] combined with the RANSAC [15] scheme, or a residual minimization of the reprojection error using the Levenberg-Marquardt optimizer [30, 34, 40]. For the relaxed problem of camera alignment without known matches, a recent survey from An *et al.* [1] classifies automatic registration methods as explicit or implicit from their estimation of point-pixel pairs or not.

2.1 Explicit registration

In order to remove the manual task of selecting matches, some work has been focusing on automatically estimating such correspondences, either via feature matching algorithms in 2D, such as SuperGlue [30, 49] or MatchAnything [20], or with direct estimation of 2D-3D matches [25, 35, 44, 48, 54, 55]. CoFiI2P [25] trains a transformer model to generate matches between image and point features at different scales. CorrI2P [48] adds global descriptors on both modalities, given to an overlapping region detector network that produces frustum-consistent matches. I2PNet [54] estimates soft matches between points from the LiDAR scan and pixels, themselves fed to a pose regression network. Diff2I2P [44] uses a deformable correspondence tuning layer to estimate matching offsets and a diffusion loss to compare RGB images and depth maps. UniCorrn [18] embeds 2D and 3D features in a common space to predict correspondences between any modality pair, regrouping 2D-2D, 2D-3D, and 3D-3D matching into a single algorithm. A key limitation of most 2D-3D matching algorithms [18, 25, 35, 44, 48, 54, 55] is their generalization ability to various point clouds. Since the models were generally trained on individual frames (LiDAR or RGBD), they tend to show weaker results given denser point clouds as input (for instance, acquired from a dense static LiDAR scanner or aggregated scans from a mobile LiDAR). Other approaches try to directly predict pointmaps (a Width $\times$ Height $\times$ 3 tensor representing the position in space of each pixel): Brachmann *et al.* [4, 5, 58] trained a deep neural network with a differentiable version of RANSAC [4] for this task. SANet [58] estimated a pointmap from deep matching between a query image and the point cloud and reference views' features. Similarly, both these methods are typically trained on specific datasets or scenes, which limits their ability to generalize across different environments or scenes. In contrast, our approach does not require any training and can be applied directly to a wide variety of image–LiDAR datasets without parameter tuning.

2.2 Implicit registration

A parallel line of work focuses on synthesizing views from the point cloud using coarse initialization, then comparing the rendered and real images to refine the position and orientation, with the goal of overlapping both images (*a.k.a.* analysis-by-synthesis). Iterations can either be carried out using neural network comparison of images [24, 39] – CalibNet [24] concatenates image and depth map features before feeding them to a fully con-

nected neural network, LCCNet [39] introduced a cost volume layer instead, and Ou *et al*. [46] added a trained diffusion model as a denoiser to the pose predicted by LCCNet – or optimization of the pose's parameters from handcrafted loss functions between images [8, 19, 30, 37, 47, 52, 62]. Early work from Taylor *et al*. [52] compared a rasterized view of the point cloud to the image using Gradient Orientation Measure which minimizes the angle between each pixel's gradient. This loss was then optimized using the non-differentiable Nelder-Mead optimizer [45]. Koide *et al*.'s method [30] minimized the normalized information distance, a comparison between the distributions of RGB values and synthesized LiDAR reflectivity to optimize the pose. Guislain *et al*. [19] refined the camera position using a mix of entropic and gradient information thanks to Mutual Information and Histogram of Oriented Gradients (HOG) features [11]. Yuan *et al*. [62] first extracted depth continuous edges from the point cloud and image, and minimized a line reprojection error over the Lie algebra of rigid transforms. Our approach (ReFineR) falls into the category of implicit registration.

2.3 Differentiable rendering

Differentiable rendering creates images whose pixel values are differentiable with respect to the 3D scene's parameters (point positions, camera extrinsics and intrinsics, etc). These methods arose jointly with smooth 3D representation and reconstruction engines like NeRF [43] or Gaussian Splatting [28], and involve mechanisms like volumetric rendering [3, 43] or Gaussian/surfel splatting [28, 32, 56, 61]. For instance, DSS [61] projects isotropic flat Gaussians centered on each point onto the image plane, and averages for each pixel the contribution of each Gaussian. To account for the discontinuous point occlusion, they computed artificial gradients based on the relative movement between the point and the deprojected pixel. 3D Gaussian Splatting [28] took a completely differentiable path by rasterizing Gaussians onto the image plane and blending them using an opacity value.

2.4 Gaussian Splatting registration

Setting aside image-to-point-cloud registration, some studies have been carried out on registering new images on pre-reconstructed Gaussian Splatting models [10, 36, 41, 42, 57] using either feature matching on rendered views [10, 36], reprojection error minimization from rendered depth maps [57], or reverse rendering to find rays intersecting the image plane [42]. Matsuki *et al*. [41] proposed to directly integrate Lie gradients over the camera pose in the rendering process to achieve end-to-end differentiability from camera to pixel values. These methods do not directly apply to our problem as they require preliminary optimization of a Gaussian Splatting model from multiple images of the same scene (which in our single-image case is not available) and are often based on photometric error minimization [10, 41], which cannot be achieved with geometry- or reflectivity-valued LiDAR point clouds. They, however, prove that differentiable rendering can be a valuable asset for estimating fine camera positions.

3 Preliminaries

Reprojection error and parallax: Pixel error p_e (a.k.a. reprojection error) can be defined as the root-mean-square value of the displacement of pixels between two projection models π_1 and π_2, given a point cloud $(X_i)_{1\leq i\leq N} \in \left(\mathbb{R}^3\right)^N$:

$$p_e = \sqrt{\frac{1}{N}\sum_{i=1}^{N} \|\pi_1(X_i) - \pi_2(X_i)\|^2} \tag{1}$$

In our problem, pixel error is interesting because it defines the registration difficulty independently of the latent point cloud scale (which would not be the case when comparing distances using the translation error). For a small perturbation $\delta\tau \in \mathfrak{se}(3)$ of the camera extrinsics around 0, we obtain:

$$\pi(X_i) - \pi_0(X_i) = J(X_i)\delta\tau + o(\|\delta\tau\|) \tag{2}$$

with $J = \frac{\mathcal{D}\pi}{\mathcal{D}\tau}$ (borrowing notation from [13]) computed explicitly in supplementary. In particular, the terms in J associated with rotation only depend on point projections $(x_c, y_c) = (f_x X_{iX}/X_{iZ}, f_y X_{iY}/X_{iZ})$ on the image, with f_x, f_y the camera intrinsics. In contrast, the ones for translation also depend on their depth X_{iZ}: the farther a point is, the less impact a translation will have on its reprojection error. This is the well-known phenomenon of *parallax*.

4 Method

We assume that a coarse initialization of the camera pose is available prior to optimization. Such an initialization may originate from a standard camera-LiDAR calibration performed on the robotic platform, combined with LiDAR-Odometry [6] or from Structure-from-Motion point clouds registered on a LiDAR point cloud [29]. Importantly, estimating this coarse pose differs fundamentally from the fine alignment addressed in this work, as it is typically solved through geometric calibration or feature matching rather than analysis-by-synthesis optimization.

The ReFineR algorithm is structured into four main stages (see Fig. 1). **1)** The input point cloud scale is normalized following Section 4.1, allowing our method to be scale independent. **2)** A rendered view of the point cloud is created with differentiable point splatting (based on Gaussian Splatting's [28] framework), as described in Section 4.2. **3)** Handcrafted features, based on gradient orientation, are computed on the synthetic and real images and compared according to the rendered channels (Section 4.3 and Section 4.4). **4)** Backpropagation is carried out with computation of Lie gradients with respect to the camera parameter τ (Section 4.5), and the process continues from step 2. This pipeline is entirely differentiable.

4.1 Point cloud rescale estimation

When handling heterogeneous point clouds (*e.g.*, indoor, outdoor, or RGB-D data) and accounting for parallax effects, optimizing the camera pose using gradient-based algorithms requires prior knowledge of the point cloud's scale in order to appropriately adjust the learning rate. In this section, we propose an expression for automatically estimating such a prior, allowing fixed learning rates for rotation and translation across all scenes.

Following Eq. (2), we can approximate the pixel error using:

$$p_e^2 = \delta\tau^T \underbrace{\left(\frac{1}{N}\sum_{i=1}^{N} J(X_i)^T J(X_i)\right)}_{S} \delta\tau + o(\|\delta\tau\|^2) \tag{3}$$

Intuitively, the symmetric positive matrix S integrates the contributions and correlations to the pixel error of each transform parameter in τ. Separating contributions from rotation and translation, and introducing the unknown scale coefficient α such that $X_i = \alpha X'_i$, this matrix can be rewritten as (details in Supplementary Section A.3):

$$S = \begin{pmatrix} S_r & S_{rt} \\ S_{rt}^T & S_t \end{pmatrix} = \begin{pmatrix} S_r & \frac{1}{\alpha} S'_{rt} \\ \frac{1}{\alpha} S'^T_{rt} & \frac{1}{\alpha^2} S'_t \end{pmatrix} \tag{4}$$

To select an appropriate α, our criterion is to balance the effects of translation and rotation on the pixel error. Concretely, we would expect a translation of 1 (written without a unit on purpose), to have approximately the same effect on the pixel error as a rotation of 1. To do so, we choose to equalize the Hilbert-Schmidt norms between S_r and S'_t. We obtain:

$$\alpha = \left(\sum_{i=1}^{3} \lambda_{r,i}^2 / \sum_{i=1}^{3} \lambda_{t,i}^2 \right)^{1/4} \tag{5}$$

with $(\lambda_{r,i}, \lambda_{t,i})_i$ the respective eigenvalues of S_r and S_t. An example of rescaled scenes is shown in Supplementary Fig. 2. Generally speaking, this rescaling procedure is a form of reparameterization to obtain better variable conditioning, which is a major issue in numerical optimization [53].

4.2 Synthesis

After rescaling the point cloud, we initialize a Splatting model by creating an isotropic Gaussian for each 3D point (with standard deviation proportional to the distance to each point's nearest neighbor by a factor p_s), and setting a fixed opacity o. To limit the visual impact of isolated points (which would therefore be initialized with a large size), we limit the size of each Gaussian such that its projection does not exceed 4 pixels in diameter. We add each point channel value as a Gaussian feature (normals calculated from the point cloud, LiDAR reflectivity, etc.).

4.3 Analysis

Edge-based cues have been explored in prior work for cross-modal alignment [19, 52]. Building on the observation that image gradients correlate with both point cloud geometry and object texture, we follow Taylor *et al.*'s work [52] and adapt the Gradient Orientation Measure (GOM). From two images $(I_p)_p$, $(I'_p)_p$ indexed by their pixels and their corresponding gradients $(G_p)_p$ and $(G'_p)_p$, computed using (differentiable) Sobel filters, the GOM is defined as

$$GOM(I, I') = \frac{\sum_p |\left(G_p | G'_p\right)|}{\sum_p \|G_p\|_2 \|G'_p\|_2} \tag{6}$$

GOM aims at maximizing the dot product between gradients, therefore aligning edges from two images.

4.4 Comparison

Comparison between modalities is achieved by creating correspondences between point clouds and image channels, and computing the GOM loss function for each correspondence.

For the point cloud, the different usable channels are: average normal direction (p^n), individual normal coordinates (p^{nx}, p^{ny}, p^{nz}), normalized projected depth (p^d), LiDAR reflectivity (p^r), or grayscale value per point (p^g) acquired from an embedded camera in the LiDAR sensor (for example the fixed terrestrial FARO scanner). For images, we compute their grayscale values (i^g) from the RGB channels. In our ablation study, we experiment with adding generated channels for images using the Marigold diffusion model [27], adding the artificial normal channels i^{nx}, i^{ny}, i^{nz} and normalized depth i^d (see Section C.4).

The final multimodal loss function between the multichannel rendered image $(I^c)_{c\in\mathcal{C}_{3D}}$ and real image $(I'^c)_{c\in\mathcal{C}_{2D}}$ is then computed as:

$$\mathcal{L}(I,I') = -\frac{1}{|\mathcal{C}_{\leftrightarrow}|}\sum_{(c,c')\in\mathcal{C}_{\leftrightarrow}} GOM\left(I^c, I'^{c'}\right) \tag{7}$$

With $\mathcal{C}_{3D}$ and $\mathcal{C}_{2D}$ denoting the sets of available point cloud and image channels and $\mathcal{C}_{\leftrightarrow} \subset \mathcal{C}_{3D} \times \mathcal{C}_{2D}$ a set of correspondences between 3D and 2D channels. In our experiments, we consider two common setups: the first, where the only available information is the point cloud geometry, and the second, with LiDAR reflectivity or grayscale value. For the first scenario, we therefore use ReFineR with $\mathcal{C}_{\leftrightarrow} = \{(p^{nx}, i^g), (p^{ny}, i^g), (p^{nz}, i^g)\}$ and for the second, we use $\mathcal{C}_{\leftrightarrow} = \{(p^r, i^g)\}$ or $\mathcal{C}_{\leftrightarrow} = \{(p^g, i^g)\}$.

4.5 Optimization

Our pipeline is entirely differentiable from the camera parameter $\tau \in \mathfrak{se}(3)$ to the loss value $\mathcal{L}(I,I')$ and uses Gaussian Splatting SLAM's [41] codebase as the differentiable point splatting rasterizer. Optimization is carried out using the Lie version of gradient descent, with a fixed learning rate. Other optimization algorithms could be possible, as previous work experimented with integrating the Levenberg-Marquardt optimizer [23, 34] through the Gaussian Splatting rasterizer, but their method additionally requires a large amount of GPU memory (∼50GB), which would reduce the applicability of our method. The number of iterations is bounded between 50 and 1000, and a minimum step magnitude s_m is fixed to dynamically stop optimization upon convergence.

5 Experiments

5.1 Setup

Data: For the task of image-to-point cloud registration with a pixel-level goal, there is, to our knowledge, no unified evaluation framework. Generally, state-of-the-art methods either choose to run their experiments on their own data [19, 30, 62], only on RGB-D images [35, 55] or on odometry-based calibration like KITTI Odometry or nuScenes [16, 25, 39, 48] which do not have pixel-precision calibration (see Table 6 from Supplementary Section B.8) with registration on individual frames rather than the fully reconstructed point clouds. As part of our open contributions, we curated a dataset of point clouds and finely registered color images from six different sources: ETH3D [50], Tanks and Temples [29], nuScenes [6], Livox [62], RGBDScenes [51], and synthetic data from the CARLA simulator [14]. ETH3D is a multiview stereo benchmark with high-resolution images and high-quality point clouds acquired with a FARO scanner, used for evaluation. The ETH3D dataset

provides subpixel camera registration, thanks to photometric error minimization across multiple views at once. This ideal scenario (RGB point cloud, high-precision scanner, multiple views) does not, however, reflect the full diversity of point clouds, and we therefore choose to enrich it from other sources. Tanks and Temples is closely related to ETH3D, as each point cloud was acquired using a similar FARO scanner, but we show (see Supplementary Section B) that this dataset has coarser image registration of the order of 5 to 10 pixels on average. Indeed, the ground truth camera poses were obtained through manual registration of an SfM model to the point cloud, and errors originating from the sparse reconstruction directly impact the final camera poses. To improve on this error, we estimated the camera parameters for 20 images of this dataset by selecting points and pixel matches and minimizing the reprojection error with Gauss-Newton optimization. We obtained reprojection errors of 2px on average after this refinement. More information on our manual registration procedure is given in Section B. nuScenes data are outdoor sequences obtained from a Velodyne HDL32E LiDAR mounted on a car, alongside cameras, radars, and an IMU. We obtained dense point clouds by running CT-ICP [12], a LiDAR odometry method that provides a precise pose for each LiDAR scan. After reconstruction and using the official sensor platform calibration, we still noticed residual errors and chose to manually register 20 images from this dataset, improving the registration from 7–10px to 3–5px. Livox data presented in Yuan *et al.*'s paper [62] consist of four scans obtained from a Livox sensor and a single image per scan. Each image was similarly manually registered. RGBDScenes consists of color and depth images recorded from a Kinect sensor, where point clouds are created by aligning frames using Patch Volumes Mapping [21]. CARLA is an autonomous driving simulator that benefits from perfect registration, while still producing realistic point clouds and images. *For the rest of this paper, except for the RGBDScenes dataset, we will consider images with a resized height of* 1024*px. For RGBDScenes, height is kept at* 480*px. Every pixel error will be expressed accordingly.* In total, this dataset is composed of 50 images from ETH3D, 50 images from CARLA, 20 images from Tanks and Temples, 20 images from nuScenes, 20 images from RGBDScenes, and 4 images from Livox, for a total of 164 finely registered images across 29 heterogeneous point cloud scenes.

Methods: ReFineR is evaluated against Koide *et al.*'s method [30] and the intensity-free method from Yuan *et al.* [62]. It is known that synthesized views of a point cloud can be used to find meaningful correspondences with real images in 2D [22, 60]. We thus compare ReFineR with the MatchAnything model [20], which extracts 2D matches between heterogeneous modality images. After matching in 2D using MatchAnything, we retrieve the 3D coordinates of non-empty pixels and estimate the pose using PoseLib's RANSAC. MatchAnything's default model configuration was kept. We added GS-SLAM, which uses photometric loss for comparison with the base work from Matsuki *et al.* [41]. The first two methods (Koide *et al.* [30] and Yuan *et al.* [62]) require hyperparameter tuning based on the scene type and scale, which we detail in Supplementary Section C.1. Conversely, ReFineR is a scale-invariant approach, for which we fix $s_m = 2 \times 10^{-5}$, generally stopping our registration loop after 100 to 200 iterations. Gradient descent is handled by the Adam optimizer in PyTorch with learning rate $l_r = 2 \times 10^{-4}$. The splatting model is initialized using $o = 0.2$ and $p_s = 0.5$. All our experiments ran on a GeForce RTX 5090 graphics card with 32GB VRAM, but for most sets, 8GB VRAM would suffice. Indeed, ReFineR's computational cost is directly related to the number of points to be registered to, which in general doesn't exceed 10M (including frustum culling). We experimented with pretrained deep learning-based approaches [18, 24, 55, 63], but noticed that they tend to predict abnormal 2D-3D

correspondences for our task of fine-level camera calibration to dense point clouds. These methods were pretrained on individual nuScenes scans or RGBD depth maps, which we suspect introduced a distribution gap with inference on dense depth maps. A more detailed reflection is shared in Section C.9.

Randomized Pose Initialization: Since point clouds are acquired at different scales and to keep consistency with our scale-invariant approach, we perturb our camera poses such that the resulting *average* pixel error has a fixed value. This can be done using a first-order approximation of the pixel error (see Eq. (2)) and selecting a random transform on the ellipsoid of constant error. Computation details are given in Supplementary Section A.4. For the sets that don't have manual matches between points and pixels (e.g., ETH3D, CARLA, RGBDScenes), we sample 100 random points from each image's depth map. This initialization technique has the advantage of avoiding a fixed translation perturbation, which might have a varying effect on the reprojection error due to parallax. We chose an initial perturbation of 10 px as it was, on average, our observed residual reprojection errors for the nuScenes and Tanks and Temples datasets. In practice, an error of about 10 pixels can induce translation discrepancies of several tens of centimeters in outdoor scenes (*e.g.* CARLA). Each camera pose was perturbed from random starting positions twice for the CARLA and ETH3D sets, and 5 times for the others, adding up to a common 100 registrations per set (except Livox, which has 20).

Metrics: We report results in terms of percentage of registrations below a given threshold t_h in pixels, denoted pix@t_h with pix@$t_h = \frac{1}{N}\sum_i \mathbb{1}(p_i < t_h)$, N the number of evaluated pairs, and p_i the resulting pixel error of sample i. We also give median pixel errors across all registered scenes per set. Because translation error is scale-dependent, we only report it along with rotation error in supplementary Section C.2. Both metrics are respectively defined for two rigid transform matrices $T = \begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix}$ and $T' = \begin{pmatrix} R' & t' \\ 0 & 1 \end{pmatrix}$ as $\text{transl} = \|t' - t\|$ and $\text{rot} = \arccos((\text{trace}(R'^T R) - 1)/2)$.

5.2 Registration with raw point cloud

In this first setup, we assume that no other information than the point cloud geometry is available. Indeed, this is a rather common setup, as for low-cost LiDARs, the point intensity is strongly correlated with the laser range or incidence angle, and not the underlying material, which makes this channel unusable when fusing multiple scans in a dense point cloud [26]. From the raw point cloud, we compute normal directions using 50-nearest-neighbor PCA. Since the benchmarking methods [20, 30, 41, 52] only take as input mono-intensity images, we synthesize the mean normal value p^n and compare it with the image grayscale intensity i^g. For our method, which is capable of handling multiple channels at once, we use each normal's coordinate individually, compared with the image's grayscale value i_g: $\mathcal{C}_{\leftrightarrow} = \{(p^{nx}, i^g), (p^{ny}, i^g), (p^{nz}, i^g)\}$.

Results from Table 1 (one stage) show that for the more geometrically accurate datasets, ReFineR greatly improves on MatchAnything's baseline (41% subpixel registration on ETH3D versus 8%, 44% below 2px on CARLA against 14% for MatchAnything) but lacks robustness for noisier datasets (Livox and RGBDScenes) and underperforms in the 5–10px ranges in which MatchAnything excels. It is unsurprising that this experimental setup is a failing case for GS-SLAM and Koide *et al.* as both these methods directly compare point intensities

	ETH3D				Tanks and Temples				CARLA				Livox			nuScenes				RGBDScenes			
	pix@ (%↑)			p_ϵ(px↓)	pix@ (%↑)			p_ϵ(px↓)	pix@ (%↑)			p_ϵ(px↓)	pix@ (%↑)		p_ϵ(px↓)	pix@ (%↑)			p_ϵ(px↓)	pix@ (%↑)			p_ϵ(px↓)
	1	2	5		2	3	5		1	2	5		5	10		3	5	10		3	5	10	
One stage																							
GS-SLAM [📷]	0	0	0	123.23$_{\pm 64.59}$	0	0	0	264.12$_{\pm 181.23}$	0	0	0	371.52$_{\pm 214.75}$	0	0	113.85$_{\pm 34.13}$	0	0	0	309.77$_{\pm 110.45}$	0	0	0	90.32$_{\pm 32.97}$
Yuan *et al.* [📷]	0	2	20	10.29$_{\pm 9.74}$	1	17	52	4.39$_{\pm 7.71}$	0	2	4	15.20$_{\pm 41.70}$	25	100	5.83$_{\pm 1.21}$	0	0	5	21.10$_{\pm 37.53}$	0	0	5	26.95$_{\pm 26.01}$
Koide *et al.* [📷]	0	3	25	11.10$_{\pm 10.86}$	0	0	4	29.00$_{\pm 67.16}$	2	8	28	8.82$_{\pm 16.73}$	25	75	6.61$_{\pm 2.60}$	0	0	40	12.03$_{\pm 12.21}$	0	0	5	36.19$_{\pm 28.14}$
Taylor *et al.* [📷]	0	1	27	7.13$_{\pm 3.00}$	2	5	18	7.79$_{\pm 2.47}$	0	0	10	8.87$_{\pm 3.75}$	0	75	8.35$_{\pm 1.09}$	0	0	25	14.61$_{\pm 26.77}$	0	10	25	12.77$_{\pm 7.73}$
MatchAnything [📷]	8	65	100	1.80$_{\pm 0.83}$	1	46	95	3.05$_{\pm 0.94}$	0	14	65	4.12$_{\pm 3.88}$	65	100	4.44$_{\pm 0.86}$	0	28	88	5.71$_{\pm 2.56}$	0	28	95	6.07$_{\pm 1.90}$
ReFineR	41	73	79	1.10$_{\pm 3.17}$	64	88	94	1.86$_{\pm 1.87}$	12	44	74	2.36$_{\pm 3.45}$	30	90	5.50$_{\pm 2.13}$	9	41	87	6.29$_{\pm 3.08}$	1	8	72	8.96$_{\pm 2.08}$
Two stages																							
MatchAnything + GS-SLAM [📷]	0	0	0	122.55$_{\pm 64.26}$	0	0	0	219.32$_{\pm 204.55}$	0	0	0	356.62$_{\pm 492.77}$	0	0	117.12$_{\pm 33.88}$	0	0	0	320.79$_{\pm 110.81}$	0	0	0	86.55$_{\pm 43.01}$
MatchAnything + Yuan *et al.* [📷]	0	1	30	8.99$_{\pm 46.80}$	0	11	55	4.56$_{\pm 7.63}$	0	2	13	12.32$_{\pm 17.25}$	50	100	4.94$_{\pm 1.30}$	0	2	14	21.78$_{\pm 44.35}$	0	0	5	32.08$_{\pm 20.41}$
MatchAnything + Koide *et al.* [📷]	1	14	45	5.90$_{\pm 17.34}$	0	1	7	22.25$_{\pm 60.57}$	0	5	29	7.70$_{\pm 12.69}$	0	85	7.16$_{\pm 1.92}$	0	13	54	8.38$_{\pm 16.17}$	0	0	12	37.68$_{\pm 26.45}$
MatchAnything + Taylor *et al.* [📷]	21	74	96	1.48$_{\pm 1.85}$	12	65	98	2.72$_{\pm 0.81}$	0	9	51	4.97$_{\pm 6.77}$	25	100	5.69$_{\pm 1.17}$	0	31	80	6.62$_{\pm 42.65}$	1	27	63	7.31$_{\pm 6.01}$
MatchAnything + MatchAnything [📷]	14	70	99	1.58$_{\pm 0.85}$	8	71	99	2.63$_{\pm 0.77}$	3	24	66	3.38$_{\pm 4.31}$	50	100	5.13$_{\pm 0.76}$	0	32	98	5.70$_{\pm 1.93}$	9	37	91	5.58$_{\pm 2.51}$
MatchAnything* + ReFineR	58	96	100	0.90$_{\pm 0.40}$	70	95	100	1.76$_{\pm 0.51}$	16	68	91	1.54$_{\pm 3.04}$	55	100	4.94$_{\pm 0.94}$	6	54	97	4.90$_{\pm 2.75}$	13	43	92	5.36$_{\pm 4.25}$

Table 1: Fine image to raw point cloud registration results (Section 5.2). Each image was registered using rendered point cloud normals.

with the image using photometric loss and Mutual Information, but here normal orientation values are very weakly correlated with real image color. This explains the large error values for GS-SLAM, since it completely diverges in most cases. Yuan *et al.*'s method does not seem robust for this fine registration task for most datasets. Their method extracts edges from the point cloud and matches them with 2D edges after projection. This procedure may be efficient for the specific Livox scans, which were taken at fixed positions and therefore do not contain occluded edges. The occluded edges are not filtered by the algorithm and could create unwanted supervision and wrong alignment. Additionally, 3D edges might be more difficult to extract in the first place, using plane intersection for sparser point clouds, as in CARLA and nuScenes. Taking advantage of MatchAnything's and ReFineR's strengths, we combine the two algorithms into our complete registration pipeline (two stages). A multi-stage approach was also considered by Koide *et al.* using SuperGlue [49]. To do so, we propose to use our point splatting rendering of the point cloud as input to MatchAnything instead of point rasterization in order to create a denser image (referred to as MatchAnything*). This choice is closely related to the reconstruction procedure of ETH3D, which fills the empty pixels using nearest neighbors before extracting SIFT [38] features, and we show in the ablation that our design slightly increases the reliability of MatchAnything alone. This combination appears fruitful as performance is increased under all registration thresholds (58% under 1px on ETH3D, 68% under 2px on CARLA). This also proves that ReFineR is particularly performant as a "refinement" tool capable of improving registration even at a few pixels of displacement. In comparison, two successive inferences of MatchAnything are not able to reach as fine results. As for computation times, MatchAnything is generally constant, under 2s per image, while the other methods vary between 10s and 40s because of the variable number of iterations and point cloud size. On CARLA, ReFineR and Koide *et al.*'s methods took about 10s, while Yuan *et al.*'s took 45s per image. We provide more insight on computation times in supplementary Section C.6.

5.3 Registration with intensity point clouds

In this second setup, we consider the point cloud intensity when available. This intensity may either be a reflectivity value returned by each emitted light ray, which is related to the measured range or material absorption (Livox, nuScenes), or it may be, in some cases, a color value. Color is specific to the Faro Focus scanner of ETH3D and Tanks and Temples, and the RGBD camera of RGBDScenes, as color comes from an embedded, pre-calibrated camera.

	ETH3D				Tanks and Temples				Livox			nuScenes				RGBDScenes			
	pix@ (%↑)			p_e(px↓)	pix@ (%↑)			p_e(px↓)	pix@ (%↑)		p_e(px↓)	pix@ (%↑)			p_e(px↓)	pix@ (%↑)			p_e(px↓)
	1	2	5		2	3	5		5	10		3	5	10		3	5	10	
One stage																			
GS-SLAM [📼]	19	40	47	7.54±39.00	45	62	67	2.06±59.53	20	25	56.41±32.79	0	8	8	131.07±94.68	1	5	11	37.54±29.24
Koide *et al.* [📼]	57	98	100	0.93±0.29	51	95	100	1.99±0.50	75	100	4.66±0.28	0	10	40	11.14±7.43	25	45	85	5.32±22.24
Taylor *et al.* [📼]	1	1	29	6.76±5.05	3	7	23	7.25±3.48	0	75	9.09±6.69	0	0	15	22.44±46.66	0	5	55	9.07±6.65
MatchAnything [📼]	35	87	100	1.12±0.61	38	87	100	2.13±0.63	75	100	4.17±0.92	4	64	100	4.72±3.19	81	100	100	1.81±0.93
ReFineR	61	70	73	0.71±3.62	71	91	98	1.93±0.94	45	95	5.17±1.68	6	35	84	6.99±2.88	12	22	81	8.07±3.00
Two stages																			
MatchAnything + GS-SLAM [📼]	32	66	73	1.30±38.55	36	74	83	2.12±38.45	25	25	65.64±30.08	0	11	17	139.64±101.83	5	5	10	35.67±31.37
MatchAnything + Koide *et al.* [📼]	57	100	100	0.95±0.25	48	96	100	2.02±0.48	50	100	4.99±0.71	0	18	68	7.99±10.92	38	60	85	3.96±27.30
MatchAnything + Taylor *et al.* [📼]	68	89	98	0.81±1.45	36	91	97	2.23±0.95	70	100	4.48±0.63	4	45	80	5.35±26.49	50	78	93	2.99±3.53
MatchAnything + MatchAnything [📼]	62	96	100	0.90±0.43	52	96	99	1.98±0.64	75	100	4.12±0.57	2	54	100	4.83±1.24	87	100	100	2.00±0.64
MatchAnything* + ReFineR	86	100	100	0.57±0.29	74	95	100	1.90±0.55	75	100	4.17±0.63	10	62	100	4.29±1.40	99	100	100	1.39±0.54

Table 2: Fine image to intensity point cloud registration results (Section 5.3). Each image was registered using rendered point cloud reflectivity or grayscale value.

We observe that intensity provides, in general, richer information for registration. This is naturally the case for the color channel, but also for reflectivity, as drawn patterns (white arrows on roads...) will now appear on reflectivity images, thanks to the uneven absorption strength of colored surfaces. Examples of rendered images are shown in supplementary Fig. 4. Thanks to the color information, photometric losses become a relevant supervision channel, which allows GS-SLAM to perform well on ETH3D and Tanks and Temples and to obtain competitive results on RGBDScenes, but it fails on the remaining reflectivity sets. Koide *et al.*'s method is a little more resilient as it compares intensity histograms to extract correlation information between image color and LiDAR reflectivity. Our 2-stage variant with the MatchAnything* + ReFineR combination obtained near 100% subpixel registration for ETH3D and best results at lower thresholds for all sets. We show qualitative results of both experiments in supplementary Fig. 7 and Fig. 8.

5.4 Ablation studies

Our design choices are evaluated in Table 3 on all datasets with raw point clouds. As before, each experiment is initialized at a 10px mean error. Experiments (I-IV) incorporate the MatchAnything* first stage with variations of ReFineR. Experiment (V) shows the performance gain from MatchAnything* alone. Experiment (VI) compares two stages of MatchAnything and MatchAnything*.

Automatic rescaling (I): First, we test the impact of not using our automatic point cloud rescaling factor, keeping the point cloud at its acquired scale. In practice, this implies that since the learning rate remains fixed, translation is less updated at each iteration, and the optimization is more prone to fall into local translation minima. This is why we observe a more significant impact on the CARLA set, as this set has the largest point clouds, being simulated outdoors and on large streets and with large depth ranges. Practically, adding an automatic rescale factor also serves the purpose of fixing the hyperparameters independently of the point cloud scale (learning rate, point size).

Splatting parameters (II) & (III): We change the values of the opacity and point size parameters used for splatting. These parameters are directly linked to the loss landscape of the optimization, as a larger point size creates larger splats, which can reduce the fine registration ability of the method by having too smooth images, but a smaller point size creates rendering holes, which can bias the gradient computations. We found that using a size 0.5 gave reasonable results. Similarly, opacity controls a form of smoothness in the optimiza-

	ETH3D				Tanks and Temples					CARLA				Livox			nuScenes				RGBDScenes			
	pix@ (%↑)			p_ϵ(px↓)	pix@ (%↑)				p_ϵ(px↓)	pix@ (%↑)			p_ϵ(px↓)	pix@ (%↑)		p_ϵ(px↓)	pix@ (%↑)			p_ϵ(px↓)	pix@ (%↑)			p_ϵ(px↓)
	1	2	5		1	2	3	5		1	2	5		5	10		3	5	10		3	5	10	
(I)																								
W/o rescale	57	97	100	0.90±0.38	0	70	95	100	1.75±0.48	17	61	88	1.66±2.97	50	100	5.10±0.96	9	50	95	5.00±3.59	10	44	93	5.43±3.35
MatchAnything* + ReFineR (W/ rescale)	58	96	100	0.90±0.40	0	70	95	100	1.76±0.51	16	68	91	1.54±3.04	55	100	4.94±0.94	6	54	97	4.90±2.75	13	43	92	5.36±4.25
(II)																								
Opacity-0.5	51	98	100	0.99±0.37	0	70	88	100	1.78±0.55	9	56	93	1.89±2.90	80	100	4.31±1.18	1	48	96	5.26±3.05	4	23	82	6.69±3.36
Opacity-0.8	35	98	100	1.09±0.39	0	70	89	100	1.86±0.62	2	40	81	2.12±2.87	25	100	5.15±1.44	0	37	95	5.47±2.30	4	12	72	7.51±3.69
MatchAnything* + ReFineR (Opacity-0.2)	58	96	100	0.90±0.40	0	70	95	100	1.76±0.51	16	68	91	1.54±3.04	55	100	4.94±0.94	6	54	97	4.90±2.75	13	43	92	5.36±4.25
(III)																								
Size-0.2	63	98	100	0.85±0.38	0	70	95	100	1.71±0.61	13	60	86	1.63±3.67	30	100	5.68±1.30	5	42	91	6.08±12.93	7	43	94	5.53±3.19
Size-0.8	40	95	100	1.15±0.47	0	68	85	100	1.89±0.70	6	55	94	1.81±3.04	45	100	5.02±1.21	0	49	94	5.03±4.88	5	38	86	5.68±3.33
MatchAnything* + ReFineR (Size-0.5)	58	96	100	0.90±0.40	0	70	95	100	1.76±0.51	16	68	91	1.54±3.04	55	100	4.94±0.94	6	54	97	4.90±2.75	13	43	92	5.36±4.25
(IV)																								
MatchAnything features	5	34	81	3.09±5.99	0	12	51	89	2.96±1.56	0	2	50	4.97±5.83	70	100	4.73±0.82	0	21	77	6.91±6.95	4	34	87	5.86±3.39
HOG features	63	98	100	0.86±0.41	0	65	100	100	1.79±0.41	13	58	93	1.72±2.36	45	100	5.05±0.65	10	60	99	4.55±2.11	11	48	95	5.08±2.99
MatchAnything* + ReFineR (GOM features)	58	96	100	0.90±0.40	0	70	95	100	1.76±0.51	16	68	91	1.54±3.04	55	100	4.94±0.94	6	54	97	4.90±2.75	13	43	92	5.36±4.25
(V)																								
MatchAnything	8	65	100	1.80±0.83	0	1	46	95	3.05±0.94	0	14	65	4.12±3.88	65	100	4.44±0.86	0	28	88	5.71±2.56	0	28	95	6.07±1.90
MatchAnything*	12	66	100	1.61±0.74	0	14	77	100	2.67±0.59	0	9	85	3.21±2.44	50	100	4.93±1.31	5	30	100	5.69±1.48	6	38	83	5.99±3.20
(VI)																								
MatchAnything + MatchAnything	14	70	99	1.58±0.85	0	8	71	99	2.63±0.77	3	24	66	3.38±4.31	50	100	5.13±0.76	0	32	98	5.70±1.93	9	37	91	5.58±2.51
MatchAnything* + MatchAnything*	11	69	100	1.63±0.72	0	9	80	100	2.60±0.51	0	12	79	3.33±2.51	50	100	4.93±1.26	5	29	97	5.62±3.63	5	34	80	7.11±3.84
MatchAnything + ReFineR	58	96	100	0.92±0.43	0	71	90	100	1.84±0.55	17	69	92	1.53±2.96	35	100	5.18±1.48	6	52	98	4.89±1.69	5	39	82	5.88±4.47
MatchAnything* + ReFineR	58	96	100	0.90±0.40	0	70	95	100	1.76±0.51	16	68	91	1.54±3.04	55	100	4.94±0.94	6	54	97	4.90±2.75	13	43	92	5.36±4.25

Table 3: Ablation study (Section 5.4).

tion, as low opacity increases the individual contribution of each Gaussian, making the loss more robust, but also enables occluded structures to appear in the render. We chose to fix the opacity at 0.2.

Loss functions (IV): We experimented with the idea of using intermediate MatchAnything features as a replacement for the GOM loss. The intuition was to leverage the model's cross-modality features at a dense and multiscale level. We compared features at scales 2, 4, and 8, computed for the initial RGB and rendered images, using an L2 loss. Due to memory constraints, rendered images at 1024px were differentiably resized to 664px before giving them as input to MatchAnything, which we suspect had an impact on the comparatively weak results for this experiment. Additionally, we guess that the presence of pooling layers in the CNN encoder reduces the fine-registration abilities, as these layers divide the image resolution by 2, 4, and 8. We believe that a less naive approach could certainly be proposed to better leverage the MatchAnything features, which may be investigated in future work. Inspired by Guislain *et al.*'s work [19], we propose to implement a differentiable version of the HOG features [11]. To compute the HOG features, one should first estimate the image gradients, then apply a smooth histogram convolution kernel onto the image, creating cell histograms, followed by a local block normalization of each histogram bin along blocks of cells (an explicit formulation of differentiable HOGs can be found in Supplementary Section A.5). The idea behind this loss is quite similar to the GOM, which locally compares images' gradients. This is probably why we observe, on average, comparable results between GOM and HOG features. On average, GOM loss is a little faster than HOG, as the former took on average 0.05s per iteration, whereas the latter took 0.2s.

MatchAnything (V) & (VI): Comparing performance between MatchAnything inference on image rasters and Gaussian splatting renders, we observe a net gain in the final pixel error for the CARLA and Tanks and Temples datasets, hinting that a denser image is a better quality input for MatchAnything. A very small improvement can be observed on the ETH3D dataset, which is coherent, as this dataset already has very dense point clouds that completely fill the image, even with simple rasterization. When comparing two consecutive inferences of MatchAnything and MatchAnything*, this gap closes, demonstrating the limits of the

MatchAnything model itself. The similar performances when adding ReFineR as a second stage tend to show that our method is capable of handling some of MatchAnything's weaker cases, but given results from (VI), we chose to keep MatchAnything* for our final pipeline.

5.5 Registration at scale

To capture the performance of the MatchAnything* + ReFineR combination, we vary the initialization of our registration experiment from 0 to 100 pixels (Table 4). Starting from a 0px perturbation, we observe that ReFineR alone keeps the registration stable under 2px 76% of the time, which is about three times as often as MatchAnything alone, but when combined, the fine registration capabilities of ReFineR mitigate MatchAnything's instabilities. As the perturbation increases, our pipeline proves to stay relatively robust even at the large 100px initialization, keeping its median final reprojection error below 2px. Another interesting result is the net reduction in translation error, probably caused by the automatic rescaling factor, which allows better conditioning for the joint optimization of rotation and translation. A similar experiment was carried out on the ETH3D set in supplementary Table 10.

	CARLA										
	pix@ (%↑)								p_e(px↓)	rot (°↓)	transl (cm↓)
	1	2	3	5	10	20	50	100			
0px initialization											
MatchAnything	3	26	44	85	98	100	100	100	$3.20_{\pm 2.60}$	$0.14_{\pm 0.08}$	$33.94_{\pm 27.77}$
ReFineR	20	76	92	100	100	100	100	100	$1.47_{\pm 0.79}$	$0.06_{\pm 0.04}$	$2.99_{\pm 1.74}$
MatchAnything* + ReFineR	16	71	83	95	96	99	100	100	$1.49_{\pm 2.72}$	$0.05_{\pm 0.09}$	$3.03_{\pm 6.32}$
5px initialization											
MatchAnything	2	17	39	78	95	99	100	100	$3.35_{\pm 3.20}$	$0.13_{\pm 0.12}$	$32.50_{\pm 32.13}$
ReFineR	11	63	82	95	99	100	100	100	$1.68_{\pm 1.72}$	$0.06_{\pm 0.07}$	$3.29_{\pm 3.82}$
MatchAnything* + ReFineR	16	66	84	96	99	100	100	100	$1.49_{\pm 1.59}$	$0.06_{\pm 0.06}$	$3.05_{\pm 3.72}$
10px initialization											
MatchAnything	0	14	30	65	90	100	100	100	$4.12_{\pm 3.88}$	$0.15_{\pm 0.17}$	$37.95_{\pm 53.56}$
ReFineR	12	44	57	74	92	100	100	100	$2.36_{\pm 3.45}$	$0.08_{\pm 0.25}$	$4.44_{\pm 7.52}$
MatchAnything* + ReFineR	16	68	81	91	97	99	100	100	$1.54_{\pm 3.04}$	$0.06_{\pm 0.09}$	$3.05_{\pm 6.39}$
20px initialization											
MatchAnything	0	11	19	49	85	96	100	100	$5.13_{\pm 4.94}$	$0.19_{\pm 0.24}$	$43.47_{\pm 66.34}$
ReFineR	2	14	18	21	26	82	100	100	$16.67_{\pm 7.70}$	$0.94_{\pm 0.64}$	$26.50_{\pm 16.88}$
MatchAnything* + ReFineR	18	66	82	90	94	98	100	100	$1.54_{\pm 4.19}$	$0.06_{\pm 0.22}$	$3.06_{\pm 8.75}$
50px initialization											
MatchAnything	0	8	18	34	65	93	98	99	$6.79_{\pm 14.99}$	$0.23_{\pm 0.32}$	$50.53_{\pm 79.29}$
ReFineR	0	0	0	0	0	0	56	100	$49.60_{\pm 6.50}$	$3.63_{\pm 1.19}$	$82.69_{\pm 29.06}$
MatchAnything* + ReFineR	17	54	73	80	92	96	99	100	$1.64_{\pm 9.36}$	$0.06_{\pm 0.29}$	$3.35_{\pm 20.37}$
100px initialization											
MatchAnything	1	3	9	31	55	77	95	100	$8.32_{\pm 15.98}$	$0.28_{\pm 0.87}$	$81.66_{\pm 202.21}$
ReFineR	0	0	0	0	0	0	0	52	$99.66_{\pm 13.05}$	$7.43_{\pm 2.28}$	$170.25_{\pm 56.97}$
MatchAnything* + ReFineR	12	52	60	69	80	89	95	99	$1.98_{\pm 20.24}$	$0.07_{\pm 0.65}$	$4.29_{\pm 41.36}$

Table 4: Registration at scale experiment, each camera pose was initialized at increasing pixel errors (Section 5.5).

6 Limitations of this work

Several limitations were not addressed in this paper and still represent challenges for the fine image-to-point cloud registration task, which we will discuss here. First, we show in this paper that there is a strong dependence between registration capabilities and the point cloud quality. In particular, floating individual points in the 3D model and in front of the camera might be rendered, biasing the optimization. This is also the case for dynamic objects in

the 3D scene, which create discrepancies between the real image and the rendered view. This issue was mitigated by manually removing such points in preprocessing, but results on nuScenes vs ETH3D show that noisier point clouds represent a greater challenge. In general, for any analysis-by-synthesis method, the correct rendering of occlusion is a major challenge: by standing close to walls or having a sparse point cloud, one might render points that are supposedly concealed by others, which, in the end, creates discrepancies with the real image once again.

For RGBDScenes, ETH3D, and Tanks and Temples datasets, color is different from reflectivity as it does not correspond to real frequency information associated with a given light ray/point. Instead, color is captured by a separate camera sensor and synchronized in post-processing with the 3D data and precomputed calibration. This implies that some error in color projection can be caused by bad sensor registration, which is precisely the problem we are aiming to solve. For our current resolution of 1024 px height, however, this error seems to be negligible (or less than a pixel), but it will surely appear at finer resolutions.

Finally, our manual dataset and evaluation present their limitations. On the one hand, as we saw for the scale estimation method, pixel error is entirely dependent on the chosen point's depth: because of parallax, farther points will be less displaced than closer points for a fixed translation. This means that pixel error is simply a function of the depth distribution, which can be manipulated by selecting far points only. During our data acquisition, we ensured not to fall into this loophole by selecting evenly distributed points in depth. Second, camera estimation through point-pixel selection is itself a biased process toward certain kinds of rigid transforms. Indeed, because of the uncertainty matrix S defined in Eq. (3), some transform directions have less observability than others (e.g., the z axis in camera coordinates), which would not occur in multi-sensor setups with LiDAR-odometry registration. Third, due to uncertainty in selecting "the right pixel" for our manual registration, we were unable to achieve sub-pixel registration on our three manual sets. In Table 6 of the Supplementary, we observe final reprojection errors of 2px for Tanks and Temples and 3.5px for nuScenes on average. Additionally, the 1px threshold column in the Tanks and Temples ablation in Table 3 displays a complete column of 0%, while our method achieves subpixel registration on the similar ETH3D set. This evidence suggests that, with ReFineR, we reached the limits of manual registration, and that the true capabilities of our method are probably slightly better at the pixel level than the results we obtained from manual registration.

7 Conclusion

Through the paradigm of analysis-by-synthesis, we present ReFineR: a generalist, differentiable, and training-free approach for fine-level image-to-point cloud registration. Combined with MatchAnything, ReFineR achieves state-of-the-art results on heterogeneous image and point cloud data and is capable of fine registration even at large camera perturbations up to 100px. Furthermore, we have shown that thanks to a first-order analysis of the pinhole camera model, it is possible to remove the point cloud scale ambiguity from the optimization parameters, creating a unified method for both indoor and outdoor scenes.

References

[1] Pei An, Junfeng Ding, Siwen Quan, Jiaqi Yang, You Yang, Qiong Liu, and Jie Ma. Survey of extrinsic calibration on lidar-camera system for intelligent vehicle: Challenges, approaches, and trends. *IEEE Transactions on Intelligent Transportation Systems*, 25 (11):15342–15366, 2024. doi: 10.1109/TITS.2024.3419758.

[2] Gilad Baruch, Zhuoyuan Chen, Afshin Dehghan, Tal Dimry, Yuri Feigin, Peter Fu, Thomas Gebauer, Brandon Joffe, Daniel Kurz, Arik Schwartz, and Elad Shulman. ARKitScenes - a diverse real-world dataset for 3D indoor scene understanding using mobile RGB-d data. In *Thirty-Fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*. URL https://openreview.net/forum?id=tjZjv_qh_CE.

[3] Hugo Blanc, Jean-Emmanuel Deschaud, and Alexis Paljic. Raygauss: Volumetric gaussian-based ray casting for photorealistic novel view synthesis. In *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 1808–1817, 2025. doi: 10.1109/WACV61041.2025.00183.

[4] Eric Brachmann and Carsten Rother. Visual camera re-localization from rgb and rgb-d images using dsac. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9):5847–5865, 2022. doi: 10.1109/TPAMI.2021.3070754.

[5] Eric Brachmann, Tommaso Cavallari, and Victor Adrian Prisacariu. Accelerated coordinate encoding: Learning to relocalize in minutes using rgb and poses. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5044–5053, 2023. doi: 10.1109/CVPR52729.2023.00488.

[6] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11618–11628, 2020. doi: 10.1109/CVPR42600.2020.01164.

[7] Nicholas Carlevaris-Bianco, Arash K Ushani, and Ryan M Eustice. University of michigan north campus long-term vision and lidar dataset. *The International Journal of Robotics Research*, 35(9):1023–1035, 2016. doi: 10.1177/0278364915614638.

[8] Juan Castorena, Ulugbek S. Kamilov, and Petros T. Boufounos. Autocalibration of lidar and optical cameras via edge alignment. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2862–2866, 2016. doi: 10.1109/ICASSP.2016.7472200.

[9] Xu Chen, Zijian Dong, Jie Song, Andreas Geiger, and Otmar Hilliges. Category level object pose estimation via neural analysis-by-synthesis. In *European Conference on Computer Vision (ECCV)*, 2020.

[10] Yuzhou Cheng, Jianhao Jiao, Yue Wang, and Dimitrios Kanoulas. LoGS: Visual localization via gaussian splatting with fewer training images. *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 15029–15036.

[11] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 886–893 vol. 1, 2005. doi: 10.1109/CVPR.2005.177.

[12] Pierre Dellenbach, Jean-Emmanuel Deschaud, Bastien Jacquet, and François Goulette. Ct-icp: Real-time elastic lidar odometry with loop closure. In *International Conference on Robotics and Automation (ICRA)*, pages 5580–5586, 2022. doi: 10.1109/ICRA46639.2022.9811849.

[13] Jérémie Deray and Joan Solà. Manif: A micro Lie theory library for state estimation in robotics applications. *Journal of Open Source Software*, 5(46):1371, 2020. doi: 10.21105/joss.01371.

[14] Alexey Dosovitskiy, Germán Ros, Felipe Codevilla, Antonio M. López, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Conference on Robot Learning*, 2017. URL https://carla.org/.

[15] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981. ISSN 0001-0782. doi: 10.1145/358669.358692. URL https://doi.org/10.1145/358669.358692.

[16] A Geiger, P Lenz, C Stiller, and R Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013. doi: 10.1177/0278364913491297.

[17] Ben Glocker, Shahram Izadi, Jamie Shotton, and Antonio Criminisi. Real-time RGB-d camera relocalization. In *International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE. URL https://www.microsoft.com/en-us/research/publication/real-time-rgb-d-camera-relocalization/.

[18] Prajnan Goswami, Tianye Ding, Feng Liu, and Huaizu Jiang. UniCorrn: Unified correspondence transformer across 2D and 3D. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

[19] Maximilien Guislain, Julie Digne, Raphaëlle Chaine, and Gilles Monnier. Fine scale image registration in large-scale urban lidar point sets. *Computer Vision and Image Understanding*, 157:90–102, 2017. ISSN 1077-3142. doi: https://doi.org/10.1016/j.cviu.2016.12.004. URL https://www.sciencedirect.com/science/article/pii/S1077314216302016.

[20] Xingyi He, Hao Yu, Sida Peng, Dongli Tan, Zehong Shen, Hujun Bao, and Xiaowei Zhou. MatchAnything: Universal Cross-Modality Image Matching with Large-Scale Pre-Training. *arXiv e-prints*, art. arXiv:2501.07556, January 2025. doi: 10.48550/arXiv.2501.07556.

[21] Peter Henry, Dieter Fox, Achintya K. Bhowmik, and Rajiv Mongia. Patch volumes: Segmentation-based consistent mapping with rgb-d cameras. *2013 International Conference on 3D Vision*, pages 398–405, 2013.

[22] Qi-Xing Huang, Hai Wang, and Vladlen Koltun. Single-view reconstruction via joint analysis of image and shape collections. *ACM Transactions on Graphics (TOG)*, 34:1 – 10, 2015.

[23] Lukas Höllein, Aljavz Bovzivc, Michael Zollhofer, and Matthias Nießner. 3DGS-LM: Faster Gaussian-Splatting Optimization with Levenberg-Marquardt. *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, abs/2409.12892, 2024. URL https://github.com/lukasHoel/3DGS-LM?tab=readme-ov-file.

[24] Ganesh Iyer, R. Karnik Ram, J. Krishna Murthy, and K. Madhava Krishna. Calibnet: Geometrically supervised extrinsic calibration using 3d spatial transformer networks. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1110–1117, 2018. doi: 10.1109/IROS.2018.8593693.

[25] Shuhao Kang, Youqi Liao, Jianping Li, Fuxun Liang, Yuhao Li, Xianghong Zou, Fangning Li, Xieyuanli Chen, Zhen Dong, and Bisheng Yang. Cofii2p: Coarse-to-fine correspondences-based image to point cloud registration. *IEEE Robotics and Automation Letters*, 9(11):10264–10271, 2024. doi: 10.1109/LRA.2024.3466068.

[26] Alireza G. Kashani, Michael J. Olsen, Christopher E. Parrish, and Nicholas Wilson. A review of lidar radiometric processing: From ad hoc intensity correction to rigorous radiometric calibration. *Sensors*, 15(11):28099–28128, 2015.

[27] Bingxin Ke, Kevin Qu, Tianfu Wang, Nando Metzger, Shengyu Huang, Bo Li, Anton Obukhov, and Konrad Schindler. Marigold: Affordable adaptation of diffusion-based image generators for image analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–18, 2025. doi: 10.1109/TPAMI.2025.3591076.

[28] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkuehler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), July 2023. ISSN 0730-0301. doi: 10.1145/3592433. URL https://doi.org/10.1145/3592433.

[29] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics*, 36 (4), July 2017. ISSN 0730-0301. doi: 10.1145/3072959.3073599. URL https://doi.org/10.1145/3072959.3073599.

[30] Kenji Koide, Shuji Oishi, Masashi Yokozuka, and Atsuhiko Banno. General, single-shot, target-less, and automatic lidar-camera extrinsic calibration toolbox. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 11301–11307, 2023. doi: 10.1109/ICRA48891.2023.10160691.

[31] Kevin Lai, Liefeng Bo, and Dieter Fox. Unsupervised feature learning for 3d scene labeling. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3050–3057, 2014. doi: 10.1109/ICRA.2014.6907298.

[32] Christoph Lassner and Michael Zollhöfer. Pulsar: Efficient sphere-based neural rendering. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1440–1449, 2021.

[33] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. EPnP: An Accurate O(n) Solution to the PnP Problem. *International Journal of Computer Vision*, 81:155–166, 2008. doi: 10.1007/s11263-008-0152-6. URL https://www.tugraz.at/fileadmin/user_upload/Institute/ICG/Images/team_lepetit/publications/lepetit_ijcv08.pdf.

[34] Kenneth Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly of Applied Mathematics*, 2(2):164–168, 1944. ISSN 0033569X, 15524485. URL http://www.jstor.org/stable/43633451.

[35] Minhao Li, Zheng Qin, Zhirui Gao, Renjiao Yi, Chenyang Zhu, Yulan Guo, and Kai Xu. 2d3d-matr: 2d-3d matching transformer for detection-free registration between images and point clouds. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 14082–14092, 2023. doi: 10.1109/ICCV51070.2023.01299.

[36] Changkun Liu, Shuai Chen, Yash Bhalgat, Siyan Hu, Zirui Wang, Ming Cheng, Victor Adrian Prisacariu, and Tristan Braud. Gs-cpr: Efficient camera pose refinement via 3d gaussian splatting. In *Proceedings of the Thirteenth International Conference on Learning Representations (ICLR)*, 2025.

[37] Xiyuan Liu, Chongjian Yuan, and Fu Zhang. Targetless extrinsic calibration of multiple small fov lidars and cameras using adaptive voxelization. *IEEE Transactions on Instrumentation and Measurement*, 71:1–12, 2022. doi: 10.1109/TIM.2022.3176889.

[38] David G. Lowe. Object recognition from local scale-invariant features. *Proceedings of the Seventh IEEE International Conference on Computer Vision*, 2:1150–1157 vol.2.

[39] Xudong Lv, Boya Wang, Ziwen Dou, Dong Ye, and Shuo Wang. Lccnet: Lidar and camera self-calibration using cost volume network. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2888–2895, 2021. doi: 10.1109/CVPRW53098.2021.00324.

[40] Donald W. Marquardt. An Algorithm for Least-Squares Estimation of Nonlinear Parameters. *Journal of The Society for Industrial and Applied Mathematics*, 11:431–441, 1963.

[41] Hidenobu Matsuki, Riku Murai, Paul H. J. Kelly, and Andrew J. Davison. Gaussian splatting slam. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18039–18048, 2024. doi: 10.1109/CVPR52733.2024.01708.

[42] Bortolon Matteo, Theodore Tsesmelis, Stuart James, Fabio Poiesi, and Alessio Del Bue. 6dgs: 6d pose estimation from a single image and a 3d gaussian splatting model. In *European Conference on Computer Vision (ECCV)*, page 420–436, Berlin, Heidelberg, 2024. ISBN 978-3-031-72942-3. doi: 10.1007/978-3-031-72943-0_24. URL https://doi.org/10.1007/978-3-031-72943-0_24.

[43] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, December 2021. ISSN 0001-0782. doi: 10.1145/3503250. URL https://doi.org/10.1145/3503250.

[44] Juncheng Mu, Chengwei Ren, Weixiang Zhang, Liang Pan, Xiao-Ping Zhang, and Yue Gao. Diff2I2P: Differentiable image-to-point cloud registration with diffusion prior. *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 25777–25787, 2025.

[45] John A. Nelder and Roger Mead. A Simplex Method for Function Minimization. *Comput. J.*, 7:308–313, 1965.

[46] Ni Ou, Zhuo Chen, Xinru Zhang, and Junzheng Wang. Iterative camera-LiDAR extrinsic optimization via surrogate diffusion. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3069–3075, 2025. doi: 10.1109/IROS60139.2025.11247343.

[47] Gaurav Pandey, James R. McBride, Silvio Savarese, and Ryan M. Eustice. Automatic targetless extrinsic calibration of a 3d lidar and camera by maximizing mutual information. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, AAAI'12, page 2053–2059. AAAI Press, 2012.

[48] Siyu Ren, Yiming Zeng, Junhui Hou, and Xiaodong Chen. Corri2p: Deep image-to-point cloud registration via dense correspondence. *IEEE Transactions on Circuits and Systems for Video Technology*, 33(3):1198–1208, 2023. doi: 10.1109/TCSVT.2022.3208859.

[49] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4937–4946, 2020. doi: 10.1109/CVPR42600.2020.00499.

[50] Thomas Schöps, Johannes L. Schönberger, Silvano Galliani, Torsten Sattler, Konrad Schindler, Marc Pollefeys, and Andreas Geiger. A multi-view stereo benchmark with high-resolution images and multi-camera videos. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[51] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurélien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2443–2451, 2020. doi: 10.1109/CVPR42600.2020.00252.

[52] Zachary Taylor, Juan Nieto, and David Johnson. Multi-modal sensor calibration using a gradient orientation measure. *Journal of Field Robotics*, 32(5):675–695, 2015. doi: https://doi.org/10.1002/rob.21523. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21523.

[53] Bill Triggs, Philip F. McLauchlan, Richard I. Hartley, and Andrew W. Fitzgibbon. Bundle adjustment - a modern synthesis. In *Workshop on Vision Algorithms*, 1999.

[54] Guangming Wang, Yu Zheng, Yuxuan Wu, Yanfeng Guo, Zhe Liu, Yixiang Zhu, Wolfram Burgard, and Hesheng Wang. End-to-end 2d-3d registration between image and

lidar point cloud for vehicle localization. *IEEE Transactions on Robotics*, 41:4643–4662, 2025. doi: 10.1109/TRO.2025.3588454.

[55] Haiping Wang, Yuan Liu, Bing Wang, Yujing Sun, Zhen Dong, Wenping Wang, and Bisheng Yang. Freereg: Image-to-point cloud registration leveraging pretrained diffusion models and monocular depth estimators. In *Proceedings of the Twelfth International Conference on Learning Representations (ICLR)*, 2024.

[56] Olivia Wiles, Georgia Gkioxari, Richard Szeliski, and Justin Johnson. Synsin: End-to-end view synthesis from a single image. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7465–7475, 2020. doi: 10.1109/CVPR42600.2020.00749.

[57] Yueming Xu, Haochen Jiang, Zhongyang Xiao, Jianfeng Feng, and Li Zhang. Dg-slam: robust dynamic gaussian splatting slam with hybrid pose optimization. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA, 2025. ISBN 9798331314385.

[58] Luwei Yang, Ziqian Bai, Chengzhou Tang, Honghua Li, Yasutaka Furukawa, and Ping Tan. Sanet: Scene agnostic network for camera localization. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 42–51, 2019. doi: 10.1109/ICCV.2019.00013.

[59] Lin Yen-Chen, Pete Florence, Jonathan T. Barron, Alberto Rodriguez, Phillip Isola, and Tsung-Yi Lin. inerf: Inverting neural radiance fields for pose estimation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1323–1330, 2021. doi: 10.1109/IROS51168.2021.9636708.

[60] Chandan Yeshwanth, Yueh-Cheng Liu, Matthias Nießner, and Angela Dai. ScanNet++: A high-fidelity dataset of 3D indoor scenes. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 12–22. URL https://github.com/scannetpp/scannetpp.

[61] Wang Yifan, Felice Serena, Shihao Wu, Cengiz Öztireli, and Olga Sorkine-Hornung. Differentiable surface splatting for point-based geometry processing. *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH ASIA)*, 38(6), 2019.

[62] Chongjian Yuan, Xiyuan Liu, Xiaoping Hong, and Fu Zhang. Pixel-level extrinsic self calibration of high resolution lidar and camera in targetless environments. *IEEE Robotics and Automation Letters*, 6(4):7517–7524, 2021. doi: 10.1109/LRA.2021.3098923.

[63] Kaiwen Yuan, Zhenyu Guo, and Z. Jane Wang. Rggnet: Tolerance aware lidar-camera online calibration with geometric deep learning and generative model. *IEEE Robotics and Automation Letters*, 5(4):6956–6963, 2020. doi: 10.1109/LRA.2020.3026958.

A Theory and implementation

A.1 Pinhole camera model

The pinhole camera model computes, for each point $X \in \mathbb{R}^3$ in space, the projection $x = \pi(X)$ onto the camera plane, in pixel units. Each camera is defined by its respective intrinsics and extrinsic parameters $K_i \in \mathbb{R}^{3\times3}$ and $K_e \in SE(3)$ such that:

$$\pi(X) = \varphi(K_i \varphi\left(K_e \varphi^{-1}(X)\right)) \tag{8}$$

with φ and φ^{-1} defined by

$$\varphi(x_1,\ldots,x_n) = (x_1/x_n,\ldots,x_{n-1}/x_n) \tag{9}$$

$$\varphi^{-1}(x_1,\ldots,x_{n-1}) = (x_1,\ldots,x_{n-1},1) \tag{10}$$

A.2 Reprojection error asymptotic development

Writing $K_i = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}$ and $A(\tau,X) = \exp(\theta_\times)X + V(\theta)\rho = \begin{pmatrix} X_X & X_Y & X_Z \end{pmatrix}^T$, with

$$V(\theta) = I_3 + \frac{1-\cos(\|\theta\|)}{\|\theta\|^2}\theta_\times + \frac{\|\theta\| - \sin(\|\theta\|)}{\|\theta\|^3}\theta_\times^2 \tag{11}$$

and $._\times$ the antisymmetric operator, we then have from Eq. (8):

$$\pi(\tau,X) = \varphi(K_i A(\tau,X)) \tag{12}$$

$$= \begin{pmatrix} f_x \frac{X_X}{X_Z} + c_x \\ f_y \frac{X_Y}{X_Z} + c_y \end{pmatrix} \tag{13}$$

Therefore

$$\frac{\mathcal{D}\pi}{\mathcal{D}\tau}(0,X) = \frac{\partial\varphi}{\partial X}(K_i X) K_i \frac{\mathcal{D}A}{\mathcal{D}\tau}(0,X) \tag{14}$$

$$= \begin{pmatrix} \frac{1}{X_Z} & 0 & -\frac{f_x X_X + c_x X_Z}{X_Z^2} \\ 0 & \frac{1}{X_Z} & -\frac{f_y X_Y + c_y X_Z}{X_Z^2} \end{pmatrix} \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & X_Z & -X_Y & 1 & 0 & 0 \\ -X_Z & 0 & X_X & 0 & 1 & 0 \\ X_Y & -X_X & 0 & 0 & 0 & 1 \end{pmatrix} \tag{15}$$

$$= \begin{pmatrix} \frac{f_x}{X_Z} & 0 & -\frac{f_x X_X}{X_Z^2} \\ 0 & \frac{f_y}{X_Z} & -\frac{f_y X_Y}{X_Z^2} \end{pmatrix} \begin{pmatrix} 0 & X_Z & -X_Y & 1 & 0 & 0 \\ -X_Z & 0 & X_X & 0 & 1 & 0 \\ X_Y & -X_X & 0 & 0 & 0 & 1 \end{pmatrix} \tag{16}$$

$$= \begin{pmatrix} -\frac{x_c y_c}{f_y} & \frac{x_c^2+f_x^2}{f_x} & -\frac{f_x y_c}{f_y} & \frac{f_x}{X_Z} & 0 & -\frac{x_c}{X_Z} \\ -\frac{y_c^2+f_y^2}{f_y} & \frac{x_c y_c}{f_x} & \frac{f_y x_c}{f_x} & 0 & \frac{f_y}{X_Z} & -\frac{y_c}{X_Z} \end{pmatrix} \tag{17}$$

with $(x_c, y_c) = (f_x X_X/X_Z, f_y X_Y/X_Z)$

A.3 Scale estimation

Writing $X = \alpha X'$ and

$$J(X) = \frac{\mathcal{D}\pi}{\mathcal{D}\tau}(0,X) \tag{18}$$
$$= \begin{pmatrix} J_r(X) & J_t(X) \end{pmatrix} \tag{19}$$
$$= \begin{pmatrix} J_r(X) & \frac{1}{\alpha} J_{t'}(X) \end{pmatrix} \tag{20}$$

then

$$S = \frac{1}{N}\sum_{i=1}^{N} J(X_i)^T J(X_i) \tag{21}$$
$$= \frac{1}{N}\sum_{i=1}^{N} \begin{pmatrix} J_r(X_i)^T J_r(X_i) & \frac{1}{\alpha} J_r(X_i)^T J_{t'}(X_i) \\ \frac{1}{\alpha} J_{t'}(X_i)^T J_r(X_i) & \frac{1}{\alpha^2} J_{t'}(X_i)^T J_{t'}(X_i) \end{pmatrix} \tag{22}$$
$$= \begin{pmatrix} S_r & \frac{1}{\alpha} S_{rt'} \\ \frac{1}{\alpha} S_{rt'}^T & \frac{1}{\alpha^2} S_{t'} \end{pmatrix} \tag{23}$$
$$= \begin{pmatrix} S_r & S_{rt} \\ S_{rt}^T & S_t \end{pmatrix} \tag{24}$$

The eigenvalues between S_t and $S_{t'}$ are related by

$$\lambda_{t',i} = \alpha^2 \lambda_{t,i} \tag{25}$$

Therefore, equalizing the Hilbert-Schmidt norms:

$$\sum_{i=1}^{3} \lambda_{t',i}^2 = \sum_{i=1}^{3} \lambda_{r,i}^2 \tag{26}$$
$$\Leftrightarrow \alpha^4 \sum_{i=1}^{3} \lambda_{t,i}^2 = \sum_{i=1}^{3} \lambda_{r,i}^2 \tag{27}$$
$$\Leftrightarrow \alpha = \left(\sum_{i=1}^{3} \lambda_{r,i}^2 / \sum_{i=1}^{3} \lambda_{t,i}^2 \right)^{1/4} \tag{28}$$

Fig. 2 shows examples of 3 point clouds rescaled using the above equations for α.

A.4 Initialization at a target pixel error

We explain here how to perturb the ground truth camera pose to obtain a fixed pixel error. Starting back from Eq. (3) around the identity, and using the first-order approximation, we wish to find a transform $\delta\tau$ such that for a given pixel error Δ we have:

$$\delta\tau^T S \delta\tau = \Delta^2 \tag{29}$$

which by diagonalizing S reads:

$$\Delta^2 = (P\delta\tau)^T D (P\delta\tau) \tag{30}$$
$$= \sum_{i=1}^{6} \lambda_i (P\delta\tau)_i^2 \tag{31}$$

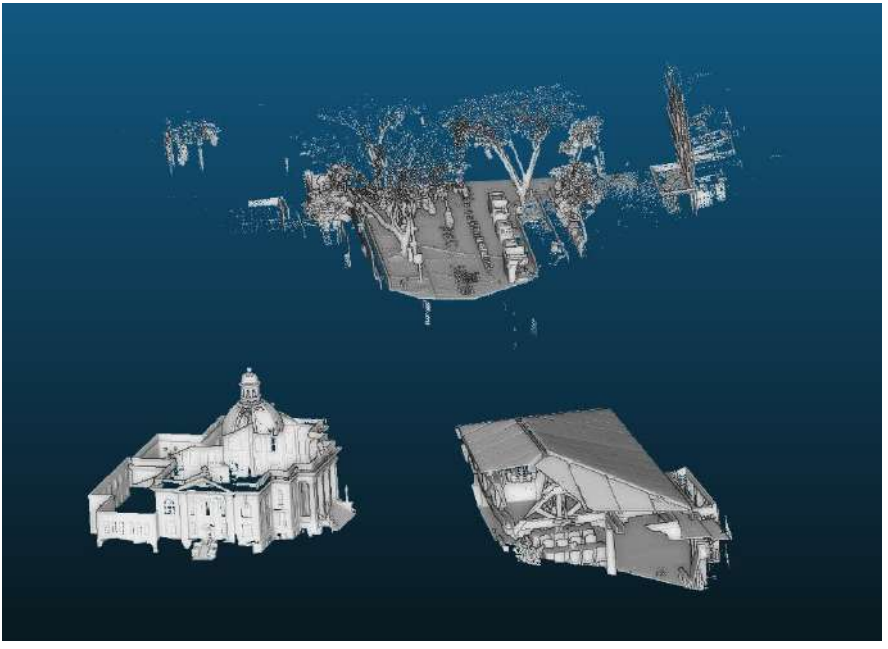


Figure 2: Three scenes from heterogeneous sources (nuScenes and Tanks and Temples), transformed to a unified scale using the method of Section 4.1.

with P an orthogonal matrix and $D = \text{diag}(\lambda_1, \dots \lambda_6)$. Therefore, such a $\delta\tau$ can be found by randomly selecting a unit vector $\tau = (\tau_1, \dots, \tau_6)^T$ and computing:

$$\delta\tau = \Delta P^T \begin{pmatrix} \frac{\tau_1}{\sqrt{\lambda_1}} \\ \vdots \\ \frac{\tau_6}{\sqrt{\lambda_6}} \end{pmatrix} \tag{32}$$

Note that this formulation remains an approximation of the pixel error due to the 1st order asymptotic development, which, in practice, might not produce exactly a pixel error of magnitude Δ for large perturbations.

A.5 Implementation of differentiable HOG features

As a comparison to Guislain *et al.* [19], who proposed using HOG features [11], we implement a differentiable version of this metric to fit our pipeline. In four steps:

1. Gradient computation: Starting from an image $(I_p)_p$ indexed by its pixel p, we compute the gradient magnitude and orientation images $(G_p, \Theta_p)_p$ using Sobel filters.

2. Cell histogram: A cell C is defined as a set of locally adjacent pixels: $C = \{p_1^C, \dots, p_{|C|}^C\}$. In practice, we used a kernel with cell size $k_C = 4$, with padding and stride hyperparameters. The cell size is then $|C| = k_C^2$. For each cell, a differentiable histogram F_m^C of unsigned

orientation is computed with M evenly spaced bin centers $(m_i)_{1\leq i\leq M} \in [0,\pi]^M$:

$$F_i^C = \sum_{p\in C} G_p f_\sigma(\Theta_p, m_i) \tag{33}$$

with $f_\sigma(\theta, m)$ the Gaussian's density with mean m and variance σ^2, and σ fixed to $0.5 * (m_2 - m_1)$

3. Block normalization: Similarly, a block B is defined as a set of adjacent cells: $B = \{C_1^B \dots C_{|B|}^B\}$, and the block size is defined using block kernel size $|B| = k_B^2$. We used $k_B = 4$. The normalized features $\left(F_i^{B,C}\right)_{B,C,i}$ are computed as:

$$N^B = \sqrt{\frac{1}{M|B|} \sum_{C\in B, 1\leq i\leq M} \left(F_i^C\right)^2} \qquad F_i^{B,C} = \frac{F_i^C}{N_B} \tag{34}$$

4. Loss function: Given two images I and I', and the features $\left(F_i^{B,C}\right)_{B,C,i}$, $\left(F_i'^{B,C}\right)_{B,C,i}$, the full feature loss is:

$$HOG(I, I') = \frac{\sum_{B,C,i} \left(F_i^{B,C} - F_i'^{B,C}\right)^2}{|B||C|M} \tag{35}$$

B Dataset preparation

B.1 ETH3D

From the ETH3D dataset [50], we reconstructed the global point clouds using ground truth alignment files and subsampled them using a 5 mm voxel grid. Then the normals' orientations were computed using CloudCompare. Fifty images spanning ten scenes were then randomly chosen to form our dataset, along with their ground truth camera positions and intrinsics. Thanks to its pixel-level refinement scheme, this dataset presents sub-pixel camera registration for resolutions up to 6048 × 4032. Additionally, each point cloud was acquired using a Faro Focus X 330, providing centimeter-level point precision.

B.2 Tanks and Temples

We chose the preprocessed Church and Meetingroom LiDAR point clouds from Tanks and Temples [29] and ten images from each scene. The available LiDAR point clouds were cleanly preprocessed by the original authors, and because of their high number of points, we randomly subsampled them, keeping 40% of the points during our experiments. Since camera positions and intrinsics were not given by the authors, we used a manual point-pixel matching protocol to estimate the extrinsic parameters as well as the shared intrinsics and 5 distortion parameters following the OpenCV distortion model. The final reprojection errors were around 2px on average.

B.3 nuScenes

We chose four scenes (1, 3, 7, and 12) from the nuScenes [6] dataset. For each scene, the LiDAR scans were aggregated into a global point cloud using CT-ICP [12], and we removed dynamic objects using the ground truth annotations and manual selection. We also applied histogram equalization on the LiDAR intensity values to enhance contrast. Five images were selected for each scene, all from the same front-right camera, taking care to choose images where the point cloud would be dense enough to fill most of the image when projected. Ground truth camera poses were similarly obtained using our manual matching.

B.4 Livox

We took 4 Livox scans with their associated images from Yuan *et al.*'s work [62], consisting of outdoor scenes. Despite its relatively small size, we chose to include this set for a fair comparison with Yuan *et al.*'s work [62], whose authors built a method for the specific case of Livox sensors.

B.5 RGBDScenes

We selected 5 images from scenes 01, 07, 09, and 13 of the RGBDScenes dataset [31]. Calibration was kept unchanged, and we used $(f_x, f_y, c_x, c_y) = (525, 525, 320, 240)$ as our intrinsics values. This set represents a drastic shift in point cloud quality from the other sets because of the sensor's resolution, but has the advantage of intrinsically having color intensity, which can be leveraged for our task.

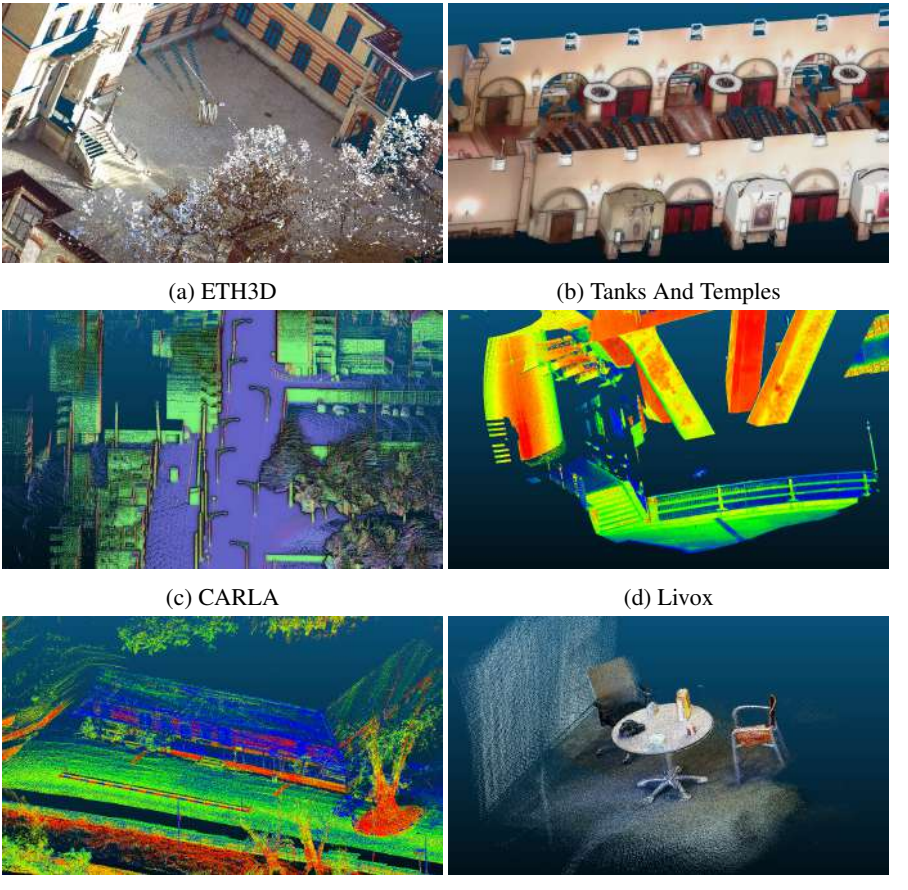
B.6 CARLA

We used the CARLA simulator [14] to generate five sequences of synthetic data. The sequences were recorded in Towns 2, 3, 4, 5, and 10. We placed a rotating LiDAR and two cameras on the main vehicle, one facing forward and the other facing right; the sensor parameters are given in Table 5. Parameters that are not mentioned are kept at their default values (see here[1]). We also provide for each modality the associated semantic information, which was used in Section C.8 to assess a 2D to 3D feature transfer quality in regard to a perturbed camera position. We noticed that the CARLA LiDAR had inaccuracies (missing objects or buildings); therefore, we manually selected 10 images from each sequence to mitigate the discrepancies between images and point clouds. Being synthetically generated, this dataset has the benefit of having floating-point-perfect image registration.

B.7 Common preprocessing

For these different sets, we computed normal directions using a local PCA of 50 neighbors. All images were resized to a fixed height of 1024px except for the RGBDScenes set, which was kept at a resolution of 640×480. Example point clouds are shown in Fig. 3, and images with point splatting renders are shown in Fig. 4.

[1] https://carla.readthedocs.io/en/latest/ref_sensors/

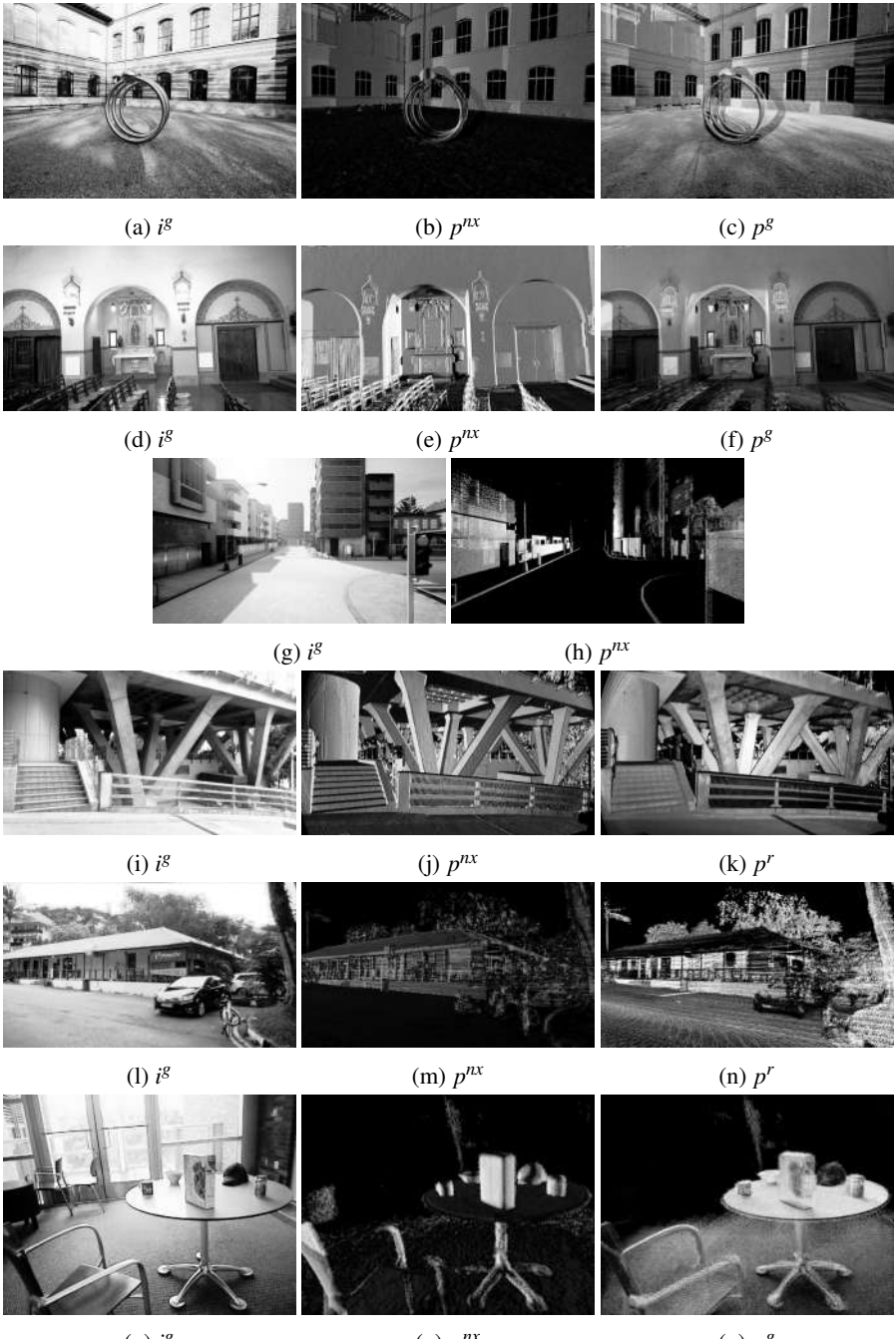


(a) ETH3D (b) Tanks And Temples

(c) CARLA (d) Livox

(e) nuScenes (f) RGBDScenes

Figure 3: Example point clouds from our benchmark dataset.



(a) i^g (b) p^{nx} (c) p^g

(d) i^g (e) p^{nx} (f) p^g

(g) i^g (h) p^{nx}

(i) i^g (j) p^{nx} (k) p^r

(l) i^g (m) p^{nx} (n) p^r

(o) i^g (p) p^{nx} (q) p^g

Figure 4: Example of images and rendered views from our dataset using point normals or intensity. From top to bottom: ETH3D, Tanks and Temples, CARLA, Livox, nuScenes, RGBDScenes.

Sensor / Parameter	value
Camera	
image_size_x	1920
image_size_y	1080
fov	90
LiDAR	
points_per_second	500,000
rotation_frequency	5
dropoff_general_rate	0
dropoff_intensity_limit	0
range	200
channels	64

Table 5: CARLA sensor setup.

B.8 Manual registration

For the nuScenes, Tanks and Temples, and Livox datasets, we manually computed our ground truth from 2D-3D correspondences and least-squares minimization. For each image, we manually selected around 15 point-pixel pairs, which we used to find the camera position.

For Tanks and Temples, we also noticed the presence of camera distortion and approximate intrinsic values, which could not be solved by simply estimating the camera position. For this set, we therefore jointly optimized the camera extrinsics, intrinsics, and distortion through least-squares minimization, keeping a single camera model (intrinsics and distortion) among all the images of the same scene. Final reprojection errors are given in Table 6.

This manual process may itself present biases as the final given errors are computed with respect to the selected matches, which may themselves be inaccurate. This was especially the case on the nuScenes set, for which the image edges were often blurry because of the video acquisition. RGBDScenes samples were kept with their official ground truth, as the point cloud noise prevented us from selecting accurate enough correspondences. For most images, we were able to significantly reduce the reprojection error down to a few pixels compared to the official registration for each dataset. Our approach is all the more motivated by finding a method to avoid this manual labor. We also observe from Table 6 that the initial error given by the official registration is on the order of 7-10 pixels, based on our manual correspondences on Tanks and Temples and nuScenes. This supports our assumption that fine registration should be studied from an initialization with a reprojection error of about 10 pixels. Qualitative results of our registration procedure are shown for nuScenes and Tanks and Temples in Fig. 5 and Fig. 6.

C Benchmark information and additional experiments

C.1 Benchmark methods

We compared ReFineR with 4 different state-of-the-art methods for fine-level registration.

MatchAnything [20] is a transformer-based feature-matching model trained on multi-



Figure 5: Manual registration results on nuScenes. Left: before manual registration, right: after.



Figure 6: Manual registration results on Tanks and Temples. Left: before manual registration, right: after.

modal images. We used their code and pretrained models available here[2]. For our benchmark, we used the RoMa model as it yielded better matches for our task.

We ran Yuan *et al.*'s method [62] from their repository code[3]. Depending on the scene modality, we used their predefined configurations in order to adapt the edge extraction to the point cloud scale. For instance, the voxel size parameter is set to 0.5 for indoor scenes and 1.0 for outdoor scenarios. All parameters can be found in their configuration file.

For Koide *et al.*'s method [30][4], we only changed the "*delta_trans_thresh*" parameter, which defines the translation threshold used to claim algorithm convergence. For indoor and outdoor scenes, we respectively set it to 1 and 10 cm.

GS-SLAM [41] is the least robust method for our benchmark as it was designed to jointly optimize camera poses and Gaussians' parameters in a Gaussian Splatting model, which only leverages photometric information between the 3D and 2D modalities, and is difficult to apply to broader ranges of point clouds (LiDAR, Livox, RGBD).

We reimplemented the method from Taylor *et al.* [52] with SciPy's version of the Nelder-Mead [45] optimizer.

Contrary to Yuan *et al.* and Koide *et al.*'s methods, ReFineR does not need any adjustment to the point cloud scale and scene modality.

C.2 Rotation and translation errors

Results from Section 5.2 and Section 5.3 are expressed in terms of rotation and translation errors in Table 7 and Table 8, respectively.

C.3 Qualitative results

Fig. 7 and Fig. 8 show qualitative results for registration on the raw point cloud and using point cloud intensity (grayscale value or LiDAR reflectivity).

C.4 Supervision channel

Apart from point cloud normals and intensity, as well as image RGB color, we experimented with different comparison modalities. Supervising the registration with the rendered depth ($\mathcal{C}_{\leftrightarrow} = \{(p^d, i^g)\}$), we generally observe a decrease in registration performance in all benchmarking datasets. We suspect that this is the case because the main source of gradients in depth images arises from discontinuities between objects and gradients converging towards vanishing points, whereas rendered normal images from point clouds provide a richer set of gradients, including regions where the surface orientation changes (*e.g.*, wall edges).

Another source of supervision could come from the monocular estimation of depth and normals with the Marigold diffusion model [27]. From a single RGB image, Marigold uses its trained geometric priors to estimate the normal orientation and normalized depth from a single image. For the normals, we kept the point splatting rendering of normals and reoriented them using the rendered view such that both the normal frames from the render and the generation coincide, and the rendered depth was normalized. We compared these true rendered channels with the monocular estimated normals and depth ($\mathcal{C}_{\leftrightarrow} =$

[2] https://huggingface.co/spaces/LittleFrog/MatchAnything/tree/main/imcui/third_party/MatchAnything

[3] https://github.com/hku-mars/livox_camera_calib

[4] https://github.com/koide3/direct_visual_lidar_calibration

$\{(p^{nx}, i^{nx}), (p^{ny}, i^{ny}), (p^{nz}, i^{nz})\}$ and $\mathcal{C}_{\leftrightarrow} = \{(p^{d}, i^{d})\}$). Table 9 shows that using the monocular normals produces better registration than the monocular depth, and sometimes matches the performance of the ReFineR baseline. Both synthetic modalities, however, generally struggle at finer registration levels. We may explain this because of the coarse resemblance between the image's gradient and its diffusion-generated counterpart, which allows for some edge supervision, but this supervision is limited by the current performance of diffusion models, which still create blurry edges, removing any pixel-level registration ability.

C.5 Registration at scale on ETH3D

We ran a similar experiment to the one in Section 5.5 on the ETH3D dataset. Results in Table 10 show strong performance of the MatchAnything* + ReFineR pipeline even at large perturbations such as 100px.

C.6 Computation times

The experiments were carried out on Ubuntu 24.04.2 LTS with an Intel Core Ultra 9 285K CPU and NVIDIA GeForce RTX 5090 GPU. We provide in Table 11 average computation times on CARLA with normals and ETH3D with RGB color. ReFineR on its own is on average as fast as Koide *et al.*'s method on CARLA, but a little slower when adding MatchAnything* as a first stage. Two main factors impact ReFineR's runtime: first, the quality of the point cloud: a more precise point cloud will require fewer iterations to be registered; second, the number of points, which directly correlates with the Gaussian splatting forward and backward passes as shown in Table 12. However, this second point is mitigated by the tile-based Gaussian Splatting rendering CUDA engine, which parallelizes computations. By contrast, Koide *et al.*'s method does not have such an implementation and scales more slowly with the number of points. Large computation times for GS-SLAM on CARLA are caused by its complete divergence, forcing the algorithm to run until the maximum number of iterations (arbitrarily fixed) is reached. On the contrary, on the ETH3D set with color registration, GS-SLAM is able to converge in roughly the same amount of time as ReFineR. From our experiments with ReFineR alone on the ETH3D set, our method converges on average after 200 iterations, while using GS-SLAM's photometric loss, the algorithm settled after 330 iterations. This result explains the time gap between GS-SLAM (10s / image) and ReFineR (7s / image), and tends to prove that GOM is also a more efficient supervision metric for this registration task than photometric loss. Adding MatchAnything as a first stage provides more accurate initialization, which consequently reduces the number of required iterations. For ETH3D, the algorithm looped only 80 times on average, therefore reducing the total amount of time compared to ReFineR alone.

C.7 Point downsampling

In Table 13 we show a controlled robustness experiment on CARLA and ETH3D by randomly downsizing the point cloud (from 75% to 5%) to better characterize the density boundary of the method. We observe that starting from a 10-pixel perturbation, the final error monotonically decreases when the density increases, proving that ReFineR performs better with denser point clouds.

C.8 Impact of pixel-level misalignment on 3D semantic label transfer

To assess the practical importance of fine image-LiDAR registration, we conducted an additional experiment measuring how semantic labels transferred from a single segmented image to a dense point cloud degrade as the camera pose is increasingly perturbed. Starting from the reference pose, we introduced controlled perturbations on 100 cameras of our CARLA dataset corresponding to reprojection errors ranging from 0 to 20 pixels, projected the point cloud into the image, and assigned semantic labels only to geometrically visible points using occlusion estimation from a splatting rendering. We then evaluated the consistency of the resulting 3D labels with the reference assignment using mIoU on 20 semantic classes.

The results in Fig. 9 show that semantic transfer is highly sensitive to small pose errors. In particular, the average mIoU drops sharply from 1.0 at 0 pixels to about 0.4 at 5 pixels, and to about 0.33 at 10 pixels. This strong degradation is mainly caused by thin structures and object boundaries (such as poles, barriers, lamp posts, and contour regions), which are the first to be incorrectly reprojected under small pose perturbations. In contrast, the degradation becomes much slower beyond 10 pixels, with mIoU decreasing only marginally from about 0.33 to 0.27 between 10 and 20 pixels. This observation supports the importance of pixel-level image-LiDAR registration.

C.9 Qualitative registration experiment using UniCorrn

In preliminary experiments, we ran different inferences of the state-of-the-art image-to-point cloud registration model UniCorrn [18]. This model trains several layers of self- and cross-attention, embedding 2D and 3D features in a common space in order to generate dimensionality-agnostic descriptors and correspondences. In the 2D-3D case, the model was trained using the 7scenes [17] and RGB-D Scenes v2 [31] datasets, supplemented with synthetic depth maps from ScanNet++ [60] and ARKitScenes [2]. From Fig. 10, we suspect that the training data specialized the model for indoor depth images, which have particular geometry and density. Running the same model on general LiDAR point clouds may then represent a distribution shift for the model, which explains the weak results. We believe that this claim should be studied in more depth as it may represent a limitation for other similar methods [35, 55].

Set / Scene / Image	Official registration (px)	Manual registration (px)
Tanks and Temples		
Church		
21.png	5.13	1.37
40.png	4.35	1.89
83.png	7.27	1.99
110.png	4.4	1.28
190.png	4.0	1.68
281.png	7.6	1.24
332.png	6.44	1.07
358.png	7.66	1.37
408.png	5.11	1.19
438.png	364.66	1.31
Meetingroom		
2.png	7.24	1.83
28.png	8.1	1.37
43.png	7.88	1.31
75.png	7.47	1.55
117.png	7.01	1.49
171.png	5.09	2.15
183.png	10.04	2.16
281.png	7.48	1.15
301.png	11.34	1.55
338.png	9.21	1.44
nuScenes		
scene 1		
12.png	11.26	3.67
18.png	10.98	4.69
23.png	8.3	2.36
32.png	8.35	4.6
38.png	7.53	4.7
scene 12		
7.png	9.03	3.17
26.png	8.54	2.55
31.png	7.53	2.62
34.png	8.37	2.97
45.png	7.89	3.19
scene 3		
6.png	8.76	2.77
11.png	9.32	3.1
18.png	8.55	2.47
32.png	9.62	2.91
39.png	9.2	2.17
scene 7		
1.png	8.46	4.74
9.png	8.24	2.72
23.png	9.05	2.54
27.png	9.58	2.71
33.png	8.18	5.19
Livox		
0.png	N/A	3.6
1.png	N/A	3.84
2.png	N/A	3.56
3.png	N/A	3.65

Table 6: Reprojection error before and after manual registration of our images from Tanks and Temples, nuScenes, and Livox. Initial registration was obtained from the official dataset's ground truth for Tanks and Temples and using a state-of-the-art odometry algorithm for nuScenes.

	ETH3D		Tanks and Temples		CARLA		Livox		nuScenes		RGBDScenes	
	rot (°↓)	transl (cm↓)	rot (°↓)	transl (cm↓)	rot (°↓)	transl (cm↓)	rot (°↓)	transl (cm↓)	rot (°↓)	transl (cm↓)	rot (°↓)	transl (cm↓)
One stage												
GS-SLAM [📷]	8.19±2.30	57.08±37.98	8.05±1.32	101.69±69.42	9.00±1.23	333.86±101.20	7.58±1.15	71.95±38.52	6.92±2.08	180.34±37.02	7.93±2.57	22.35±5.49
Yuan *et al.* [📷]	0.75±0.92	7.99±22.47	0.30±0.47	6.12±11.04	0.79±1.51	199.68±497.18	0.46±0.12	6.79±6.46	1.17±2.76	90.93±179.32	4.92±4.11	12.78±20.35
Koide *et al.* [📷]	0.79±2.53	11.52±17.04	1.33±3.68	32.38±67.31	0.48±0.79	94.46±351.19	0.35±0.47	9.78±8.92	0.49±0.60	46.30±67.05	6.60±3.71	23.38±14.46
Taylor *et al.* [📷]	0.55±0.48	6.28±10.30	0.38±0.35	7.19±12.59	0.37±0.41	84.86±172.55	0.57±0.25	8.17±7.14	0.76±1.09	65.97±165.04	2.26±1.02	5.64±2.86
MatchAnything [📷]	0.13±0.11	1.32±1.62	0.12±0.08	2.78±3.75	0.15±0.17	37.95±53.56	0.28±0.11	4.84±4.99	0.15±0.11	14.93±20.84	1.13±0.55	2.48±1.28
ReFineR	0.10±0.36	1.02±4.69	0.08±0.22	1.24±4.97	0.08±0.25	4.44±7.52	0.33±0.13	6.86±5.93	0.17±0.32	5.63±9.76	2.20±1.00	6.16±2.58
Two stages												
MatchAnything + GS-SLAM [📷]	7.70±2.22	58.45±33.06	7.98±0.81	142.22±67.17	8.75±1.32	331.93±105.16	7.75±0.94	83.07±33.79	7.41±1.83	184.24±36.25	nan±nan	22.11±11.17
MatchAnything + Yuan *et al.* [📷]	0.77±4.87	7.45±16.26	0.26±0.29	6.79±8.19	0.58±1.31	117.48±455.80	0.34±0.20	2.77±6.56	1.23±3.20	94.80±582.17	5.08±3.94	17.73±16.64
MatchAnything + Koide *et al.* [📷]	0.39±1.33	4.40±12.41	1.21±3.30	25.68±81.56	0.40±1.52	87.48±258.70	0.39±0.14	10.80±6.77	0.34±1.65	33.79±186.61	5.30±4.23	26.55±19.88
MatchAnything + Taylor *et al.* [📷]	0.13±0.15	1.38±1.46	0.12±0.08	2.22±2.76	0.18±0.45	44.04±72.88	0.31±0.17	4.02±4.82	0.24±1.66	17.31±241.26	1.39±0.72	3.39±1.63
MatchAnything + MatchAnything [📷]	0.11±0.08	1.03±1.65	0.10±0.08	2.80±2.38	0.19±0.17	36.51±56.88	0.31±0.09	4.22±4.59	0.16±0.13	19.34±16.04	0.99±0.54	2.74±1.49
MatchAnything* + ReFineR	0.06±0.05	0.70±0.62	0.06±0.04	1.24±0.65	0.06±0.09	3.05±6.39	0.38±0.09	5.81±3.25	0.13±0.14	4.92±4.80	1.09±0.89	2.78±2.14

Table 7: Fine image to raw point cloud registration results (Section 5.2). Each image was registered using rendered point cloud normals.

	ETH3D		Tanks and Temples		Livox		nuScenes		RGBDScenes	
	rot (°↓)	transl (cm↓)	rot (°↓)	transl (cm↓)	rot (°↓)	transl (cm↓)	rot (°↓)	transl (cm↓)	rot (°↓)	transl (cm↓)
One stage										
GS-SLAM [📷]	0.47±1.87	4.31±20.80	0.10±1.36	1.62±67.28	2.57±2.03	74.66±39.92	4.37±2.66	131.13±57.91	2.81±2.22	17.61±7.13
Koide *et al.* [📷]	0.05±0.03	0.38±0.56	0.06±0.04	1.82±0.73	0.26±0.09	8.13±3.55	0.48±0.53	30.81±61.62	0.77±3.38	2.06±11.74
Taylor *et al.* [📷]	0.55±0.53	6.03±10.20	0.33±0.43	6.30±14.72	0.93±0.29	9.58±7.64	0.92±1.74	50.40±183.95	2.13±0.82	5.92±2.40
MatchAnything [📷]	0.07±0.10	0.66±0.79	0.07±0.05	1.64±1.52	0.26±0.15	3.14±5.77	0.11±0.06	11.24±10.67	0.40±0.24	0.85±0.64
ReFineR	0.08±0.50	0.57±7.09	0.07±0.10	1.20±1.56	0.28±0.16	4.83±6.31	0.23±0.28	7.43±9.85	2.12±1.19	5.77±2.96
Two stages										
MatchAnything + GS-SLAM [📷]	0.13±2.00	1.15±18.81	0.08±1.74	1.51±24.68	3.00±2.09	82.11±35.87	4.91±3.09	141.23±68.28	2.73±3.67	16.21±8.91
MatchAnything + Koide *et al.* [📷]	0.05±0.03	0.37±0.54	0.06±0.04	1.76±0.75	0.25±0.09	10.38±4.60	0.34±0.52	32.70±96.20	0.68±3.56	1.75±15.71
MatchAnything + Taylor *et al.* [📷]	0.07±0.11	0.66±0.85	0.08±0.06	1.63±1.37	0.28±0.14	2.52±6.02	0.16±1.00	18.14±55.96	0.45±0.37	1.38±1.06
MatchAnything + MatchAnything [📷]	0.05±0.05	0.49±0.66	0.07±0.05	1.51±1.03	0.24±0.16	3.85±5.85	0.12±0.06	10.63±9.80	0.35±0.16	0.88±0.46
MatchAnything* + ReFineR	0.04±0.04	0.40±0.60	0.06±0.04	1.17±0.65	0.22±0.16	3.55±6.21	0.11±0.08	3.88±3.02	0.24±0.13	0.50±0.62

Table 8: Fine image to intensity point cloud registration results (Section 5.3). Each image was registered using rendered point cloud reflectivity or grayscale value.

	ETH3D				Tanks and Temples				CARLA				Livox			nuScenes				RGBDScenes			
	pix@ (%↑)			p_e(px↓)	pix@ (%↑)				p_e(px↓)	pix@ (%↑)			p_e(px↓)	pix@ (%↑)		p_e(px↓)	pix@ (%↑)			p_e(px↓)	pix@ (%↑)			p_e(px↓)
	1	2	5		1	2	3	5		1	2	5		5	10		3	5	10		3	5	10	
(VII)																								
Rendered depth	39	93	100	1.04±0.55	0	58	92	100	1.95±0.76	8	36	78	2.45±7.23	25	95	6.48±2.32	3	36	92	5.66±8.57	6	22	80	6.61±4.58
Monocular normals	0	2	9	8.34±2.26	0	28	90	100	2.43±0.58	2	23	71	3.29±6.49	65	100	4.94±0.27	0	12	71	7.15±13.75	18	57	94	4.47±2.83
Monocular depth	0	0	24	7.20±3.17	0	5	32	83	3.62±2.67	2	7	41	5.76±59.08	5	60	9.77±2.87	0	2	37	11.88±34.59	5	61	90	4.59±3.03
MatchAnything* + ReFineR (RGB image)	58	96	100	0.90±0.40	0	70	95	100	1.76±0.81	16	68	91	1.54±3.04	55	100	4.94±0.94	6	54	97	4.90±2.75	13	43	92	5.36±4.25

Table 9: Ablation study on ReFineR's supervision channel (Section C.4).

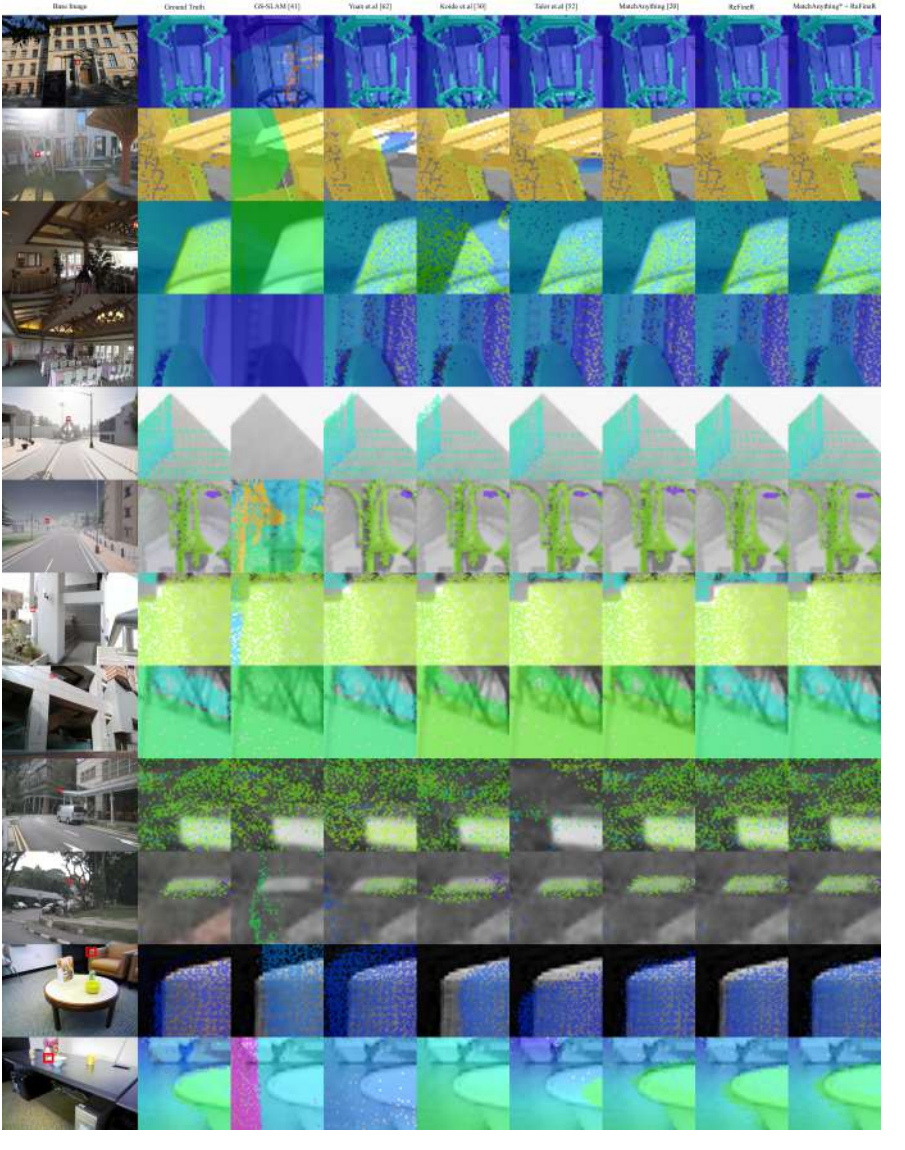


Figure 7: Qualitative results from experiment Section 5.2.

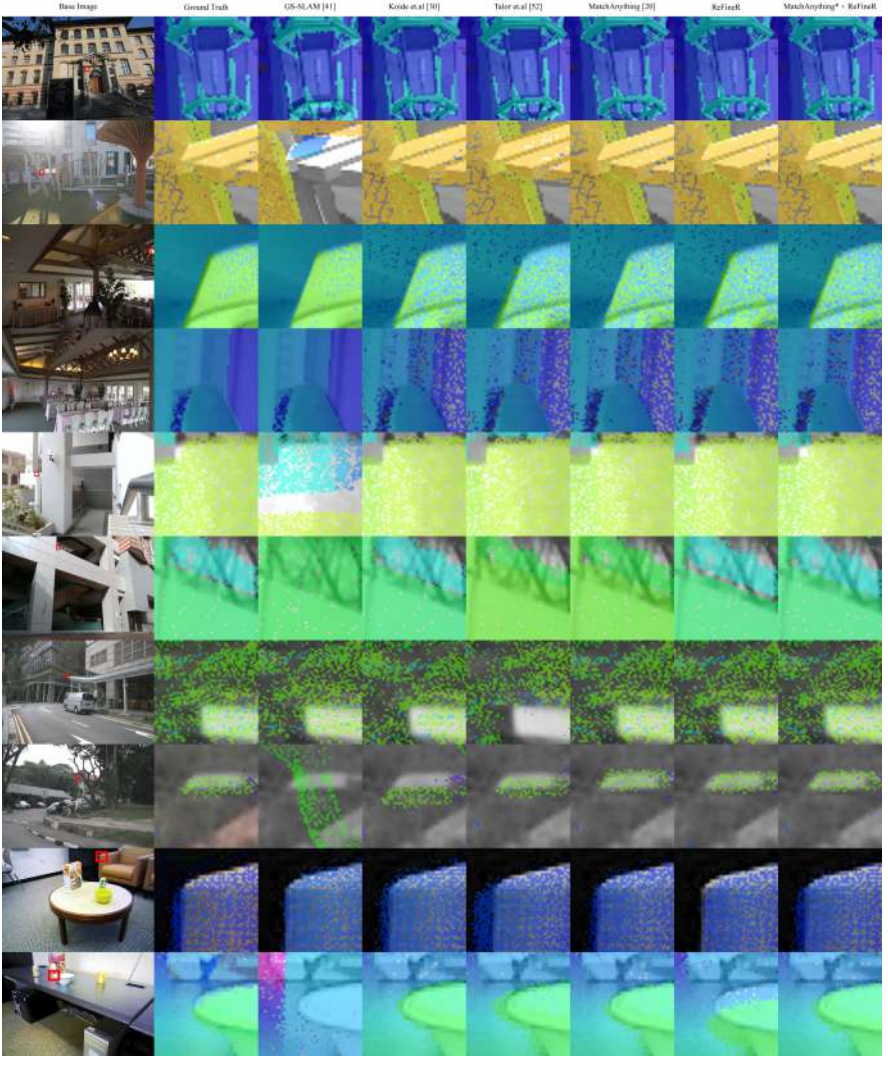


Figure 8: Qualitative results from experiment Section 5.3.

	ETH3D										
	pix@ (%↑)								p_e(px↓)	rot (°↓)	transl (cm↓)
	1	2	3	5	10	20	50	100			
0px initialization											
MatchAnything	14	67	98	100	100	100	100	100	$1.66_{\pm 0.66}$	$0.10_{\pm 0.05}$	$0.94_{\pm 1.55}$
ReFineR	64	100	100	100	100	100	100	100	$0.89_{\pm 0.35}$	$0.06_{\pm 0.04}$	$0.70_{\pm 0.62}$
MatchAnything* + ReFineR	65	97	100	100	100	100	100	100	$0.87_{\pm 0.38}$	$0.06_{\pm 0.05}$	$0.68_{\pm 0.74}$
5px initialization											
MatchAnything	8	62	91	100	100	100	100	100	$1.72_{\pm 0.84}$	$0.13_{\pm 0.09}$	$1.20_{\pm 1.72}$
ReFineR	52	92	96	98	100	100	100	100	$0.95_{\pm 0.86}$	$0.07_{\pm 0.12}$	$0.75_{\pm 1.95}$
MatchAnything* + ReFineR	58	97	100	100	100	100	100	100	$0.87_{\pm 0.38}$	$0.06_{\pm 0.05}$	$0.69_{\pm 0.66}$
10px initialization											
MatchAnything	8	65	90	100	100	100	100	100	$1.80_{\pm 0.83}$	$0.13_{\pm 0.11}$	$1.32_{\pm 1.62}$
ReFineR	41	73	78	79	96	100	100	100	$1.10_{\pm 3.17}$	$0.10_{\pm 0.36}$	$1.02_{\pm 4.69}$
MatchAnything* + ReFineR	58	96	100	100	100	100	100	100	$0.90_{\pm 0.40}$	$0.06_{\pm 0.05}$	$0.70_{\pm 0.62}$
20px initialization											
MatchAnything	7	52	87	96	100	100	100	100	$1.92_{\pm 1.16}$	$0.15_{\pm 0.11}$	$1.28_{\pm 1.82}$
ReFineR	20	35	39	41	48	87	100	100	$10.74_{\pm 8.11}$	$0.79_{\pm 1.10}$	$10.21_{\pm 14.01}$
MatchAnything* + ReFineR	60	96	100	100	100	100	100	100	$0.89_{\pm 0.39}$	$0.06_{\pm 0.05}$	$0.70_{\pm 0.67}$
50px initialization											
MatchAnything	10	54	77	96	97	100	100	100	$1.96_{\pm 2.17}$	$0.13_{\pm 0.15}$	$1.43_{\pm 1.85}$
ReFineR	1	1	1	1	1	2	60	100	$47.85_{\pm 9.06}$	$4.90_{\pm 2.23}$	$41.26_{\pm 45.44}$
MatchAnything* + ReFineR	58	97	100	100	100	100	100	100	$0.90_{\pm 0.40}$	$0.06_{\pm 0.05}$	$0.70_{\pm 0.65}$
100px initialization											
MatchAnything	3	39	71	88	97	100	100	100	$2.14_{\pm 2.37}$	$0.14_{\pm 0.25}$	$1.94_{\pm 2.47}$
ReFineR	0	0	0	0	0	0	0	36	$107.20_{\pm 17.05}$	$10.17_{\pm 4.36}$	$84.29_{\pm 92.17}$
MatchAnything* + ReFineR	60	97	100	100	100	100	100	100	$0.87_{\pm 0.41}$	$0.06_{\pm 0.05}$	$0.70_{\pm 0.70}$

Table 10: Registration at scale experiment, each camera pose was initialized at increasing pixel errors (Section C.5).

	CARLA (normals)	ETH3D (intensity)
	Time (s)	Time (s)
GS-SLAM [41]	36.13	9.68
Render	11.31	3.48
Loss	0.12	0.62
Backward	22.24	4.62
Yuan *et al.* [52]	47.03	N/A
Koide *et al.* [30]	10.89	26.80
Taylor *et al.* [53]	80.01	87.79
MatchAnything [7]	4.84	4.90
Weight Loading	3.12	3.11
Inference	1.69	1.75
RANSAC	0.02	0.03
ReFineR	11.62	6.83
Render	3.69	2.89
Loss	0.08	0.10
Backward	7.31	3.14
MatchAnything* + ReFineR	13.81	6.97

Table 11: Detailed computation times for the benchmarking methods on the CARLA and ETH3D sets.

	time / iter (s)	Num Points (M)
ETH3D	$0.03_{\pm 0.02}$	$3.21_{\pm 2.82}$
Tanks and Temples	$0.14_{\pm 0.07}$	$19.92_{\pm 11.40}$
CARLA	$0.08_{\pm 0.05}$	$3.74_{\pm 2.72}$
Livox	$0.07_{\pm 0.02}$	$5.01_{\pm 1.68}$
nuScenes	$0.02_{\pm 0.00}$	$1.02_{\pm 0.35}$
RGBDScenes	$0.03_{\pm 0.01}$	$0.65_{\pm 0.17}$

Table 12: Average time per ReFineR iteration and average number of rendered points for each set.

	ETH3D							CARLA						
	pix@ (%↑)					p_e(px↓)	time (s)	pix@ (%↑)					p_e(px↓)	time (s)
	1	2	3	5	10			1	2	3	5	10		
ReFineR (s=100%)	41	73	78	79	96	$1.10_{\pm 3.17}$	$6.83_{\pm 5.03}$	12	44	57	74	92	$2.36_{\pm 3.45}$	$11.62_{\pm 7.98}$
ReFineR (s=75%)	31	64	68	73	96	$1.35_{\pm 3.26}$	$6.73_{\pm 4.15}$	2	25	48	60	92	$3.58_{\pm 3.36}$	$12.38_{\pm 10.85}$
ReFineR (s=50%)	29	58	61	69	92	$1.45_{\pm 3.52}$	$5.35_{\pm 3.09}$	2	22	36	54	88	$4.27_{\pm 4.01}$	$9.63_{\pm 8.06}$
ReFineR (s=25%)	19	43	46	51	92	$4.20_{\pm 3.71}$	$4.00_{\pm 2.17}$	0	8	29	48	80	$5.17_{\pm 3.92}$	$5.97_{\pm 4.05}$
ReFineR (s=10%)	8	34	42	50	90	$4.94_{\pm 3.75}$	$3.18_{\pm 1.57}$	0	0	9	26	71	$7.52_{\pm 3.66}$	$3.77_{\pm 1.98}$
ReFineR (s=5%)	4	14	20	24	78	$8.05_{\pm 3.60}$	$2.63_{\pm 1.52}$	0	0	0	15	69	$9.23_{\pm 3.06}$	$2.86_{\pm 1.56}$

Table 13: Downsampling experiment with raw point clouds. Each registration was initially perturbed by 10px and carried out with random downsampling of the point cloud by a varying amount.

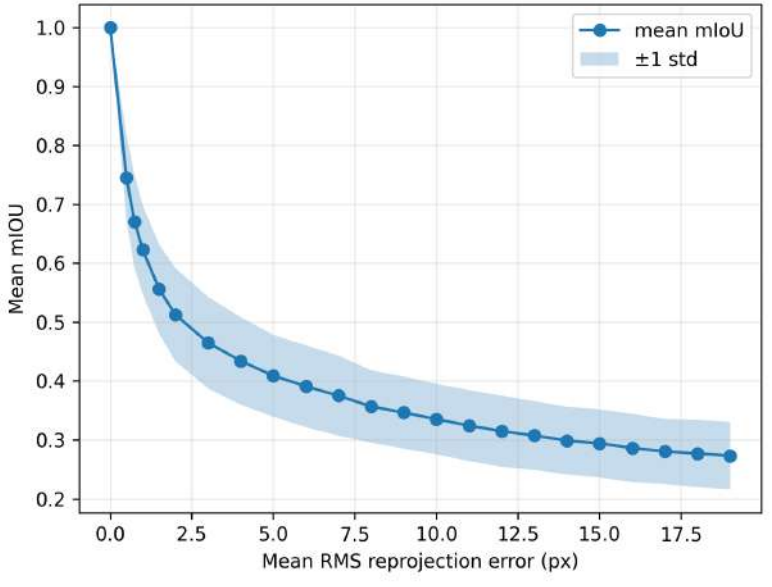


Figure 9: Average impact of camera pose error on projected 3D semantic labels.

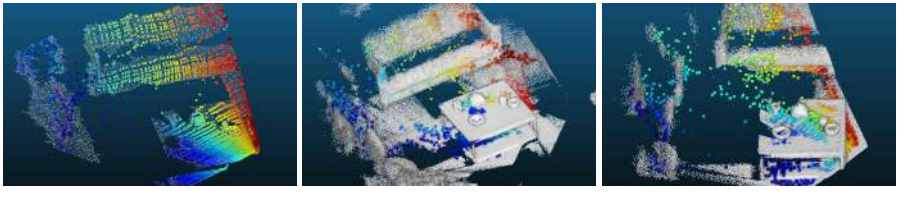


Figure 10: Left to right: UniCorrn inference on a single RGBD frame (accurate predictions), on densely aggregated scans, and on synthetic depth from the dense point cloud (large errors).