Point Cloud Mapping for Pose Estimation of Uncooperative Satellites

Sidharth Anantha
Dept. of AA
Stanford University
anathas@stanford.edu

Sammie Staudinger
Dept. of ME
Stanford University
sammiest@stanford.edu

Andrew Pasco
Dept. of ME
Stanford University
apasco@stanford.edu

1 Abstract

In this paper, we investigate three approaches for mapping point clouds to estimate the pose of uncooperative satellites: a modified iterative closest point (ICP) algorithm, Deep Closest Point (DCP), and a neural network trained with MSE and Chamfer distance losses. By using 3D Gaussian splatting and FisherRF uncertainty metric, the goal is to leverage this uncertainty during point cloud alignment to improve robustness under noisy or dynamic conditions, especially common in space environments. Our results show that classical ICP with FPFH initialization performs well, while uncertainty-weighted ICP, DCP, and neural methods face challenges under scale mismatch, indexing variability, and degeneracy.

2 Introduction

As space becomes more crowded, understanding how to characterize unknown satellites or space junk by learning their pose and shape, is an important problem [1] [2] [3]. The mission con-ops involves an observing satellite rendezvousing with a space object, taking images to build and refine a 3D Gaussian Splat (3DGS) model, which adaptively learns the space object's shape. Embedded in this 3DGS model is a metric called Fisher RF [4], which effectively gives the uncertainty of a given point. The process of learning an unknown target's shape is actively being studied in the Space Rendezvous Lab (SLAB) [5], and this project is an extension to learn the target's pose as well. Through a known 3DGS model, the pose at any given time can be estimated, however if the target were to apply control torques or be affected by perturbations that are unaccounted for, the estimate of the target's pose would be inaccurate. To account for this, the estimate can be compared to a ground truth (GT) image taken by the observer spacecraft, and the transformation between the two models yields an update to the estimated attitude. To make the problem tractable, a point cloud discretization of the 3DGS model and GT image can be compared. The 3DGS model has point-wise uncertainty metric FisherRF [4], which is useful for space applications, which suffer from noisy images and dynamic environments.
Research Objective: Thus the innovation we propose is embedding the FisherRF uncertainty metric into the 3DGS point cloud, and allowing for a *malleable* point cloud mapping that more strongly enforces mappings between points with greater uncertainty. This learned mapping between the ground truth and estimate point clouds will be used to update the estimate of the target's pose (Figure 1).

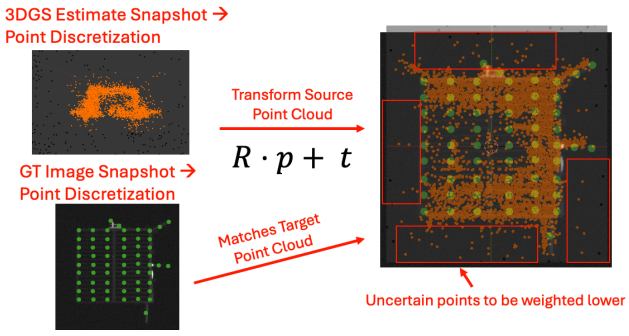


Figure 1: Proposed Conops of This Work

3 Related Works

Past examples from the literature like DCP [6], PRNet [7], and 3DRegNet [8] developed deep learning approaches for generating 3D registrations between point clouds. DCP's approach creates a "feature-per-point" embedding from features of both clouds, and the uncertainty metric could enhance the "soft-matching" step. PRNet builds upon this with an iterative approach alternating between keypoint detection, correspondence via a Gumbel-Softmax mapping with "adjustably-sharp" matching based

on a parameter predicted each iteration by a small auxiliary network, and transformation prediction. In this case, the auxiliary network could be given information about uncertainty near particular keypoints, which could help improve how this "temperature" parameter is learned. Contrastingly, 3DRegNet assumes given correspondences and classifies them as inliers or outliers before predicting the transformation. In this case, the uncertainty metric associated with one of the points in a correspondence pair could be used as an additional input to the classification block.

4 Dataset

4.1 Spacecraft Renderer

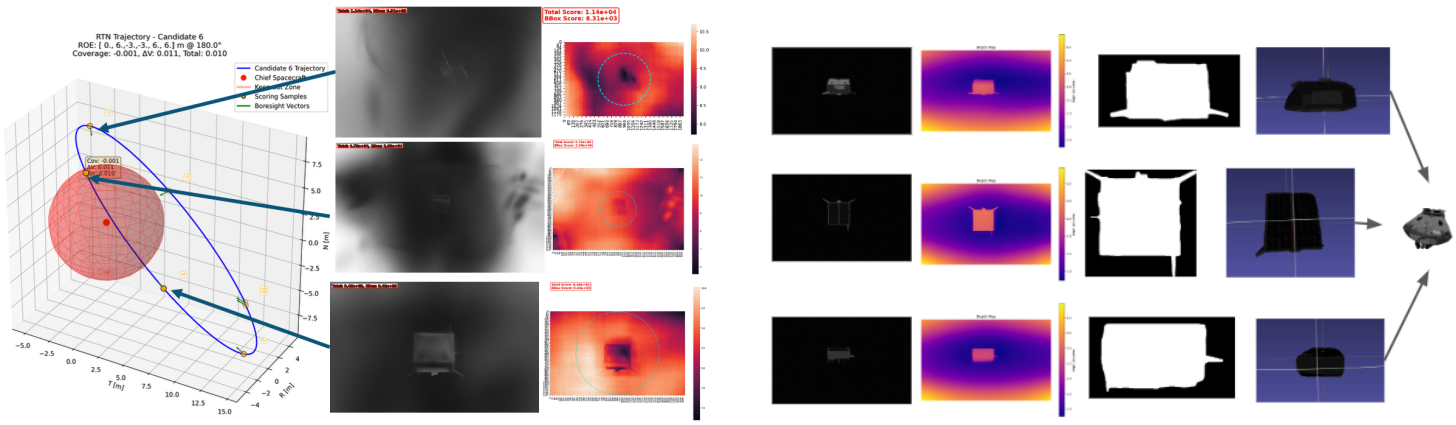
As part of the SLAB research group, an in-house renderer is made that can render a satellite in a space environment with proper background (stars, earth) and lighting conditions, shown as input examples in Figure 2b. This renderer can be queried by the camera pose, which consists of a position vector $\mathbf{r}$ which is the position of the target spacecraft in the camera's reference frame, and a rotation quaternion $\mathbf{q}$ which is the rotation from the camera's axes to the spacecraft or world axes. This relative orbit and attitude information is known from our own orbit and attitude propagator and is used to query rendered images at each timestep. This dataset will be used to both generate the ground truth image but also train the 3DGS model.

4.2 FisherRF Uncertainty Metric

In 3DGS, a model is comprised by a set of thousands of *gaussians* [9], which are 3D probability distributions, which for gaussian i is defined by the parameter set $\theta_i = [\mu_x, \mu_y, \mu_z, \sigma_x, \sigma_y, \sigma_z, \alpha, c_0, ..., c_k]$, with position μ, volume σ, opacity α and color spherical harmonics $\mathbf{c}$. Each of these parameters have an impact on the model's rendered image, and this can be modeled by the 2nd derivative of the rendered image $f(\mathbf{x}; \mathbf{w}^*)$, where $\mathbf{x}$ is the camera pose of the rendered image, and $\mathbf{w}^*$ is the current parameter set of the model. Mathematically the FisherRF score is given by the hessian of the model: $\mathbf{H}^{''}[\mathbf{y}|\mathbf{x}, \mathbf{w}^*] = \nabla_{\mathbf{w}} f(\mathbf{x}; \mathbf{w}^*)^T \nabla_{\mathbf{w}} f(\mathbf{x}; \mathbf{w}^*)$. The score can be computed per gaussian, and when rasterizing pixels or points the gaussian's uncertainty is summed into the point.

4.3 3DGS Point Cloud

Using a set of 200 rendered images, a 3DGS model can be trained. Using known relative orbital and attitude properties, a random camera pose can be queried via the same camera inputs $(\mathbf{r}, \mathbf{q})$ to render that view, in addition to the uncertainty score from that view, as seen in Figure 2a. Because we desire to work with point clouds, rather than rendering an image, we can generate a point cloud from this view, as detailed in Figure 1, and assign a FisherRF uncertainty weight to each point. Thus when performing point cloud mapping, each point is weighted in proportion to how certain its geometry is.



(a) On-Orbit Rendered 3DGS Images and Respective FisherRF Uncertainty Maps and Score

(b) Image to Pointcloud Process with Three Perspectives Shown

Figure 2: Comparison of rendered 3DGS uncertainty maps (left) and image to pointcloud outputs (right).

4.4 2D Image Point Cloud

The ground truth input is an RGB image of the target satellite, and is processed into a point cloud using a monocular depth estimation model. The target point cloud is obtained from the depth image output

of the pre-trained DepthAnythingV2 model [10], which provides per-pixel depth values representing the relative distance of each surface point from the camera. In the second step, each depth map is pre-processed to remove background noise. Assuming the satellite is centered against a black space background, the object is isolated by smoothing the depth map with a bilateral filter and comparing all pixel depths to the central pixel depth, selecting the largest connected region as the main object and producing a mask. Each cleaned depth map is then converted into a 3D point cloud using the camera's intrinsic parameters and extrinsic poses, transforming image coordinates into 3D space to yield one partial point cloud per viewpoint. An example of the results of this process is found in Fig. 2b.

5 Methodology

5.1 Pipeline

As detailed in Figure 1, an estimate of pose is determined from the 3DGS point cloud render, and the ground truth is a 2D image discretized into a 3D point cloud. Using the two point clouds, the rigid body transform can be found that represents the error in the pose estimate.

5.2 Modified Iterative Closest Point

ICP (with an adapted implementation from the `trimesh.registration`[11] library) was utilized to align the source (with weightings based on FisherRF output) to the target. By default, ICP weights all points equally. The algorithm alternates between a step creating correspondences from all points in the source (A) to their closest match in the target (B), then determining the best fit transform by $\min_{(\tau_{BA}, R_{BA})} ||P_B - (R_{BA}P_A + \tau_{BA})||_{\mathcal{F}}^2$, solved via Singular Value Decomposition (SVD). Including point-wise weightings ($\hat{w}$) from FisherRF, higher-confidence points will contribute more to the computation of the centroid ($\bar{A} = \sum_i a_i \hat{w}_i$) and impact the construction of the weighted cross-covariance matrix for SVD as $[(B - \bar{B})]^T[\hat{w} * (A - \bar{A})]$. ICP is also often initialized with a coarse feature-based correspondence; here Fast Point Feature Histograms [12] were used for correspondence, and RANSAC was used to obtain an initial transform.

5.3 Neural Network

In terms of a basic neural network, we learn the soft correspondence between two point clouds. In other words, for each point in the source point cloud, a probability distribution over points for the target point cloud is produced, thus creating a virtual point that is the soft weighted average of the target point cloud. Knowing these correspondences makes determining the rotation and translation a trivial problem through SVD. To find the correspondences, the first step is to use a Multi-Layer Perceptron (MLP) to encode each point from the source and target set into a higher dimension [13], via $f_i = \phi(p_i) \in \mathbf{R}^D$. The weights of this neural network must be learned by training on existing data, and two different loss metrics can be used. The first is Mean Squared Error (MSE) loss, where the neural network predicts $\hat{Y}_i$ for each source point X_i such that $\mathcal{L}_{\text{MSE}} = \frac{1}{N}\sum_{i=1}^{N} \left\| \hat{Y}_i - Y_i \right\|_2^2$. Intuitively, MSE loss measures how close each predicted point is to its target point at the same index. Critically, this assumes that the two point clouds have the same number of points and the indices of the clouds are the same between each case. The second metric is chamfer distance loss, which compares the shapes geometrically, specifically from two point sets $\mathcal{L}_{\text{Chamfer}}(X, Y) = \sum_{x \in X} \min_{y \in Y} \|x - y\|_2^2 \; + \; \sum_{y \in Y} \min_{x \in X} \|x - y\|_2^2$. Intuitively, for each point in one cloud, chamfer distance loss finds the closest point to the other cloud in terms of shape similarity. Notably this can be used for point clouds of different sizes.

Once the MLP is trained, all points in a point cloud can be transformed, namely $F_{src} = [f_1^{src}, f_2^{src}, \cdots, f_N^{src}]^T \in \mathbf{R}^{N \times D}$ and $F_{tgt} = [f_1^{tgt}, f_2^{tgt}, \cdots, f_N^{tgt}]^T \in \mathbf{R}^{M \times D}$. The structural similarity relating the two point clouds is given by $C_{N \times M} = F_{src,N \times D} \cdot F_{tgt,D \times M}^T$, and then softmax can be used to compute the mapping. In other words, for a given point p_i, what is the probability that it matches with point q_i: $S_{i,j} = \frac{\exp(C_{i,j})}{\sum_{k=1}^{m} \exp(C_{i,k})}$. Knowing this probability distribution a soft correspondence, or virtual point, can be learned that relates point p_i to a weighted average of points in q_i: $\tilde{q}_i = \sum_{j=1}^{M} S_{i,j} \cdot q_j$. Using the source cloud $\mathbf{P}$ and virtual target cloud $\tilde{\mathbf{Q}}$, the linear transform can be made, as shown in Eqn. 1. Here, the source X is transformed by the rigid transformation W, a combination of rotation $\mathbf{a}$ and translation $\mathbf{b}$, and yields the virtual point cloud Y. The matrix W can be solved via SVD.

$$\underbrace{\begin{bmatrix} \tilde{q}_x^{(1)} & \tilde{q}_y^{(1)} & \tilde{q}_z^{(1)} \\ \tilde{q}_x^{(2)} & \tilde{q}_y^{(2)} & \tilde{q}_z^{(2)} \\ \vdots & \vdots & \vdots \\ \tilde{q}_x^{(N)} & \tilde{q}_y^{(N)} & \tilde{q}_z^{(N)} \end{bmatrix}}_{\text{Y (N×3)}} = \underbrace{\begin{bmatrix} p_x^{(1)} & p_y^{(1)} & p_z^{(1)} & 1 \\ p_x^{(2)} & p_y^{(2)} & p_z^{(2)} & 1 \\ \vdots & \vdots & \vdots & \vdots \\ p_x^{(N)} & p_y^{(N)} & p_z^{(N)} & 1 \end{bmatrix}}_{\text{X (N×4)}} \underbrace{\begin{bmatrix} \begin{bmatrix} a_{xx} & a_{xy} & a_{xz} \\ a_{yx} & a_{yy} & a_{yz} \\ a_{zx} & a_{zy} & a_{zz} \end{bmatrix} \\ \begin{bmatrix} b_x & b_y & b_z \end{bmatrix} \end{bmatrix}}_{\text{W (4×3)}} \tag{1}$$

6 Results and Discussion

6.1 Depth Anything

The DepthAnything-based dataset cannot be aligned with the 3DGS point clouds using DCP due to a fundamental geometric/scale mismatch. DepthAnything produces relative, non-metric depth and does not incorporate the true camera intrinsics, and thus we believe these limitations are inherent to the monocular depth model and lead to the observed failure of DCP on this dataset. Attempts were made to take into account the scaling mismatch within the DCP framework by modeling the relationship between the point clouds as $Y \approx s\,A(X) + t + \varepsilon$, where s is an unknown global scale factor, A is transformed source point, and ε denotes noise as proposed in [14]. Unsurprisingly due to the harsh lighting conditions of an outer-space environment, 2D image to 3D point cloud cannot be used as a ground truth. Moving forward, the internal estimate and ground truth will both be treated as a rendered point cloud from the 3DGS model.

6.2 Neural Network

6.2.1 MSE Loss Metric

MSE loss is the simplest metric to use, however it requires that the point clouds have the same number of points, and the ordering of the point cloud is the same between different input cases. We have a simple test case that involves our observer satellite hovering close and far away from a stationary chief satellite, in such a case that the point cloud size and ordering is consistent between different viewpoints. The neural network's loss converges quickly during training, and because the train and test datasets are closely related there is low variance. Viewing Figure 5 the accuracy is limited to 40% and does not converge quickly. Inspecting a test case shown in Figure 3, most points in the virtual cloud (green) converge to the target cloud (red), however the noisy points reduce accuracy. The computed rigid body transformation from the output is in line with the expected transformation

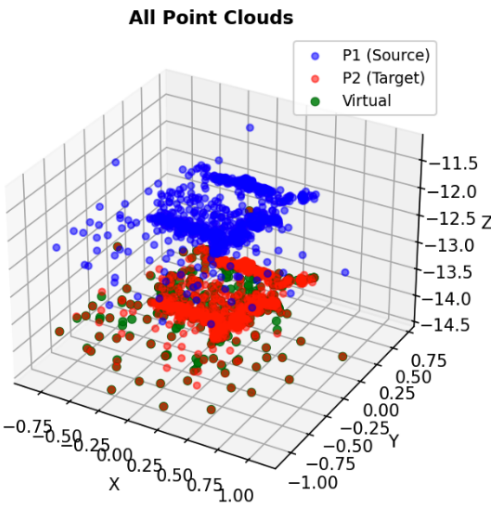


Figure 3: Test Case on Trained MLP Using MSE Loss

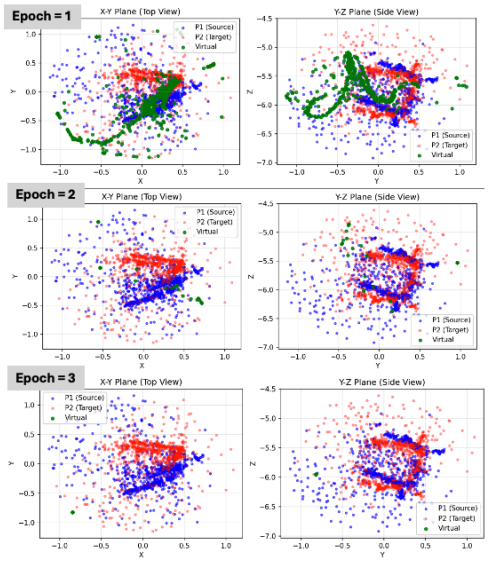


Figure 4: Visualizing Training Iterations on MLP Using Chamfer Loss

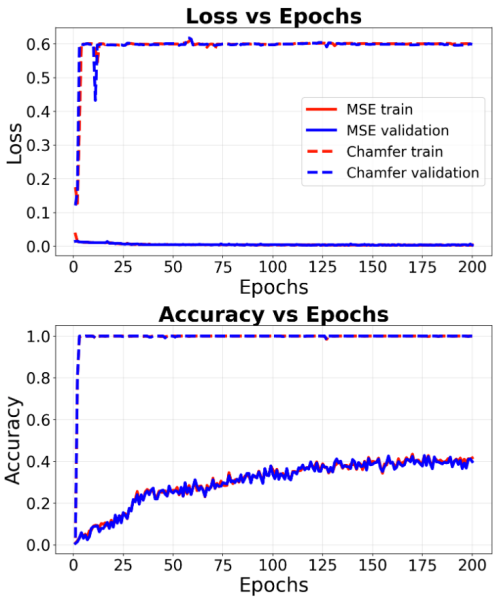


Figure 5: Accuracy and Loss Metrics for MSE and Chamfer

computed through an orbital and attitude simulation. In reality, the point cloud size and ordering changes with each input, so using MSE loss is not generalizable.

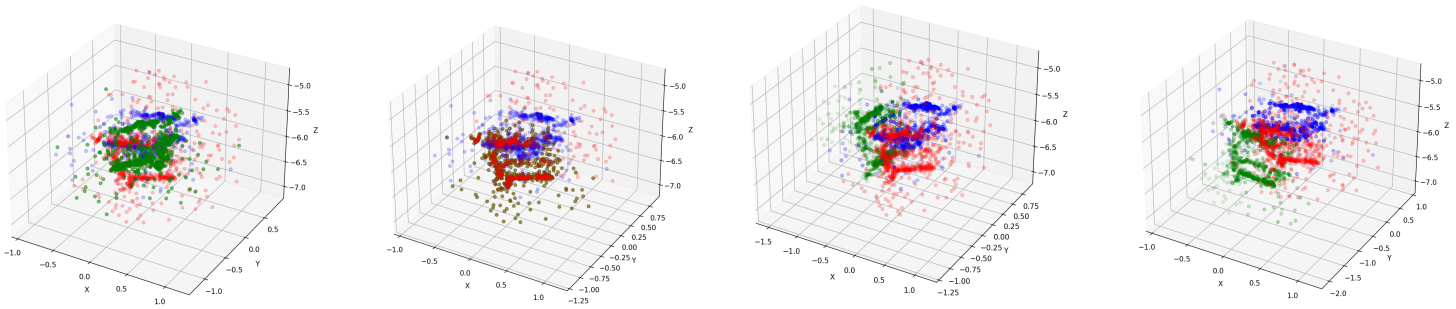
6.2.2 Chamfer Distance Loss

Chamfer distance loss is an alternative loss metric that is geometry-based rather than index-based. This enables point clouds of different sizes and orderings to be used, however the chamfer loss criteria does not guarantee that each point $q_i \in \mathbf{Q}$ will map to a unique point $p_i \in \mathbf{P}$. This results in a failure mode where each point $p_i \in \mathbf{P}$ maps to the *same* point $q_i \in \mathbf{Q}$. This is illustrated in Figure 4, where within the first 3 epochs a large point cloud degenerates into a single point. Adding regularization to prevent points from being too close together can attempt to remedy this issue, however after many attempts we could not prevent the ultimate degeneracy problem. This shows that the baseline of using a neural network to learn the correspondences is not easily feasible, so a more robust approach using ICP will be used.

6.3 ICP

Experiments were performed with base ICP (with/without FPFH initialization) and uncertainty-weighted ICP (with/without FPFH initialization). Qualitatively, FPFH+RANSAC provided an excellent coarse alignment, from which unweighted ICP converged in 2-3 iterations. In contrast, even with a similar initialization, the weighted approach diverged from the desired alignment until the stop condition was reached.

The main hypothesis for the cause of this divergence was the presence of severe outlier weights which dominated the computation of the best-fit transform. Presence of these outliers was confirmed via summary statistics, with means on the order of 10^3 and maximum values on the order of 10^5 or 10^6. A non-linear (sqrt, then $\log$) transform was implemented in an attempt to rectify this. However, results (Figure 6) were still poor for all implementations except FPFH+base ICP, which refined slightly from the high-quality initialization.



(a) No initialization, base ICP. (b) FPFH initialization, base ICP. (c) No initialization, weighted ICP. (d) FPFH initialization, weighted ICP.

Figure 6: Visual examples of method failures. Only FPFH+base ICP visually converges. Blue is source, red is target, and green is transformed source.

Once the source cloud is aligned with the target, as in (b), the resulting transformation matrix represents the relative pose between the two representations of the satellite. Compared against the ground truth, we verify that only method (b) produces satisfactory output.

7 Conclusion and Future Work

Our study found that the standard ICP method is robust and consistently delivers good results for point cloud alignment. Integrating FisherRF uncertainty weights does not yield consistent solutions, mostly due to the large range in uncertainty between noisy points and certain points. This results in such a strong imbalance in weighting between points that over-skews the point cloud toward a cluster of certain points. Future work will involve pruning low uncertainty points to further normalize the score to an acceptable range. The standard neural network with chamfer distance resulted in point degeneracy, however more literature review and experimentation may yield a regularization method that can regulate unique mappings between source and target point clouds. Lastly a more accurate sensor input can be used for the ground truth. While using 2D images for depth is unreliable, using another sensor such as LiDAR will provide a more robust ground truth estimate point cloud.

8 Contributions

Code for this project can be found here.

- Sidharth: Designed the 3DGS architecture, algorithm and point cloud conversion. Also wrote the neural network implementation
- Sammie: DepthEverything Dataset generation pipeline/testing/debugging, DCP implementation/debugging; Metrics eval
- Andrew: ICP & weighted ICP implementation/testing, dataset/dataloader implementation for NN

References

[1] Y. K. Nakka, W. Hönig, C. Choi, A. Harvard, A. Rahmani, and S.-J. Chung. Information-based guidance and control architecture for multi-spacecraft on-orbit inspection. *Journal of Guidance, Control, and Dynamics*, 45(7):1184–1201, 2022.

[2] M. Sabatini, R. Volpe, and G. B. Palmerini. Centralized visual-based navigation and control of a swarm of satellites for on-orbit servicing. *Acta Astronautica*, 171:323–334, 2020.

[3] B. Bernhard, C. Choi, A. Rahmani, S.-J. Chung, and F. Hadaegh. Coordinated motion planning for on-orbit satellite inspection using a swarm of small-spacecraft. In *Proceedings of the IEEE Aerospace Conference*, pages 1–13. IEEE, 2020.

[4] Wen Jiang, Boshu Lei, and Kostas Daniilidis. FisherRF: Active view selection and uncertainty quantification for radiance fields using fisher information. Version Number: 2.

[5] P. Huc and E. Bates. Sq3dgs: Real-time 3d gaussian splatting for autonomous satellite inspection. Summer 2025 technical report, Stanford Space Rendezvous Lab, 2025.

[6] Yue Wang and Justin M. Solomon. Deep closest point: Learning representations for point cloud registration. *arXiv preprint arXiv:1905.03304*, 2019.

[7] Yue Wang and Justin M. Solomon. Prnet: Self-supervised learning for partial-to-partial registration. *arXiv preprint arXiv:1910.12240*, 2019.

[8] Gonçalo Dias Pais, Srikumar Ramalingam, Venu Madhav Govindu, João C. Nascimento, Rama Chellappa, and Pedro Miraldo. 3dregnet: A deep neural network for 3d point registration. *arXiv preprint arXiv:1904.01701*, 2019.

[9] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. `https://arxiv.org/abs/2308.04079`, 2023. arXiv preprint arXiv:2308.04079, v1.

[10] Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything: Unleashing the power of large-scale unlabeled data. In *CVPR*, 2024.

[11] Dawson-Haggerty et al. trimesh.

[12] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. In *2009 IEEE International Conference on Robotics and Automation*, pages 3212–3217, 2009.

[13] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. Version Number: 1.

[14] Shinji Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(4):376–380, 1991.