ShadowCorr: Cross-View Correspondence of 3D Segments via Volumetric Consensus

Yiyan Ruan and Erik Komendera

***Abstract*—Associating per-view 3D object segments that originate from the same rock instance—segment-level correspondence across viewpoints—is challenging when partial visibility produces non-overlapping fragments or when physically adjacent rocks create ambiguous boundaries. Given a set of per-view 3D segments produced by 2D instance segmentation back-projected through depth, together with calibrated camera poses, this paper introduces ShadowCorr, a voxel-based framework that infers segment correspondences through volumetric consensus rather than direct surface matching. Each segment is extended backward into the space its underlying rock must occupy, using an estimated thickness, generating confidence-weighted occupancy voxels within a shared 3D grid. Segments belonging to the same rock instance then produce overlapping volumes even when their visible surfaces do not, and a sparse 3D convolutional neural network reads these co-occurrence patterns into discriminative per-voxel embeddings. Clustering then yields the final segment-level correspondences. Evaluated on 200 synthetic lunar terrain test scenes containing 18–20 rock instances per scene, ShadowCorr achieves 99.04% rock purity and 98.88% cluster purity, for an average of 98.96%, exceeding the best baseline by 9.64 points (89.32%).**

***Index Terms*—Sparse convolutional networks, cross-view correspondence, 3D segmentation, multi-view geometry, planetary robotics, point cloud processing, volumetric scene reasoning**

I. Introduction

ESTABLISHING correspondences between *segments*, partial 3D object observations produced by per-view instance segmentation and back-projection through depth, across multiple viewpoints is a fundamental challenge in robotics and computer vision. Unlike classical correspondence pipelines, which match local point-level descriptors across scans, the problem here is to determine which per-view segments belong to the same physical object. Solving this problem allows downstream tasks such as 3D reconstruction, navigation planning, and object manipulation to reason at the object level rather than at the viewpoint level. The challenge is especially important in unstructured environments such as planetary surfaces and disaster-response scenes. Fig. 1 contrasts terrestrial earthquake rubble with lunar boulder fields; despite their different origins, both exhibit dense clutter, severe occlusion, and fragmented object visibility.

Prior work on related problems falls into several families. *Point-level* feature methods match local descriptors across scans using hand-crafted [3], [4] or learned [5], [6] representations. *Image-plane* multi-object trackers such as SORT [7] and ByteTrack [8] propagate object identities across video frames. A third alternative fuses all views into a single point cloud and applies a 3D instance segmentation method such as Mask3D [9] or SoftGroup [10]. None of these families was designed for the combination of partial per-view visibility, near-uniform surface appearance, and segment-level correspondence considered here.

(a) (b)

Fig. 1. Comparison of (a) urban destruction in Idlib, Syria, after an earthquake [1] and (b) boulders on the rim of Hadley Rille photographed during Apollo 15 [2]. Both scenes exhibit visually similar cluttered rubble fields, including dense irregular fragments and strong occlusions, motivating the focus on correspondence in highly cluttered environments.

These limitations motivate a different perspective. Rather than matching visible surface geometry directly, this work infers segment-level correspondence from volumetric consensus using known camera poses. Each observed segment is ray cast through an estimated thickness to generate a volumetric "shadow" extending backward into the space the underlying rock must occupy. When segments from different viewpoints cast overlapping shadows into the same 3D region, that spatial consensus provides strong evidence that they correspond to the same rock instance, even when their visible surfaces do not overlap. This principle is illustrated in Fig. 2.

This voxel-based formulation is particularly useful because a direct segment-level clustering strategy is less suitable for the problem setting considered here. Segments contain highly variable numbers of points and do not conform to a regular spatial grid. Direct comparison therefore either requires global pooling, which discards local spatial relationships, or exhaustive pairwise computations across all segment pairs. In contrast, voxelization converts these irregular observations into a shared discretized representation in which overlap is expressed

Y. Ruan and E. Komendera are with the Department of Mechanical Engineering, Virginia Polytechnic Institute and State University, Blacksburg, VA, USA (e-mail: yiyanr97@vt.edu; komendera@vt.edu).

Code and pretrained model: https://github.com/JXNICHOLAS/ShadowCorr. Dataset: https://doi.org/10.5281/zenodo.18917286.

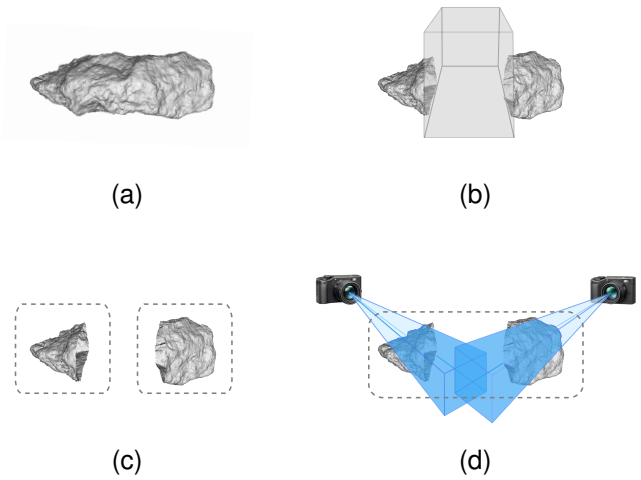

Fig. 2. (a) One lunar rock. (b) When the center of the rock is occluded, only the two side segments remain visible. (c) Without additional information, such as prior shape knowledge, traditional algorithms tend to cluster the two non-overlapping segments separately. (d) With known camera poses, the correct correspondence can instead be inferred from the overlap of the projected volumetric shadows.

implicitly through position. When multiple segments project into the same voxel, potential correspondence is naturally encoded at that location. A permutation-invariant encoder can then aggregate the variable-length set of segments associated with each voxel into a fixed-length representation, enabling efficient sparse 3D convolution for correspondence inference.

The method assumes that RGB-D imagery, 2D instance segmentation, and calibrated camera poses are available as input. These assumptions are reasonable in many robotic perception pipelines. State-of-the-art 2D instance segmentation models, such as SAM 2 [11], Mask R-CNN [12], and YOLOv11 [13], have demonstrated strong performance across diverse benchmarks. RGB-D sensing is also widely used in robotics. Camera poses can be estimated through visual odometry [14], visual-inertial odometry [15], SLAM [16], or registration methods such as ICP [17] when sufficient background geometric overlap is present. Training and evaluation use a large-scale synthetic dataset generated in Unreal Engine 5.5 [18] with NASA lunar rock meshes [19], providing pixel-perfect ground-truth annotations for controlled assessment.

Together, these design choices form ShadowCorr, a framework that addresses segment-level correspondence from a fundamentally different perspective by reasoning about where rocks must be rather than matching only what is directly visible. The main contributions of this work are:

- A volumetric segment-level correspondence formulation that adapts classical multi-view back-projection ideas to the problem of associating unidentified partial 3D segments across viewpoints.
- A finite-thickness, confidence-weighted projection scheme that converts each visible segment shell into a volumetric support region suitable for cross-view correspondence reasoning.
- A scene-specific, permutation-invariant voxel encoder inspired by Word2Vec-style co-occurrence learning [20], which learns segment-ID embeddings from within-voxel co-occurrence and maps segment-ID sets to fixed-dimensional voxel features.
- A sparse-tensor correspondence learning framework that integrates local attention and a tailored multi-loss objective to learn discriminative embeddings for cross-view segment association.
- A publicly released Unreal Engine multi-view RGB-D lunar rock dataset with instance annotations [21].
- Experimental results on synthetic lunar terrain scenes demonstrating strong gains over existing 3D instance segmentation and clustering baselines.

II. Related Work

A. Volumetric Inference

The broader idea of back-projecting observations into a shared 3D volume has classical roots in visual hull [22] and space carving [23]. These methods established that consistent evidence from multiple calibrated viewpoints can be accumulated in 3D space to constrain object geometry, providing a principled foundation for volumetric multi-view reasoning. Visual hull constructs the maximal volume consistent with object silhouettes, while space carving further refines the volume by removing voxels whose projected appearance is inconsistent across views. However, both methods assume that the relevant object support in each image is already known through silhouettes or appearance cues associated with the same object. As a result, correspondence is treated as an implicit prerequisite for reconstruction rather than as the problem to be solved.

ShadowCorr builds on this volumetric intuition but reformulates it for correspondence discovery. Instead of reconstructing shape from silhouettes or carving a globally consistent volume, ShadowCorr projects pre-segmented partial depth shells into a shared voxel grid and uses a learned sparse CNN to infer correspondences from cross-view co-occurrence patterns. Unlike unbounded silhouette back-projection, ShadowCorr extends each segment only a finite distance behind the observed surface using a confidence-weighted projection that limits volumetric bleeding and captures depth uncertainty. In this sense, ShadowCorr inherits the volumetric consensus intuition from prior multi-view geometry but replaces explicit reconstruction rules with a learned correspondence inference pipeline.

B. 3D Instance Segmentation

Point cloud instance segmentation methods partition a single scan into object-level instances. Early approaches used hand-crafted geometric features together with region-growing algorithms, whereas more recent learning-based methods extract discriminative features with deep neural networks. Proposal-based methods such as 3D-SIS [24] and 3D-MPA [25] generate object proposals and refine them through classification, while proposal-free methods such as PointGroup [26], HAIS [27], and SoftGroup [10] directly group points using learned embeddings and bottom-up clustering strategies.

Transformer-based architectures, including Mask3D [9] and SPFormer [28], use attention mechanisms to capture global context and improve instance boundary detection. These methods achieve strong performance on indoor-scene benchmarks such as ScanNet [29] and S3DIS [30], where observations are largely complete and represented in a unified coordinate frame.

These methods are designed for scenes in which observations are relatively complete and per-view identity is discarded by construction when point clouds are fused. The segment-level, partial-observation regime considered here differs on both counts, motivating a representation that reasons about cross-view co-occurrence rather than fused surface geometry alone.

C. Point-Level Matching and Low-Overlap Registration

Classical local descriptor methods such as FPFH [3] and SHOT [4] encode geometric context around each point using histograms of surface normals and curvature. Learned counterparts such as 3DMatch [5] and FCGF [6] replace hand-crafted features with deep network descriptors trained on corresponding patch pairs. Both families perform well when objects exhibit distinctive geometric structure and the scans to be matched share substantial visible surface area. In the segment-level correspondence setting considered here, however, viewpoint-dependent segmentation often yields fragments with little or no shared surface. Touching rocks additionally produce adjacent segments with similar boundary geometry, causing geometric matching to either split same-instance segments or merge different-instance ones.

To address limited overlap more directly, several deep-learning approaches infer shape compatibility or complete missing geometry before matching. Methods such as CTF-Net [31] improve matching under occlusion by jointly predicting visibility masks and deformed geometry flows. Completion-driven registration methods such as Mutual-Prior [32] and PuzzleNet [33] use cross-fragment reasoning to reconstruct missing surfaces and align complementary shapes, enabling registration even when shared geometry is minimal. Despite their effectiveness, these methods typically assume that fragments arise from rigid objects with predictable or learnable shape priors, such as indoor furniture or vehicles. Lunar rocks, in contrast, exhibit highly irregular morphology and viewpoint-dependent fragmentation, making category-driven or prior-based completion unreliable.

D. 2D Multi-Object Tracking

Two-dimensional multi-object trackers such as SORT [7] and ByteTrack [8] propagate object identity across frames using Kalman-filter motion models and frame-to-frame bounding-box IoU association, and are explicitly designed for sequential monocular video with short occlusions. In the multi-view setting considered here, each camera provides an independent spatial observation rather than a temporally ordered frame, so the motion-continuity assumption these trackers rely on does not hold, making them inapplicable by design rather than merely suboptimal.

Algorithm 1 ShadowCorr Preprocessing (one scene)

Require: Per-viewpoint segments $\{(\mathbf{P}_{\mathrm{gt}}, \mathbf{T}_{\mathrm{cam}}, l, y)\}$: surface points, camera pose, segment ID l, rock instance label y
Ensure: Sparse voxel grid $\{(\mathbf{c}_i,\, \gamma_i,\, \mathcal{S}_i,\, \mathbf{z}_i^{\mathrm{vox}},\, y_i)\}$

for all segments $(\mathbf{P}_{\mathrm{gt}}, \mathbf{T}_{\mathrm{cam}}, l, y)$ **do**
  Voxel-downsample and remove statistical outliers from $\mathbf{P}_{\mathrm{gt}}$
  Fit error ellipsoid to $\mathbf{P}_{\mathrm{gt}}$; compute axes $\ell_{1,2,3}$ and projection depth d_{surf} (Sec. III-B)
  for all points $\mathbf{p}_k \in \mathbf{P}_{\mathrm{gt}}$ **do**
    Compute ρ_k, assign $N_{\mathrm{proj}}^{(k)}$, ray-cast to add projected points to $\mathbf{P}_{\mathrm{vol}}$ (Sec. III-C)
  end for
  Discretize $\mathbf{P}_{\mathrm{gt}} \cup \mathbf{P}_{\mathrm{vol}}$ into voxels; retain only voxels with a point in the central 80% of the cell (Chebyshev filter) (Sec. III-D)
  for all voxels i touched by this segment **do**
    Insert l into $\mathcal{S}_i$; accumulate $\phi(\hat{d}_i)$ into γ_i; record y as candidate label for voxel i (Eq. 5)
  end for
end for
Assign each voxel label y_i by highest accumulated confidence; retain top 50% of voxels by γ_i
Build co-occurrence matrix $\mathbf{O}$ from $\{\mathcal{S}_i\}$; learn per-scene embeddings $\{\mathbf{z}(l)\}$ via $\mathcal{L}_{\mathrm{co}}$ (Sec. III-E)
For each voxel i, set $\mathbf{z}_i^{\mathrm{vox}} \leftarrow \ell_2\text{-norm}\big(\frac{1}{|\mathcal{S}_i|}\sum_{l\in\mathcal{S}_i}\mathbf{z}(l)\big)$ (Eq. 8)

III. Data Preparation

The proposed pipeline converts synthetic multi-view rock observations into the voxelized inputs used by the model. Unreal Engine scenes provide segmented rock point clouds together with calibrated camera poses, which are further processed through camera-ray projection and discretized into a fixed-resolution voxel grid. This representation encourages segments from the same rock to produce consistent, overlapping volumetric estimates, while segments from different rocks remain separated in voxel space. Point cloud processing and visualization are performed using the Open3D framework [34]. Algorithm 1 gives a complete summary; the subsections that follow derive each step in detail.

A. Unreal Engine Synthetic Data

No existing public dataset provides multi-view calibrated RGB-D, per-instance 3D labels, and ground-truth cross-view correspondence for cluttered rock fields. To accelerate data collection and increase scene diversity beyond what is practical in real-world experiments, a custom virtual environment was therefore developed in Unreal Engine 5.5 [18] to simulate lunar-surface rock scattering.

The environment includes a high-fidelity lunar terrain model, eight fixed virtual RGB-D cameras, and twenty high-resolution NASA lunar rock meshes [19]. The released dataset records all eight views, and four of them, spaced at 90° azimuthal intervals for full 360° coverage, are used in the experiments reported here. Unreal Engine's lighting system

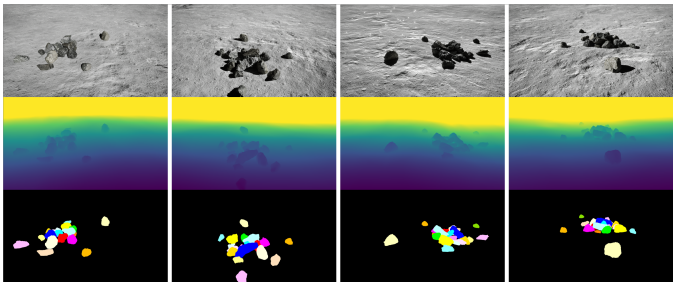

Fig. 3. Unreal Engine-generated color, depth, and semantic images for one scene from four camera positions.

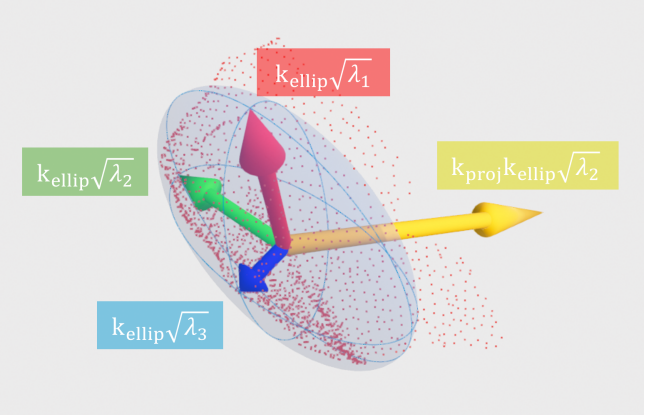

Fig. 4. Error ellipsoid, semi-axes, and camera projection direction for a rock segment. red ■ Major semi-axis; green ■ Intermediate semi-axis; blue ■ Minor semi-axis; yellow ■ Camera projection direction and length.

reproduced the strong shadowing and brightness variation characteristic of lunar polar illumination. Depth and semantic segmentation images were rendered using custom depth buffers and stencil-mask post-processing volumes. All images were captured at a resolution of 1920×1080 pixels, with depth stored in 32-bit floating-point format, as shown in Fig. 3.

A C++ script randomly instantiated 20 rocks, with diameters ranging from approximately $36\,\mathrm{mm}$ to $157\,\mathrm{mm}$, within a $1\,\mathrm{m} \times 1\,\mathrm{m} \times 1.5\,\mathrm{m}$ bounding volume above the terrain without overlap, simulating natural rock scattering. The rocks then underwent gravity-driven free-fall and settled onto the surface. The terrain's physical collision mesh was offset $5\,\mathrm{cm}$ below the visual surface mesh, so rocks partially penetrated the visual terrain layer after settling, creating rock–terrain interfaces that mimic natural partial burial in regolith. Because the terrain surface was modeled as a solid mesh, small regolith fluctuations at the rock–surface interface were not represented. However, these local variations have a limited influence on the correspondence process because the regolith is stenciled out as background during segmentation.

From these rendered observations, the pipeline extracts per-viewpoint 3D rock segments together with their associated camera poses. Both are obtained directly from the simulation: camera poses follow the scene configuration, and segmentation masks are extracted from color-coded stencil buffers, providing pixel-perfect ground-truth labels independent of upstream segmentation errors. Each segment consists of the back-projected depth points visible from a single viewpoint, forming a partial surface shell of the underlying rock. The full dataset is publicly available [21].

B. Camera-Based Point Projection and Error Ellipsoid

Because each input segment represents only a partially visible surface shell, the projection stage requires a principled estimate of the rock's local orientation and interior extent before casting points into the occluded volume. To obtain this estimate, an error ellipsoid [35] is fitted from the segment covariance, providing both principal directions and characteristic axis scales for stable volumetric projection.

Let $\mathbf{P} = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\}$ denote the set of 3D points, where $\mathbf{p}_k = (p_{kx}, p_{ky}, p_{kz})^\top$. For voxel-downsampled point clouds with approximately uniform density, the error ellipsoid is fitted from the segment's sample mean $\boldsymbol{\mu} = \frac{1}{n}\sum_{k=1}^{n} \mathbf{p}_k$ and sample covariance $\boldsymbol{\Sigma} = \frac{1}{n}\sum_{k=1}^{n}(\mathbf{p}_k - \boldsymbol{\mu})(\mathbf{p}_k - \boldsymbol{\mu})^\top$. Its eigenvalues $\{\lambda_1, \lambda_2, \lambda_3\}$ and eigenvectors $\{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3\}$ define the ellipsoid axis lengths and orientation, respectively. The eigenvalues are sorted in descending order, $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq 0$, so that ℓ_1, ℓ_2, and ℓ_3 correspond to the major, intermediate, and minor axes:

$$\ell_i = k_{\mathrm{ellip}}\sqrt{\lambda_i}, \quad i \in \{1, 2, 3\} \tag{1}$$

The axis scaling factor k_{ellip} is a confidence-level parameter derived from the chi-square distribution with 3 degrees of freedom, where each value corresponds to a specific coverage probability under a multivariate normal assumption. Here, $k_{\mathrm{ellip}} = 1.765$ was empirically tuned to fit the rock volumes in NASA's dataset [19].

Projection Direction: Each segment is represented by a point cloud $\mathbf{P}_{\mathrm{gt}} = \{\mathbf{p}_k\}$ and a camera pose $\mathbf{T}_{\mathrm{cam}} \in \mathrm{SE}(3)$, represented as a 4×4 homogeneous transform from the camera frame to the world frame. The projection direction is defined from the camera position toward the segment's error ellipsoid center $\boldsymbol{\mu}$. The ellipsoid center is also projected onto the camera image plane to obtain the 2D projection center (u_0, v_0), which is later used as the reference point for silhouette-based erosion in Section III-C.

Projection Depth: As illustrated in Fig. 4, the intermediate eigenvalue λ_2 is used as a robust estimator of lateral width. Using the largest eigenvalue λ_1 often produces overly deep projections for elongated objects, such as rod-like geometries, whereas the smallest eigenvalue λ_3 tends to underestimate thickness because of the limited depth information available from a single view. The effective projection depth is therefore defined by expanding the calibrated ellipsoid width:

$$d_{\mathrm{surf}} = k_{\mathrm{proj}}\, k_{\mathrm{ellip}}\sqrt{\lambda_2} \tag{2}$$

The expansion factor k_{proj} is tuned empirically in Section V-B to balance volumetric completeness against geometric conservativeness.

C. Silhouette-Aware Erosion

The projected volumetric points are discretized into a voxel grid with voxel size v_{size}, which controls the spatial resolution of the representation. This parameter introduces a fundamental

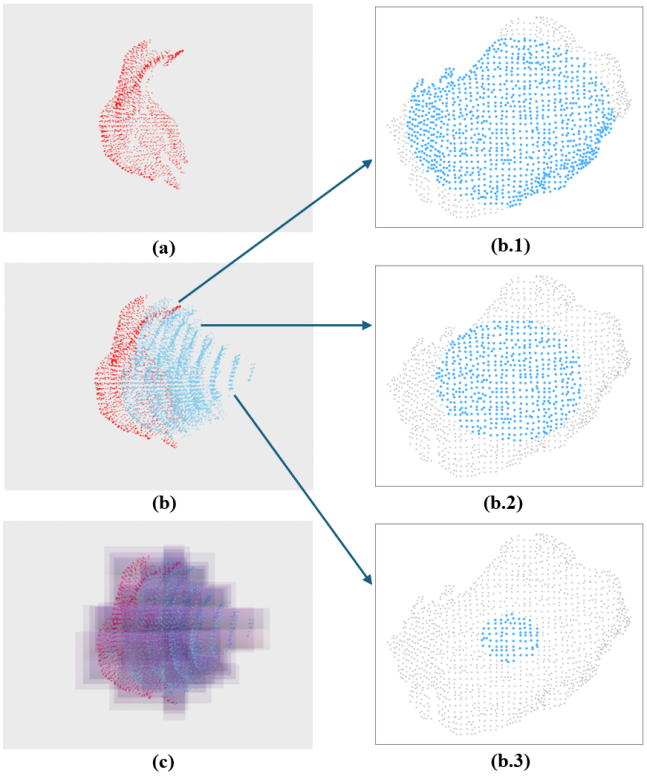

Fig. 5. Voxel generation for a rock segment. (a) red ■ denotes the ground-truth segment points. (b) blue ■ denotes the projected points. (c) purple ■ shows the resulting voxel volume; darker (less transparent) purple indicates higher voxel confidence. Insets (b.1)–(b.3) visualize representative eroded projection layers (2nd, 4th, and 7th) from the camera perspective, where filtered-out points are shown in gray ■.

trade-off: smaller voxel sizes preserve finer geometric detail at the cost of increased memory and computation, whereas larger voxel sizes reduce complexity but may under-represent small objects. The optimal value of v_{size} is determined empirically in Section V-B through evaluation of the accuracy–efficiency trade-off on the dataset.

A silhouette-aware erosion function spatially modulates the projection depth to avoid excessive lateral expansion near object boundaries, as illustrated in Fig. 5. If all surface points were projected to the same depth, the resulting volume would exhibit unrealistic "fanning" artifacts, in which the projected shape expands laterally with increasing distance from the camera.

Specifically, for each surface point $\mathbf{p}_k$, let (ξ_k, η_k) denote its image-plane projection. The normalized silhouette distance ρ_k measures the bounding-box-normalized elliptical radius of this projection relative to the projected ellipsoid center:

$$\rho_k = \sqrt{\left(\frac{2(\xi_k - u_0)}{w_{\text{bb}}}\right)^2 + \left(\frac{2(\eta_k - v_0)}{h_{\text{bb}}}\right)^2}, \tag{3}$$

where (u_0, v_0) is the projected ellipsoid center and $(w_{\text{bb}}, h_{\text{bb}})$ are the bounding-box dimensions used for normalization. This formulation produces elliptical erosion contours that taper the volumetric envelope along the image axes. Points with $\rho_k > 1$, which lie outside the normalized bounding region, are assigned zero projection layers and are excluded from volumetric generation.

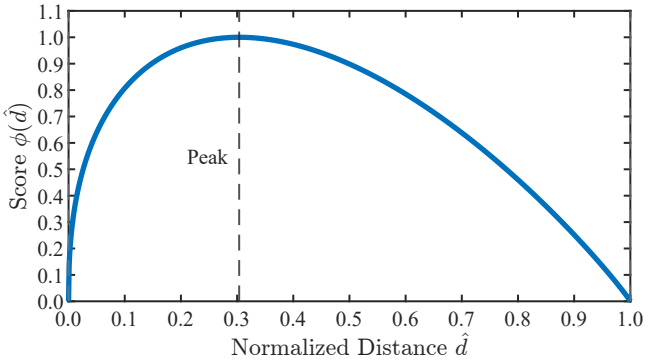

Fig. 6. Voxel confidence scoring function $\phi(\hat{d})$ based on a Beta distribution, normalized to have a maximum value of 1.0.

The number of projection layers assigned to point $\mathbf{p}_k$ is then defined as

$$N_{\text{proj}}^{(k)} = \max\left(0, \lfloor N_{\max} \cdot (1 - \rho_k) \rfloor\right), \tag{4}$$

where $N_{\max} = \lceil d_{\text{surf}} / v_{\text{size}} \rceil$ is the maximum number of layers, computed from the estimated projection depth d_{surf} and the voxel size v_{size}. As a result, boundary points ($\rho_k \to 1$) contribute few or no depth samples, while central points ($\rho_k \to 0$) are projected through the full estimated thickness. This produces a smoothly tapered volumetric envelope that reduces overlap artifacts across adjacent rocks while preserving the overall object geometry.

D. Voxel Generation and Scoring

After ray casting, the observed surface points $\mathbf{P}_{\text{gt}}$ and projected interior points $\mathbf{P}_{\text{vol}}$ are merged and discretized into a sparse voxel grid. A Chebyshev distance filter mitigates aliasing and grazing artifacts at voxel boundaries: voxels are instantiated only if a point lies within the central 80% of the voxel volume, preventing spurious voxels from being created by outlier points near voxel edges.

Within this shared voxel grid, each voxel is further assigned a confidence value to reflect its geometric support relative to the observed surface. Let $\mathbf{c}$ denote the 3D centroid of a voxel cell. The minimum Euclidean distance from $\mathbf{c}$ to the observed surface is computed as $d_{\min}(\mathbf{c})$. These distances are then normalized as $\hat{d}(\mathbf{c}) = d_{\min}(\mathbf{c})/d_{\max}$, where $d_{\max}$ is the maximum distance among all voxels in the segment, ensuring that $\hat{d}(\mathbf{c}) \in [0, 1]$. For a voxel indexed by i, we write $\hat{d}_i := \hat{d}(\mathbf{c}_i)$. The confidence score is defined by a Beta-kernel scoring function:

$$\phi(\hat{d}) = K_\phi \cdot \hat{d}^{\alpha-1}(1 - \hat{d})^{\beta-1}, \quad 0 < \hat{d} < 1 \tag{5}$$

where $\hat{d}$ denotes $\hat{d}(\mathbf{c})$ for the voxel with centroid $\mathbf{c}$, and K_ϕ is a normalization constant chosen so that $\max(\phi) = 1$. The shape is borrowed from the Beta distribution kernel but max-normalized rather than area-normalized. It is governed by $\alpha = 1.41$ and $\beta = 1.94$, chosen empirically such that the score peaks at $\hat{d} = (\alpha - 1)/(\alpha + \beta - 2) = 0.3037$ and becomes zero at the two boundaries, $\hat{d} = 0$ and $\hat{d} = 1$, as illustrated in

Fig. 6. This target peak location is motivated by two competing requirements: (i) the peak must be offset from the ground-truth surface ($\hat{d} = 0$) to prevent voxel bleeding when two rock surfaces are in contact, ensuring that adjacent instances remain separable; and (ii) the peak should remain relatively close to the surface to maintain high confidence in the immediately projected region, where geometric evidence is strongest. The ratio $\beta/\alpha \approx 1.38$ produces an asymmetric profile that decays gradually toward larger $\hat{d}$. This reflects the natural decrease in certainty with increasing depth into the occluded volume: the farther from the observed surface, the less likely the rock mass extends to that location.

After confidence assignment, projected voxels from all segments are aggregated into a unified global grid. Each global voxel stores the identifiers of all contributing segments together with the cumulative confidence score from their projections. Because $\phi(\hat{d})$ is max-normalized to 1 per segment contribution, the per-voxel confidence $\gamma_i = \sum_{l \in \mathcal{S}_i} \phi(\hat{d}_i^{(l)})$ grows with the number of supporting segments rather than being bounded to $[0, 1]$, so voxels with stronger multi-view support receive proportionally higher scores. A global per-centile filter is then applied to retain only the top 50% of voxels ranked by confidence; this threshold was chosen empirically to improve inter-instance separation while reducing computational cost.

This filtering, combined with the confidence scoring, also helps prevent large rocks from overwhelming smaller neighboring ones. Although a large rock may project across a wider spatial region, its confidence still decays with depth from its own observed surface. As a result, voxels near the true location of a smaller neighboring rock tend to receive relatively low confidence from the large rock but high confidence from the small rock itself. Retaining only the highest-confidence voxels therefore preserves the smaller rock's signature while suppressing spurious shadow overlap from the larger one. In addition, the Word2Vec-style segment-ID encoding described in Section III-E captures per-voxel segment co-occurrence patterns, further helping distinguish adjacent instances even when their projected volumes are spatially close.

E. Word2Vec-Style Voxel Segment-ID Embedding

Each voxel records the segment IDs that project into it, forming an unordered set with variable cardinality that must be mapped to a fixed-dimensional vector before it can be used by the sparse 3D CNN. To do this, a precomputed encoder inspired by the Word2Vec distributional hypothesis [20] is used. The core idea is that segment IDs that frequently co-occur in the same voxels should be embedded nearby, while unrelated IDs should be separated in embedding space. The encoder is trained during preprocessing and then held fixed during downstream network training and inference, yielding a permutation-invariant mapping from unordered ID sets to fixed-dimensional voxel features.

Here, a *segment ID* denotes the unique integer label assigned to one segmented rock observation in the input. Because the same physical rock may be observed from multiple viewpoints, it can appear under several distinct segment IDs. Voxels inside that rock's projected volume may therefore contain multiple IDs that correspond to the same underlying instance. These repeated co-occurrence patterns provide the correspondence signal used by the downstream network. Near object boundaries, however, a voxel may also contain IDs from adjacent rocks due to overlap between projected volumes. Each voxel is therefore naturally represented as an unordered set of segment IDs with variable cardinality.

Input representation and embedding table: For each voxel i, let $\mathcal{S}_i = \{l_1, \ldots, l_{N_i}\}$ denote the set of segment IDs that project into that voxel, where N_i is the number of IDs in the set and each l_m is an integer segment ID. Let K denote the number of unique segment IDs in a scene. Each segment ID l is assigned an embedding vector $\mathbf{z}(l) \in \mathbb{R}^{D_{\text{emb}}}$ through a standard embedding lookup table, where $D_{\text{emb}} = 12$ in this work. The embedding table is trained independently for each scene so that the learned vectors reflect the scene-specific co-occurrence structure of segment IDs.

Contrastive training via co-occurrence: A symmetric co-occurrence matrix $\mathbf{O} \in \mathbb{Z}_{\geq 0}^{K \times K}$ is precomputed for each scene, where O_{ab} counts the number of voxels in which segment IDs a and b both appear. In practice, O_{aa} is set to zero, pairs with $O_{ab} > 0$ are treated as positive pairs, and pairs with $O_{ab} = 0$ are treated as negatives. The raw counts are row-normalized into a distribution over positive partners:

$$\hat{O}_{ab} = \frac{O_{ab}}{\sum_{q \neq a} O_{aq}}, \tag{6}$$

so that $\hat{O}_{ab} \geq 0$ and $\sum_b \hat{O}_{ab} = 1$, with $\hat{O}_{ab} = 0$ whenever $O_{ab} = 0$. The embedding table is then trained using a co-occurrence-weighted multi-positive contrastive objective [36]:

$$\mathcal{L}_{\text{co}} = -\frac{1}{|\mathcal{I}|} \sum_{a \in \mathcal{I}} \log \frac{\sum_{b \neq a} \hat{O}_{ab} \, \exp(\text{sim}(\mathbf{z}_a, \mathbf{z}_b)/\tau_{\text{co}})}{\sum_{b \neq a} \exp(\text{sim}(\mathbf{z}_a, \mathbf{z}_b)/\tau_{\text{co}})}, \tag{7}$$

where $\mathbf{z}_a := \mathbf{z}(a)$ denotes the embedding of segment ID a, $\text{sim}(\cdot, \cdot)$ denotes cosine similarity, τ_{co} is a temperature parameter, and $\mathcal{I}$ is the set of segment IDs that have at least one positive co-occurrence partner. We place the co-occurrence weights $\hat{O}_{ab}$ *inside* the logarithm rather than outside, as is more common in supervised-contrastive formulations [37]. We found this variant to yield more stable training in our setting, where some positive pairs are weakly supported. Because $\hat{O}_{ab}$ forms a convex combination over positive partners, the numerator is bounded by the denominator, ensuring $\mathcal{L}_{\text{co}} \geq 0$. Pairs with larger $\hat{O}_{ab}$ contribute correspondingly larger terms to the numerator: segment pairs that co-occur in many voxels therefore exert stronger attractive influence during optimization, reflecting the strength of their correspondence evidence. This weighting also captures indirect co-occurrence structure. For example, two ID groups need not overlap directly to be placed nearby in embedding space if they are linked through intermediate co-occurrences. In this way, partially observed segment relationships can still influence the learned embedding geometry.

Voxel-level embedding and training strategy: After training, the embedding for voxel i is computed as the ℓ_2-

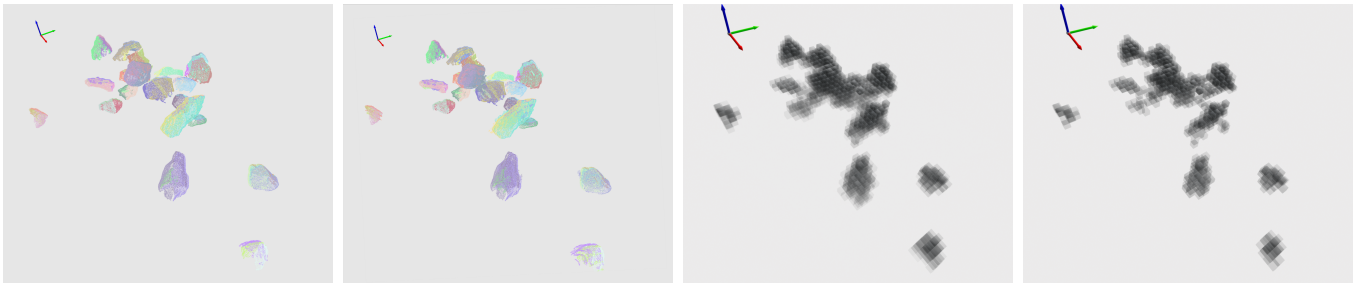

Fig. 7. ShadowCorr data preprocessing workflow: (a) original point cloud, (b) point cloud with projected dots, (c) voxelized grid, and (d) filtered voxel grid retaining voxels with the top 50% confidence scores, which forms the neural network input. For visualization clarity, the voxel grids shown in (c) and (d) use an enlarged voxel size of $v_{\text{size}} = 16$, which is larger than the value used in the main experiments.

normalized mean of the embeddings of all segment IDs in $\mathcal{S}_i$:

$$\mathbf{z}_i^{\text{vox}} = \frac{\bar{\mathbf{z}}_i}{\|\bar{\mathbf{z}}_i\|_2}, \qquad \bar{\mathbf{z}}_i = \frac{1}{N_i} \sum_{m=1}^{N_i} \mathbf{z}(l_m), \tag{8}$$

where $\mathbf{z}_i^{\text{vox}} \in \mathbb{R}^{D_{\text{emb}}}$ is the final voxel embedding ($D_{\text{emb}} = 12$). Averaging over $\mathcal{S}_i$ makes the construction permutation-invariant by design, so the ordering of IDs within $\mathcal{S}_i$ does not affect the output. The ℓ_2 normalization projects the averaged embedding onto the unit hypersphere, keeping feature magnitudes consistent across voxels. The embedding dimension of 12 was selected empirically to balance representational capacity and computational efficiency. In our implementation, the embedding table is optimized with Adam at a learning rate of 10^{-3} for 200 gradient steps, and the temperature in Eq. 7 is fixed at $\tau_{\text{co}} = 0.1$. The resulting voxel embeddings are stored together with voxel coordinates and confidence scores in the NPZ files. During downstream clustering-network training and inference (Section IV), these embeddings are loaded and preprocessed as fixed input features and are not updated by backpropagation.

IV. Neural Network Training

The data preparation pipeline described in Section III produces, for each scene, a sparse voxel grid with per-voxel features comprising spatial coordinates, confidence scores, and segment-set embeddings (Fig. 7). A sparse 3D convolutional neural network implemented in TorchSparse [38] then maps each occupied voxel to a discriminative high-dimensional embedding. Throughout this section, *features* refer to intermediate numerical representations at any stage of the network, whereas *embeddings* specifically denote the final fixed-dimensional vectors used for clustering. A multi-objective loss function described in Section IV-D supervises training to promote both global separability and local geometric consistency.

A. Instance Embedding Architecture

The input to the network is represented as a sparse tensor $\mathcal{X}$ with two distinct components. First, the *coordinate matrix* $\mathbf{C} \in \mathbb{Z}^{N\times 4}$ stores integer grid indices $[\text{batch}, x, y, z]$ for each of the N occupied voxels, defining the discrete spatial structure used by sparse convolution and attention operators for efficient neighbor retrieval. Second, the *feature matrix* $\mathbf{F} \in \mathbb{R}^{N\times 16}$ contains the learnable representations processed by the neural network.

For each occupied voxel i, the 16-dimensional input feature vector concatenates the scalar confidence $\gamma_i = \phi(\hat{d}_i)$ with $\hat{d}_i = \hat{d}(\mathbf{c}_i)$ (Section III-D, Eq. 5), the scene-relative normalized spatial coordinate $\mathbf{r}_i \in \mathbb{R}^3$ (computed from $\mathbf{c}_i$), and the 12-dimensional segment-ID embedding $\mathbf{z}_i^{\text{vox}}$ from Eq. 8, giving $\mathbf{x}_i = [\gamma_i,\ \mathbf{r}_i^\top,\ (\mathbf{z}_i^{\text{vox}})^\top]^\top \in \mathbb{R}^{16}$. The normalized coordinates $\mathbf{r}_i \in [-1, 1]^3$ are centered at the scene's spatial center and uniformly scaled by the scene's maximum extent, providing continuous positional information as input features, distinct from the integer grid indices in $\mathbf{C}$, which define topology for sparse operations.

B. Multi-Head Local Geometric Attention

A Multi-Head Local Attention mechanism complements the sparse convolutional backbone to enhance feature discriminability while maintaining computational tractability. For each voxel i, the k_{attn}-nearest spatial neighbors $\mathcal{N}_{k_{\text{attn}}}(i)$ are retrieved in 3D coordinate space. Unlike full self-attention [39], which computes pairwise relationships across all voxels with $O(N^2)$ complexity, local attention restricts the receptive field to k_{attn} neighbors, reducing the complexity to $O(Nk_{\text{attn}})$.

The key difference from convolution is that convolutional kernels apply fixed learned weights uniformly across spatial locations, whereas attention computes *input-dependent* weights based on feature similarity. Let $\mathbf{f}_i \in \mathbb{R}^{D_f}$ denote the feature vector for voxel i at the current layer, where $\mathbf{f}_i$ is the i-th row of the intermediate feature matrix and D_f is the CNN feature dimension at that stage (32 after the first convolution and 64 after the second). For each head $h \in \{1, \ldots, N_H\}$, the features $\mathbf{f}_i$ and $\mathbf{f}_j$ (for neighbor $j \in \mathcal{N}_i$, where $\mathcal{N}_i := \mathcal{N}_{k_{\text{attn}}}(i)$ is shorthand for voxel i's k_{attn}-NN neighborhood) are linearly projected into per-head query, key, and value vectors $\mathbf{q}_i^{(h)} = \mathbf{W}_q^{(h)}\mathbf{f}_i$, $\mathbf{k}_j^{(h)} = \mathbf{W}_k^{(h)}\mathbf{f}_j$, and $\mathbf{v}_j^{(h)} = \mathbf{W}_v^{(h)}\mathbf{f}_j$ via learned projection matrices $\mathbf{W}_q^{(h)}, \mathbf{W}_k^{(h)}, \mathbf{W}_v^{(h)}$.

Attention weights are then computed via scaled dot-product similarity and normalized over the neighborhood. Let $D_h = D_f/N_H$ denote the per-head dimension, i.e., the dimensional-

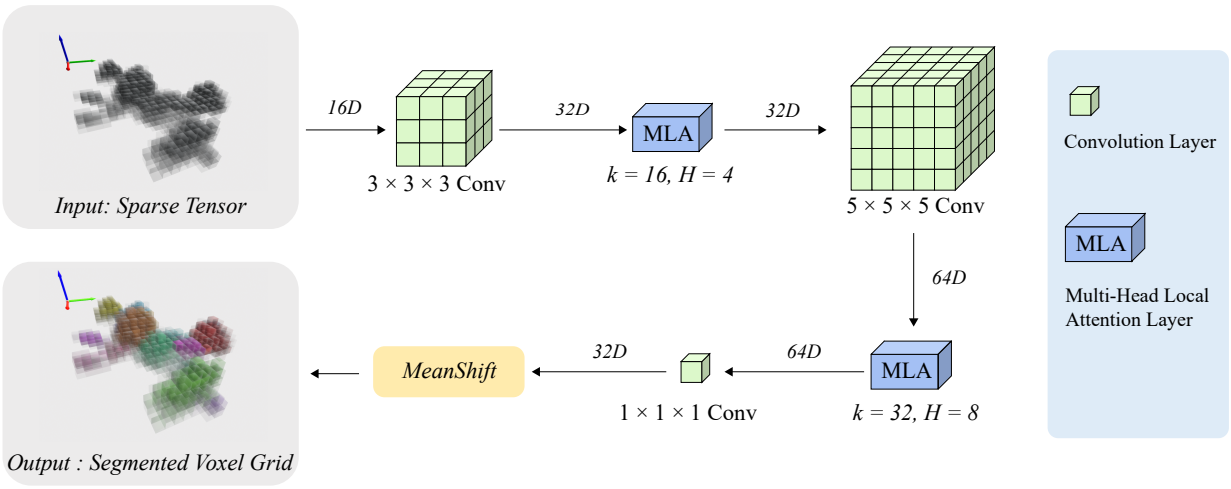

Fig. 8. Architecture overview of ShadowCorr neural network.

ity of each query and key vector:

$$a_{ij}^{(h)} = \frac{\exp\left(\mathbf{q}_i^{(h)\top}\mathbf{k}_j^{(h)}/\sqrt{D_h}\right)}{\sum_{j'\in\mathcal{N}_i}\exp\left(\mathbf{q}_i^{(h)\top}\mathbf{k}_{j'}^{(h)}/\sqrt{D_h}\right)}. \tag{9}$$

Here, $a_{ij}^{(h)}$ denotes the attention weight assigned by voxel i to neighbor j under head h. The refined per-head feature is obtained by the weighted sum $\tilde{\mathbf{f}}_i^{(h)} = \sum_{j\in\mathcal{N}_i} a_{ij}^{(h)}\mathbf{v}_j^{(h)}$.

Outputs from all N_H heads are concatenated and linearly projected to form the final enhanced feature $\tilde{\mathbf{f}}_i \in \mathbb{R}^{D_f}$, preserving the input dimension so that the attention layers remain compatible with the convolutional backbone. Because the attention weights are input-dependent, each voxel's aggregation adapts to its local feature context rather than applying a fixed kernel, and the multiple heads provide parallel projections of the same neighborhood into different subspaces.

C. Network Architecture

The overall architecture follows a shallow sparse 3D convolutional backbone with interleaved local attention blocks to better capture anisotropic rock geometries. As shown in Fig. 8, the network consists of three sequential processing stages:

Stage 1 (Feature extraction): A $3 \times 3 \times 3$ sparse 3D convolution maps each 16-dimensional input $\mathbf{x}_i$ to a 32-dimensional intermediate representation, followed by Batch Normalization [40] and ReLU activation [41]. A Multi-Head Local Attention module ($k_{\text{attn}} = 16$, $N_H = 4$) then refines the 32-dimensional features through adaptive neighbor aggregation.

Stage 2 (Feature refinement): A $5 \times 5 \times 5$ sparse 3D convolution with a larger receptive field expands the representation to 64 dimensions, followed by Batch Normalization and ReLU. A second Multi-Head Local Attention module ($k_{\text{attn}} = 32$, $N_H = 8$ heads) then operates on the higher-dimensional features with an increased neighborhood size.

Stage 3 (Embedding projection): A $1 \times 1 \times 1$ convolution projects the 64-dimensional features into the final 32-dimensional discriminative embedding space. This compact embedding dimension balances discriminative power with computational efficiency during the subsequent clustering step. The progressive expansion ($16 \to 32 \to 64$), followed by compression ($64 \to 32$), allows the network to learn rich intermediate representations while maintaining a manageable final embedding space suitable for MeanShift clustering.

D. Multi-Objective Clustering Loss

Let $\mathbf{e}_i \in \mathbb{R}^{32}$ denote the embedding produced by the network for voxel i. The network is trained with a multi-objective loss function $\mathcal{L}_{\text{total}}$ that combines three complementary clustering signals: discriminative instance spacing [42], prototypical stability [43], and local graph consistency [44].

$$\mathcal{L}_{\text{total}} = \omega_{\text{disc}}\mathcal{L}_{\text{disc}} + \omega_{\text{proto}}\mathcal{L}_{\text{proto}} + \omega_{\text{graph}}\mathcal{L}_{\text{graph}}. \tag{10}$$

1) Discriminative Loss ($\mathcal{L}_{\text{disc}}$): A vectorized discriminative loss [42] is used to enforce intra-cluster compactness and inter-cluster separation:

$$\mathcal{L}_{\text{disc}} = w_{\text{var}}\mathcal{L}_{\text{var}} + w_{\text{dist}}\mathcal{L}_{\text{dist}} + w_{\text{reg}}\|\boldsymbol{\nu}\|_2, \tag{11}$$

where $\boldsymbol{\nu}$ denotes the concatenation of all cluster centroids in the embedding space.

For each instance c with voxel set $\mathcal{C}_c$, the centroid in embedding space is defined as $\boldsymbol{\nu}_c = \frac{1}{|\mathcal{C}_c|}\sum_{i\in\mathcal{C}_c}\mathbf{e}_i$. The variance loss penalizes embeddings that deviate too far from their centroid:

$$\mathcal{L}_{\text{var}} = \frac{1}{N_{\text{c}}}\sum_{c=1}^{N_{\text{c}}}\frac{1}{|\mathcal{C}_c|}\sum_{i\in\mathcal{C}_c}\max\left(\|\mathbf{e}_i - \boldsymbol{\nu}_c\|_2 - \delta_{\text{var}}, 0\right)^2, \tag{12}$$

where N_{c} is the number of clusters in the batch.

The distance loss penalizes cluster centroids that are too close:

$$\mathcal{L}_{\mathrm{dist}} = \frac{1}{N_{\mathrm{c}}(N_{\mathrm{c}}-1)} \sum_{c=1}^{N_{\mathrm{c}}} \sum_{\substack{c'=1 \\ c' \neq c}}^{N_{\mathrm{c}}} \max\left(2\delta_{\mathrm{dist}} - \|\boldsymbol{\nu}_c - \boldsymbol{\nu}_{c'}\|_2, 0\right)^2. \tag{13}$$

2) Prototypical Loss ($\mathcal{L}_{\mathrm{proto}}$): A prototypical loss [43] stabilizes training and provides global inter-cluster supervision. This loss treats clustering as a metric-learning problem that models a distribution over instance assignments. Such a probabilistic formulation provides stronger gradient signals for ambiguous boundary voxels and encourages embeddings to be separated by clear margins while forming well-structured clusters in the embedding space.

Let $\hat{\mathbf{e}}_i = \mathbf{e}_i/\|\mathbf{e}_i\|_2$ and $\hat{\boldsymbol{\nu}}_c = \boldsymbol{\nu}_c/\|\boldsymbol{\nu}_c\|_2$ denote the ℓ_2-normalized embeddings and prototypes, respectively. ℓ_2 normalization ensures that the distance metric operates on the unit hypersphere, where Euclidean distance is proportional to angular separation, making the loss invariant to embedding magnitude and focusing purely on directional similarity. Letting $y_i \in \{1, \ldots, N_{\mathrm{c}}\}$ denote the ground-truth instance assignment of voxel i, the prototypical loss is defined as the negative log-likelihood of correct assignments under a softmax over normalized prototype distances:

$$\mathcal{L}_{\mathrm{proto}} = -\frac{1}{N} \sum_{i=1}^{N} \log \frac{\exp(-\|\hat{\mathbf{e}}_i - \hat{\boldsymbol{\nu}}_{y_i}\|_2/\tau_{\mathrm{proto}})}{\sum_{c=1}^{N_{\mathrm{c}}} \exp(-\|\hat{\mathbf{e}}_i - \hat{\boldsymbol{\nu}}_c\|_2/\tau_{\mathrm{proto}})}, \tag{14}$$

where N is the number of occupied voxels in the training batch (the same N as in the sparse tensor of Section IV-A) and τ_{proto} is a temperature parameter controlling the softness of the distribution. This loss is minimized when each voxel embedding lies closer to its own cluster prototype than to any other. Lower temperatures produce sharper decision boundaries.

3) Graph-Based Consistency Loss ($\mathcal{L}_{\mathrm{graph}}$): A spatial k_{graph}-NN graph constructed from voxel coordinates [44] enforces local smoothness of the embedding field. Let $\mathcal{G}_i$ denote the k_{graph}-nearest spatial neighbors of voxel i, and let $\mathcal{E}_{\mathrm{pos}} = \{(i,j) : j \in \mathcal{G}_i,\ y_i = y_j\}$ denote the set of directed edges connecting spatial neighbors that share the same ground-truth instance. Following graph-based manifold regularization for deep embeddings [45], the loss maximizes cosine similarity across same-instance neighbors:

$$\mathcal{L}_{\mathrm{graph}} = -\frac{1}{|\mathcal{E}_{\mathrm{pos}}|} \sum_{(i,j)\in\mathcal{E}_{\mathrm{pos}}} s_{\mathrm{graph}}\, \hat{\mathbf{e}}_i^{\top} \hat{\mathbf{e}}_j, \tag{15}$$

where $\hat{\mathbf{e}}_i$ is the ℓ_2-normalized embedding defined above and $s_{\mathrm{graph}} > 0$ is the graph similarity scale.

Loss weights and hyperparameters: The final model employs equal weights for all three loss components: $\omega_{\mathrm{proto}} = \omega_{\mathrm{disc}} = \omega_{\mathrm{graph}} = 1.0$. The discriminative loss parameters are set to $\delta_{\mathrm{var}} = 0.9$, $\delta_{\mathrm{dist}} = 1.0$, $w_{\mathrm{var}} = 0.8$, $w_{\mathrm{dist}} = 0.35$, and $w_{\mathrm{reg}} = 0.0001$. The prototypical loss uses temperature $\tau_{\mathrm{proto}} = 0.1$, and the graph-based loss uses $s_{\mathrm{graph}} = 10$ with $k_{\mathrm{graph}} = 8$ nearest neighbors.

E. MeanShift Clustering for Instance Extraction

After the network produces the 32-dimensional embeddings $\{\mathbf{e}_i\}_{i=1}^{N}$, the final instance segmentation is obtained via MeanShift clustering [46]. MeanShift is a non-parametric, mode-seeking algorithm that identifies local maxima (modes) in the embedding-space density and assigns each voxel to its nearest mode, thereby naturally discovering the number of rock instances without requiring a pre-specified cluster count. Because the number of visible rocks per scene varies with occlusion and viewpoint, methods requiring a fixed cluster count (e.g., k-means) need an additional estimation step that MeanShift avoids entirely.

The algorithm iteratively shifts each embedding toward a local density maximum by computing a weighted mean of embeddings within a bandwidth b_{ms}. Defining the kernel weight $w_{ij}^{(t)} := \kappa\Big(\|\mathbf{e}_j - \mathbf{m}_i^{(t)}\|/b_{\mathrm{ms}}\Big)$, the update rule is:

$$\mathbf{m}_i^{(t+1)} = \frac{\sum_j w_{ij}^{(t)}\, \mathbf{e}_j}{\sum_j w_{ij}^{(t)}}, \tag{16}$$

where each sum over j runs over all voxel embeddings in the scene being clustered (the same index set as $i = 1, \ldots, N$ in $\{\mathbf{e}_i\}_{i=1}^{N}$ above), and $\kappa(\cdot)$ is a flat, uniform kernel that assigns equal weight to all points within the bandwidth b_{ms} and zero weight outside. The mean $\mathbf{m}_i^{(t)}$ is updated iteratively until convergence, and embeddings that converge to the same mode are assigned to the same cluster. The bandwidth b_{ms} controls the sensitivity of mode detection: smaller values yield finer segmentation, whereas larger values may merge nearby instances. The bandwidth $b_{\mathrm{ms}} = 0.7$ is selected empirically.

A post-processing step then merges spurious clusters containing fewer than 5 voxels into the nearest larger cluster to reduce fragmentation artifacts. Once every voxel has been assigned to a cluster, each input segment inherits its correspondence label by majority voting across the cluster assignments of all voxels within its projected shadow volume.

V. Experimental Evaluation

A. Experiment Setup

Dataset: The dataset employed in this study was constructed using Unreal Engine 5.5 [18]. Each scene contains only rock instances (18–20 per scene), with non-rock background elements filtered out during data generation. The utilized dataset comprises 2,377 scenes, which were randomly partitioned into a 1,977-scene training set, a 200-scene validation set, and a 200-scene test set. Before voxel processing, the original point clouds contain an average of 220,833 points per scene with a standard deviation of 12,637. After voxelization with the selected parameters ($v_{\mathrm{size}} = 8\,\mathrm{mm}$ and $k_{\mathrm{proj}} = 3$, determined in Section V-B), each scene contains approximately 78 segments and 5,227 voxels on average.

Training Details: The model was trained on a workstation equipped with an NVIDIA RTX 5090 GPU and an AMD Ryzen 9800X3D CPU. Training was conducted from scratch using PyTorch [47] for 40 epochs, with mixed-precision (AMP) [48] enabled for computational efficiency. Optimization was performed using the Adam [49] optimizer with a

TABLE I
ACCURACY AND RUNTIME UNDER DIFFERENT VOXEL SIZES (v_{size}) AND PROJECTION DEPTH MULTIPLIERS (k_{proj}).

Voxel Size (mm)	Projection Multiplier	Metrics (mean±std, %)				Runtime (ms)			
		Voxel ARI	Rock Purity	Cluster Purity	Avg Purity	Preprocess	Inference	Postprocess	Total
10	2X	96.11±3.45	97.88±2.23	98.24±1.94	98.06±1.68	2269.3	8.7	700.6	2978.7
	3X	98.37±1.78	99.06±1.34	98.36±1.89	98.71±1.36	2634.7	10.5	584.6	3229.9
	4X	97.99±1.85	98.83±1.42	98.29±1.99	98.56±1.47	3058.6	11.6	668.9	3739.2
8	2X	96.87±3.44	97.97±2.20	98.73±1.69	98.35±1.61	3180.3	15.0	1049.8	4245.2
	3X	**98.66±1.85**	**99.04±1.37**	**98.88±1.50**	**98.96±1.23**	**3712.8**	**15.5**	**949.5**	**4677.8**
	4X	98.36±1.99	98.87±1.43	98.78±1.52	98.82±1.29	4170.4	21.3	1222.0	5413.6
6	2X	96.32±3.70	96.98±2.81	99.21±1.14	98.10±1.65	4212.5	29.9	3191.2	7433.7
	3X	98.71±1.82	98.58±1.75	99.24±1.19	98.91±1.31	5137.6	36.8	2799.6	7974.9
	4X	98.36±2.05	98.51±1.68	99.07±1.21	98.79±1.25	6208.5	51.4	3929.3	10189.2

fixed learning rate of 5×10^{-5}. Model selection was based on the Adjusted Rand Index (ARI) [50] on the validation set, ensuring optimization for clustering quality rather than proxy loss minimization alone.

Inference and Clustering: During inference, voxel embeddings are extracted from the trained network and clustered with MeanShift using a bandwidth of 0.7. A seed value of 42 is used for reproducibility, and clusters with fewer than 5 voxels are merged into neighboring larger clusters.

Evaluation Metrics: Three complementary metrics are used. *ARI* measures agreement across predicted and ground-truth assignments. *Rock purity* measures over-segmentation by evaluating whether segments from the same rock instance remain grouped together. *Cluster purity* measures under-segmentation by evaluating whether each predicted cluster contains segments from only one rock instance. Their average summarizes overall correspondence quality.

B. Voxel Size and Projection Depth Selection

Two critical parameters govern the volumetric representation: the voxel size v_{size} (introduced in Section III-C) determines the spatial discretization resolution, while the projection depth multiplier k_{proj} (introduced in Section III-B) controls how far behind the observed surface the volumetric projection extends. Both parameters involve fundamental trade-offs: smaller voxel sizes provide finer geometric detail but increase memory and computation, while larger projection depths improve volumetric completeness but risk over-projection and inter-instance overlap.

As shown in Table I, these parameters jointly affect segmentation accuracy and runtime. The projection multiplier exhibits a clear optimum: $k_{\text{proj}} = 3$ consistently achieves the highest accuracy across all voxel sizes, outperforming both $k_{\text{proj}} = 2$, which provides insufficient volumetric coverage, and $k_{\text{proj}} = 4$, which causes ambiguity from excessive projection. For voxel size, finer resolution improves representation quality, particularly for smaller rocks. The smallest rock in the test set ($36\,\text{mm}$ diameter) is represented by only 41 voxels at $10\,\text{mm}$ resolution, compared to 81 voxels at $8\,\text{mm}$ and 194 voxels at $6\,\text{mm}$. Although $6\,\text{mm}$ yields slightly higher cluster purity (99.24% vs. 98.88% at $8\,\text{mm}$), $8\,\text{mm}$ achieves slightly higher average purity overall (98.96% vs. 98.91%) with substantially lower computational cost. The total processing time at $6\,\text{mm}$ is 70% longer than at $8\,\text{mm}$ (7974.9 ms vs. 4677.8 ms), while providing only marginal performance differences. Based on this systematic evaluation, $v_{\text{size}} = 8\,\text{mm}$ and $k_{\text{proj}} = 3$ are adopted for all subsequent experiments as the best balance between accuracy and efficiency.

C. Comparison with Baseline Methods

To evaluate ShadowCorr against existing approaches, we compare it with three baselines spanning classical clustering and learning-based 3D instance segmentation methods.

MeanShift [46] is a classical non-parametric clustering algorithm applied directly to voxel centroids using projected spatial coordinates and confidence scores, with the same input representation as ShadowCorr but without co-occurrence information. This provides a geometry-only baseline without learned features. The bandwidth was selected as 0.2 as the best value from an empirical sweep.

PointGroup [26] is a learning-based 3D instance segmentation method that groups points using learned embeddings produced by a sparse 3D convolutional backbone with RGB and spatial features. The model was trained from scratch on the point cloud data using a single semantic class ("rock").

Mask3D [9] is a transformer-based 3D instance segmentation method that predicts instance masks directly from sparse 3D point cloud features using learned object queries and masked cross-attention. Built on a sparse convolutional backbone, it iteratively refines instance-aware query features across multiple scales and outputs all instance masks in parallel. It was trained under the same setting as PointGroup, using the point cloud data and a single semantic class ("rock").

All methods are evaluated according to how accurately they group rock segments belonging to the same instance. Because the compared methods operate on different underlying representations, ARI should be interpreted with care. MeanShift and ShadowCorr perform correspondence inference on projected volumetric voxels, and their ARI is therefore reported at the voxel level. PointGroup and Mask3D instead operate on the original point cloud, so their ARI is reported at the point level. Rock purity and cluster purity are computed consistently at the segment level for all methods, making them the most directly comparable metrics.

TABLE II
COMPARISON WITH BASELINE METHODS.

Method	Modality	Metrics (mean±std, %)				Runtime (ms)			
		ARI	Rock Purity	Cluster Purity	Avg Purity	Preprocess	Inference	Postprocess	Total
MeanShift [46]	Spatial + Confidence	43.62±5.15	82.22±4.62	85.37±5.28	83.80±3.69	2888.7	N/A	1151.4	4040.1
PointGroup [26]	Spatial + RGB	35.69±17.63	99.92±0.33	78.72±8.45	89.32±4.23	34.7	1371.1	17.1	1422.8
Mask3D [9]	Spatial + RGB	36.94±4.57	57.03±4.28	72.96±3.81	64.99±3.55	2178.1	222.4	6.0	2406.6
ShadowCorr (Ours)	Spatial + Confidence + Segment ID	**98.66±1.85**	**99.04±1.37**	**98.88±1.50**	**98.96±1.23**	3712.8	15.5	949.5	4677.8

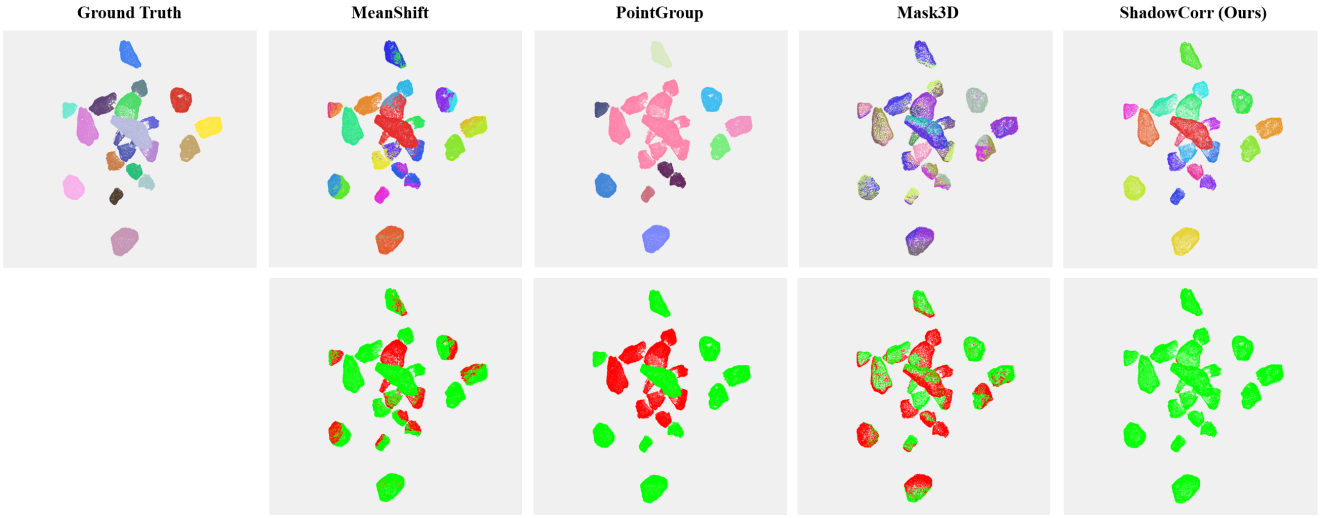

Fig. 9. Qualitative comparison of clustering results on a random test scene. The top row shows the predicted instance clusters for each method, and the bottom row visualizes correctness: correctly clustered segments are highlighted in green ■, while incorrectly clustered segments are highlighted in red ■.

As shown in Table II, ShadowCorr outperforms all baselines in correspondence quality on this task. MeanShift clustering on spatial centroids achieves 82.22% rock purity and 85.37% cluster purity, but relies only on geometric proximity without learned features. PointGroup and Mask3D perform poorly despite using RGB appearance and neural architectures. PointGroup achieves high rock purity (99.92%) but low cluster purity (78.72%), indicating severe under-segmentation: segments from multiple rocks are merged into one predicted cluster. Mask3D also performs poorly, with 57.03% rock purity and 72.96% cluster purity, showing that surface appearance is insufficient in severe-overlap scenes such as the one shown in Fig. 9. Both learning-based baselines also show much lower ARI (35.69% for PointGroup and 36.94% for Mask3D), indicating weak dense correspondence before segment aggregation.

In contrast, ShadowCorr achieves 99.04% rock purity and 98.88% cluster purity, showing that volumetric ray casting can establish correspondences even when visible surfaces do not overlap. Although PointGroup has the fastest total runtime (1,422.8 ms), ShadowCorr's total runtime (4,677.8 ms) is dominated by preprocessing and postprocessing rather than neural inference (15.5 ms); this preprocessing bottleneck is discussed further in Section V-F.

Most existing 3D instance segmentation methods, including PointGroup and Mask3D, were originally developed for benchmark settings such as ScanNet [29] and S3DIS [30], where observations are relatively complete and object boundaries are often visually distinctive. In lunar rock scenes, however, appearance is nearly uniform and observations are often partial, overlapping, or occluded. ShadowCorr instead incorporates confidence and cross-view co-occurrence cues, making it better suited for associating segments from the same rock.

D. Ablation Study

A series of ablation experiments was conducted to verify the effectiveness of the three key input cues: spatial coordinates, segment IDs, and confidence scores. As shown in Table III, spatial coordinates alone provide a strong baseline (97.95% rock purity and 95.70% cluster purity), confirming that geometric position is already highly informative. When confidence scores are added to spatial features, both rock purity and cluster purity improve. Among the tested additions, segment IDs provide the largest gain, especially in cluster purity. Pairing segment IDs with spatial features yields the highest cluster purity (98.94%, +3.24%), while the full model combining all three inputs achieves the best overall balance (99.04% rock purity and 98.88% cluster purity), showing that all three cues are complementary.

Voxel ARI is 92.14% with spatial features alone, rises slightly to 93.22% when confidence is added without segment IDs, and increases substantially once segment IDs are included (97.65% with spatial + segment IDs, 98.66% for the full

TABLE III
ABLATION STUDY.

Spatial	Confidence	Segment ID	Voxel ARI (%)	Rock Purity (%)	Cluster Purity (%)	Runtime (ms)
✓			92.14±5.10	97.95±1.91	95.70±3.56	4471.3
✓	✓		93.22±4.63	98.24±1.86	97.63±2.24	4472.0
✓		✓	97.65±3.08	98.32±1.79	98.94±1.48	4677.2
✓	✓	✓	98.66±1.85	99.04±1.37	98.88±1.50	4677.8

model). Thus co-occurrence cues drive most of the ARI gain, while confidence yields a further improvement once segment IDs are present.

This pattern matches the design motivation for the segment-ID embeddings. Voxels where multiple segments from the same rock instance project to the same location exhibit co-occurrence patterns that differ from random spatial overlap. The Word2Vec-style segment-ID embeddings capture these permutation-invariant patterns, enabling the network to distinguish true correspondences from geometric coincidence.

The incorporation of these additional features does not substantially increase runtime: segment embedding and confidence calculation add only $\approx 200\,\mathrm{ms}$ of overhead per scene, while the end-to-end runtime remains dominated by data preparation and clustering rather than the sparse CNN backbone ($15.5\,\mathrm{ms}$).

E. Failure Mode Analysis

Across 200 test scenes containing 3,975 ground-truth rocks, ShadowCorr achieves a perfect 1-to-1 instance match for 93.7% of all rocks, indicating that errors are concentrated in a relatively small set of difficult cases. The dominant failure types are over-segmentation and under-segmentation, consistent with trends reported in prior instance-segmentation work [9], [10], [26].

Over-segmentation, in which one rock is split into multiple predicted clusters, is the more common failure. It affects 118 rocks (3.0% of all instances), or 0.59 per scene on average. In nearly every case, the error is a simple pairwise split, and no rock is fragmented into more than three pieces. Under-segmentation, in which neighboring rocks are merged into one cluster, affects 105 predicted clusters (0.53 per scene) and is likewise almost always pairwise.

A more severe case is the *split-merge chain*, where a rock is first split and one of its fragments is then merged with a neighboring rock. This occurs in 64 scenes (0.32 per scene). Even in these cases, however, the dominant rock in the chain retains a mean cluster purity of 0.71. Missed detections account for only 35 instances across all scenes (0.17 per scene) and are strongly associated with small object size, with a mean of 917 points compared with a scene-wide average of roughly 1,500 points. Across all 200 scenes, the method produces no spurious clusters, indicating that it does not hallucinate phantom instances.

These three failure modes can be traced back to the same underlying coupling between the network's fixed receptive field and the fixed voxel size. Large rocks exceed the receptive field, causing over-segmentation; touching rocks fall within a single voxel boundary, causing under-segmentation; and the smallest rocks span too few voxels to be reliably detected. This coupling also has a runtime dimension: finer voxels, the most direct way to resolve small and touching instances, proportionally increase the cost of ray casting, voxel generation, and embedding computation, which already dominate total runtime. Section V-F discusses these trade-offs and the directions that could address them.

F. Limitations

ShadowCorr currently has two main limitations. First, the fixed voxel size and the network's fixed receptive field create a coupled scale trade-off. At $v_{\mathrm{size}} = 8\,\mathrm{mm}$, touching rocks separated by less than one voxel fall within a shared boundary that cannot be cleanly resolved. At the other end of the scale range, the smallest rocks occupy too few voxels for the sparse CNN and attention modules to form stable neighborhoods.

Using finer voxels would alleviate both problems, but it would also spread large rocks across more voxels than the receptive field can cover and would substantially increase memory and computation. In larger scenes, this may also exceed GPU memory and require patch-based tiling, which can in turn degrade correspondence near patch boundaries. Adaptive per-object voxel resolution, for example through an octree representation [51], is a natural direction for increasing resolution only where it is needed while controlling memory growth.

The receptive-field side of the trade-off remains unresolved. Wider convolutional kernels (e.g., $7 \times 7 \times 7$) did not solve the problem and instead reduced accuracy, likely due to overfitting. Future work should therefore investigate other ways to expand spatial context, including dilated convolutions [52], multi-scale feature aggregation [53], global attention mechanisms [39], and learnable clustering networks [54], [55] that adapt grouping behavior to local embedding density and feature variance.

Second, the preprocessing pipeline is computationally heavy. Ray casting, voxel generation, and segment-ID embedding computation dominate the total runtime ($4.68\,\mathrm{s}$), far exceeding neural network inference alone ($15.5\,\mathrm{ms}$) and limiting real-time deployment. This limitation is directly coupled to the voxel-resolution bottleneck above: finer voxels, which are the most direct way to improve separation of small or touching rocks, would also increase preprocessing cost proportionally. GPU-accelerated voxel generation and related preprocessing would therefore help address both limitations at once.

VI. CONCLUSION

This work presented ShadowCorr, a voxel-based framework that establishes multi-view segment correspondence through volumetric consensus rather than surface matching. By casting segments into a shared 3D grid and learning from spatial, confidence, and co-occurrence patterns, the method directly addresses the dual challenge in unstructured rocky environments: grouping fragments of the same object across viewpoints while correctly separating physically adjacent instances.

Experimental evaluation on test scenes shows substantial gains over all baselines, with confidence weighting and Word2Vec-style segment-ID embeddings each improving overall performance. The findings demonstrate that volumetric consensus provides an effective basis for correspondence in challenging settings where existing methods often struggle, such as cross-view matching under object contact, partial visibility, and weak texture cues. As autonomous systems are increasingly deployed in unstructured planetary and terrestrial environments, reasoning about object identity from incomplete multi-view observations is an essential capability. Although the present study is limited to synthetic data, real-world deployment is a natural next step, and the directions below outline the work required to enable it.

Beyond the engineering improvements discussed in Section V-F, several broader directions remain open. Two others target capability: a systematic study of reduced angular camera coverage (e.g., 180° or 90°) would quantify how performance degrades when full scene encirclement is not possible, while validation on real-world imagery, like terrestrial construction and demolition scenes or actual lunar surface data when available, would test whether the volumetric consensus principle transfers beyond the synthetic domain, especially when depth observations are noisy, upstream 2D segmentation masks are imperfect, and camera poses are uncertain rather than known exactly. Two others target capability: extending the framework to multi-class panoptic segmentation is a natural next step, since when upstream 2D segmentation provides semantic labels alongside instance masks, each voxel can carry an additional class label with only a lightweight classification head added to the current framework. Adapting ShadowCorr to streaming inputs from a moving camera, via incremental voxel map updates and online re-identification of displaced fragments, would broaden applicability to disaster-response robotics, where the scene evolves as rescue operations proceed.

ACKNOWLEDGMENT

The authors would like to thank Dr. Creed F. Jones for his advice and contributions to algorithm development, and Dr. John Cooper for his valuable guidance and feedback.

REFERENCES

[1] A. Akacha, "Ruins of city," Pexels, n.d., accessed: 2026-01-26. [Online]. Available: https://www.pexels.com/photo/ruins-of-city-15803717/

[2] Lunar and Planetary Institute, "Apollo Image Atlas: AS15-82-11147," 2026, catalog entry for Apollo 15 Hasselblad frame AS15-82-11147. Accessed: 2026-01-26. [Online]. Available: https://www.lpi.usra.edu/resources/apollo/frame/?AS15-82-11147

[3] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (FPFH) for 3D registration," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2009, pp. 3212–3217.

[4] F. Tombari, S. Salti, and L. Di Stefano, "Unique signatures of histograms for local surface description," in *Computer Vision – ECCV 2010*, ser. Lecture Notes in Computer Science, vol. 6313, 2010, pp. 356–369.

[5] A. Zeng, S. Song, M. Nießner, M. Fisher, J. Xiao, and T. Funkhouser, "3DMatch: Learning local geometric descriptors from RGB-D reconstructions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1802–1811.

[6] C. B. Choy, J. Park, and V. Koltun, "Fully convolutional geometric features," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 8957–8965.

[7] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *IEEE International Conference on Image Processing (ICIP)*, 2016, pp. 3464–3468.

[8] Y. Zhang, P. Sun, Y. Jiang, D. Yu, F. Weng, Z. Yuan, P. Luo, W. Liu, and X. Wang, "ByteTrack: Multi-object tracking by associating every detection box," in *European Conference on Computer Vision (ECCV)*, 2022, pp. 1–21.

[9] J. Schult, F. Engelmann, A. Hermans, O. Litany, S. Tang, and B. Leibe, "Mask3D: Mask transformer for 3D semantic instance segmentation," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 8216–8223.

[10] T. Vu, K. Kim, T. M. Luu, T. Nguyen, and C. D. Yoo, "SoftGroup for 3D instance segmentation on point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 2708–2717.

[11] N. Ravi *et al.*, "SAM 2: Segment anything in images and videos," *arXiv preprint arXiv:2408.00714*, 2024.

[12] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2961–2969.

[13] R. Khanam and M. Hussain, "YOLOv11: An overview of the key architectural enhancements," 2024. [Online]. Available: https://arxiv.org/abs/2410.17725

[14] D. Scaramuzza and F. Fraundorfer, "Visual odometry [tutorial]," *IEEE Robotics & Automation Magazine*, vol. 18, no. 4, pp. 80–92, 2011.

[15] T. Qin, P. Li, and S. Shen, "VINS-Mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[16] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: A versatile and accurate monocular SLAM system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[17] P. J. Besl and N. D. McKay, "A method for registration of 3-d shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[18] Epic Games, "Unreal engine 5.5 release notes," 2024, accessed: 2026-03-17. [Online]. Available: https://dev.epicgames.com/documentation/en-us/unreal-engine/unreal-engine-5-5-release-notes

[19] NASA Astromaterials Research and Exploration Science, "Astromaterials 3D," n.d., accessed: 2026-01-12. [Online]. Available: https://ares.jsc.nasa.gov/astromaterials3d/index.htm

[20] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 26, 2013, pp. 3111–3119.

[21] Y. Ruan, "Unreal engine multi-view RGB-D lunar rock dataset for 3D segment correspondence in complex scenes," 2026, dataset. [Online]. Available: https://doi.org/10.5281/zenodo.18917286

[22] A. Laurentini, "The visual hull concept for silhouette-based image understanding," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 2, pp. 150–162, 1994.

[23] K. N. Kutulakos and S. M. Seitz, "A theory of shape by space carving," *International Journal of Computer Vision*, vol. 38, no. 3, pp. 199–218, 2000.

[24] J. Hou, A. Dai, and M. Nießner, "3D-SIS: 3D semantic instance segmentation of RGB-D scans," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 4421–4430.

[25] F. Engelmann, M. Bokeloh, A. Fathi, B. Leibe, and M. Nießner, "3D-MPA: Multi-proposal aggregation for 3D semantic instance segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 9031–9040.

[26] L. Jiang, H. Zhao, S. Shi, S. Liu, C.-W. Fu, and J. Jia, "PointGroup: Dual-set point grouping for 3D instance segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 4867–4876.

[27] S. Chen, J. Fang, Q. Zhang, W. Liu, and X. Wang, "Hierarchical aggregation for 3D instance segmentation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 15 467–15 476.

[28] J. Sun, C. Qing, J. Tan, and X. Xu, "Superpoint transformer for 3D scene instance segmentation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 2, 2023, pp. 2393–2401.

[29] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "ScanNet: Richly-annotated 3D reconstructions of indoor scenes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 5828–5839.

[30] I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. Brilakis, M. Fischer, and S. Savarese, "3D semantic parsing of large-scale indoor spaces," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 1534–1543.

[31] Z. Yan, Z. Yi, R. Hu, N. J. Mitra, D. Cohen-Or, and H. Huang, "Consistent two-flow network for tele-registration of point clouds," *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 12, pp. 4304–4318, 2022.

[32] Y. Liu and Z. Liu, "Low-overlapping point cloud registration using mutual prior-based completion network," *IEEE Transactions on Image Processing*, vol. 33, pp. 4781–4795, 2024.

[33] H.-Y. Liu, J.-W. Guo, H.-Y. Jiang, Y.-C. Liu, X.-P. Zhang, and D.-M. Yan, "PuzzleNet: Boundary-aware feature matching for non-overlapping 3D point clouds assembly," *Journal of Computer Science and Technology*, vol. 38, no. 3, pp. 492–509, 2023.

[34] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," *arXiv preprint arXiv:1801.09847*, 2018. [Online]. Available: https://arxiv.org/abs/1801.09847

[35] T. W. Anderson, *An Introduction to Multivariate Statistical Analysis*, 3rd ed. Wiley, 2003.

[36] A. van den Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

[37] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan, "Supervised contrastive learning," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 18 661–18 673.

[38] H. Tang, Z. Liu, X. Li, Y. Lin, and S. Han, "TorchSparse: Efficient point cloud inference engine," in *Proceedings of Machine Learning and Systems (MLSys)*, 2022. [Online]. Available: https://proceedings.mlsys.org/paper_files/paper/2022/file/c48e820389ae2420c1ad9d5856e1e41c-Paper.pdf

[39] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017.

[40] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning (ICML)*, 2015, pp. 448–456.

[41] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *International Conference on Machine Learning (ICML)*, 2010, pp. 807–814.

[42] B. De Brabandere, D. Neven, and L. Van Gool, "Semantic instance segmentation with a discriminative loss function," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2017, pp. 7–15.

[43] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017.

[44] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph CNN for learning on point clouds," *ACM Trans. Graph.*, vol. 38, no. 5, pp. 146:1–146:12, 2019.

[45] J. Weston, F. Ratle, H. Mobahi, and R. Collobert, "Deep learning via semi-supervised embedding," in *Neural Networks: Tricks of the Trade*, 2nd ed., ser. Lecture Notes in Computer Science, G. Montavon, G. B. Orr, and K.-R. Müller, Eds. Berlin, Heidelberg: Springer, 2012, vol. 7700, pp. 639–655.

[46] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 603–619, 2002.

[47] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "PyTorch: An imperative style, high-performance deep learning library," *Advances in Neural Information Processing Systems*, vol. 32, pp. 8026–8037, 2019.

[48] P. Micikevicius, S. Narang, J. Alben, G. Diamos, E. Elsen, D. Garcia, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh, and H. Wu, "Mixed precision training," in *International Conference on Learning Representations (ICLR)*, 2018.

[49] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, 2015.

[50] L. Hubert and P. Arabie, "Comparing partitions," *Journal of Classification*, vol. 2, no. 1, pp. 193–218, 1985.

[51] D. Meagher, "Geometric modeling using octree encoding," *Computer Graphics and Image Processing*, vol. 19, no. 2, pp. 129–147, 1982.

[52] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," in *International Conference on Learning Representations (ICLR)*, 2016.

[53] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2881–2890.

[54] B. Gao, Y. Yang, H. Gouk, and T. M. Hospedales, "Deep clustering with concrete k-means," in *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 4252–4256.

[55] L. Yang, D. Chen, X. Zhan, R. Zhao, C. C. Loy, and D. Lin, "Learning to cluster faces via confidence and connectivity estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 13 369–13 378.

Yiyan Ruan received the B.S. degree in mechanical engineering from Virginia Polytechnic Institute and State University, Blacksburg, VA, USA, where he is currently pursuing the Ph.D. degree in mechanical engineering. His research interests include multi-axis additive manufacturing, 3D scene understanding, robotic automation, and robotic perception for planetary exploration.

Erik Komendera is an Assistant Professor of Mechanical Engineering at Virginia Polytechnic Institute and State University in Blacksburg, VA. He earned his Ph.D. from the University of Colorado in Boulder, CO. He was previously a researcher at NASA Langley Research Center in Hampton, VA. His research interests include robotic perception, Bayesian estimation, operations planning, autonomous construction, in-space assembly and servicing, and robot-enhanced additive and subtractive manufacturing. His current and previous collaborators include NASA, DOD, NSF, and space industry.