From Projection to Perception: A Mathematical Exploration of Shadow-based Neural Reconstruction

A research report submitted to the Scientific Committee of the Hang Lung Mathematics Award

Team Number
2596873

Team Members
Wong Yuk To, Hung Kwong Lam
Cheung Tsz Lung, Chan Ngo Tin, Zhou Lam Ho

Teacher
Mr. Chan Ping Ho

School
Po Leung Kuk Celine Ho Yam Tong College

Date
July 11, 2025

Abstract

This paper explores SHADOWNEUS [LWX23], a neural network that reconstructs 3D geometry from single-view camera images using shadow and light cues. Unlike traditional 3D reconstruction methods relying on multi-view cameras or sensors, SHADOWNEUS leverages a neural signed distance field (SDF) for accurate 3D geometry reconstruction. We analyze the training process and uncover its connections to projective geometry, spatial reasoning in $\mathbb{R}^3$, and the neural network's learned geometric representation of space.

Contents

1 Background **2**

1.1 What is 3D Reconstruction from Images? 2

1.2 Information Encoded in 2D Images 2

1.3 The Forward Projection: From 3D World to 2D Image 3

1.4 The Inverse Problem: From 2D Image to 3D World 4

1.5 Cues for Solving the Inverse Problem 5

2 Shadows as a Geometric Constraint **5**

2.1 Light Ray and Shadow Geometry 5

2.2 Shadow Boundary and Surface Partitioning 5

2.3 Cast Shadow Regions 6

2.4 Shadow Regions in Images 6

2.5 Shadows as Cues for 3D Reconstruction 7

3 ShadowNeuS: Neural Shadow-Based 3D Reconstruction **9**

3.1 Classical vs Neural Approaches: Method Comparison 9

3.2 From Classical Shadow Analysis to Neural Methods 10

3.2.1 ShadowNeuS Pipeline 10

3.2.2 Addressing Classical Limitations 11

3.2.3 Shadow Consistency Loss 12

1 Background

1.1 What is 3D Reconstruction from Images?

The goal of 3D reconstruction is to recover the structure of a 3D scene using only 2D images.

Definition 1.1.1 (3D Scene Representation)**.**
A 3D scene is represented by a set of points $\boldsymbol{P} = [P_x, P_y, P_z]^\top \in \mathbb{R}^3$ in Euclidean space.

Definition 1.1.2 (Image Projection)**.**
Each image I_n of the 3D scene records a set of pixel coordinates $\boldsymbol{p} = [p_x, p_y]^\top \in \mathbb{R}^2$.

The process of capturing a 3D point in a 2D image I_n can be modeled as a projection function π_n:

$$\boxed{\pi_n : \mathbb{R}^3 \to \mathbb{R}^2, \quad [P_x, P_y, P_z]^\top \mapsto [p_x, p_y]^\top} \tag{1}$$

This projection function represents how a camera maps a 3D point to a 2D pixel in the n-th image. To reconstruct the 3D scene, we need to solve the **inverse problem π_n^{-1}**:

$$\boxed{\pi_n^{-1}(\boldsymbol{p}) = \left\{\boldsymbol{P} \in \mathbb{R}^3 \mid \pi_n(\boldsymbol{P}) = \boldsymbol{p}\right\}} \tag{2}$$

However, this inverse problem is typically **ill-posed**, as multiple 3D points may project to the same 2D pixel, leading to ambiguity. We will detail this in Section 1.4.

1.2 Information Encoded in 2D Images

A 2D image I_n can provide multiple types of information encoded as mathematical structures:

Information Available from an Image:

(i) **Pixel coordinates**: $\boldsymbol{p} = [p_x, p_y]^\top \in \mathbb{R}^2$, represents the spatial location of each pixel

(ii) **Color values**: $C_n(\boldsymbol{p}) = [r, g, b]^\top \in [0,1]^3$, represents the RGB tristimulus values

(iii) **RGB gradient matrix**:

$$\nabla C_n(\boldsymbol{p}) = \begin{bmatrix} \frac{\partial r}{\partial p_x} & \frac{\partial r}{\partial p_y} \\ \frac{\partial g}{\partial p_x} & \frac{\partial g}{\partial p_y} \\ \frac{\partial b}{\partial p_x} & \frac{\partial b}{\partial p_y} \end{bmatrix} \in \mathbb{R}^{3\times 2} \tag{3}$$

This Jacobian matrix captures local intensity variations, indicating edges or texture information.

(iv) **Learned feature embedding**: $\phi(I_n)(\boldsymbol{p}) \in \mathbb{R}^d$, represents high-dimensional features extracted via neural networks like CNNs

These data structures result from projecting 3D geometry through camera optics, where $C_n(\boldsymbol{p})$ corresponds to visible surface reflectance and $\nabla C_n(\boldsymbol{p})$ encodes geometric boundaries.

1.3 The Forward Projection: From 3D World to 2D Image

We formalize the perspective projection process using homogeneous coordinates and transformation matrices.

Camera Parameter Matrices:

Definition 1.3.1 (Extrinsic Parameters)**.**
The world-to-camera transformation is characterized by:

$$\text{Camera center:} \quad \boldsymbol{C} = [C_x, C_y, C_z]^\mathsf{T} \in \mathbb{R}^3 \tag{4}$$

$$\text{Rotation matrix:} \quad R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \in \mathrm{SO}(3) \tag{5}$$

$$\text{Translation vector:} \quad \boldsymbol{t} = -R\boldsymbol{C} \in \mathbb{R}^3 \tag{6}$$

Definition 1.3.2 (Intrinsic Parameters)**.**
The camera's internal geometry is encoded by:

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{3\times 3} \tag{7}$$

where (f_x, f_y) are focal lengths in pixels and (c_x, c_y) is the principal point.

Forward Projection Pipeline:

Proposition 1.3.1 (Perspective Projection Transform)**.**
The complete forward projection involves three sequential transformations:

Step 1: World to camera coordinates

$$\boldsymbol{P}_{\text{cam}} = R\boldsymbol{P} + \boldsymbol{t} \tag{8}$$

Step 2: Camera to image coordinates

$$\boldsymbol{P}_{\text{hom}} = K\boldsymbol{P}_{\text{cam}} = \begin{bmatrix} p'_x \\ p'_y \\ z' \end{bmatrix} \tag{9}$$

Step 3: Perspective division

$$\boldsymbol{p} = \begin{bmatrix} p_x \\ p_y \end{bmatrix} = \frac{1}{z'} \begin{bmatrix} p'_x \\ p'_y \end{bmatrix}, \quad z' \neq 0 \tag{10}$$

The complete transformation matrix can be expressed as:

$$\boxed{\boldsymbol{P}_{\text{hom}} = K[R \mid \boldsymbol{t}] \begin{bmatrix} \boldsymbol{P} \\ 1 \end{bmatrix}, \quad \boldsymbol{p} = \frac{1}{z'} \begin{bmatrix} p'_x \\ p'_y \end{bmatrix}} \tag{11}$$

1.4 The Inverse Problem: From 2D Image to 3D World

We now tackle the fundamental challenge of inverting the projection function.

Lemma 1.4.1 (Camera Ray Parametrization)**.**
Given a pixel $\boldsymbol{p} = [p_x, p_y]^\mathsf{T}$ and camera parameters $(K, R, \boldsymbol{C})$, the corresponding 3D points form a ray:

$$\boxed{\boldsymbol{P}(\lambda) = \boldsymbol{C} + \lambda \cdot \boldsymbol{d}, \quad \lambda > 0} \tag{12}$$

where the ray direction is:

$$\boxed{\boldsymbol{d} = R^{-1}K^{-1}\begin{bmatrix} p_x \\ p_y \\ 1 \end{bmatrix}} \tag{13}$$

Remark 1.4.1 (Normalization)**.**
The direction vector $\boldsymbol{d}$ can optionally be normalized to unit length for physical ray tracing but not strictly necessary for the ray parametrization.

Proof. Starting from the forward projection equation (11):

$$K(R\boldsymbol{P} + \boldsymbol{t}) = z'\begin{bmatrix} p_x \\ p_y \\ 1 \end{bmatrix} \tag{14}$$

$$R\boldsymbol{P} + \boldsymbol{t} = z'K^{-1}\begin{bmatrix} p_x \\ p_y \\ 1 \end{bmatrix} \tag{15}$$

$$\boldsymbol{P} = R^{-1}\left(z'K^{-1}\begin{bmatrix} p_x \\ p_y \\ 1 \end{bmatrix} - \boldsymbol{t}\right) \tag{16}$$

Since $\boldsymbol{t} = -R\boldsymbol{C}$, we have $-R^{-1}\boldsymbol{t} = \boldsymbol{C}$. Setting $\lambda = z'$:

$$\boldsymbol{P}(\lambda) = \boldsymbol{C} + \lambda \cdot R^{-1}K^{-1}\begin{bmatrix} p_x \\ p_y \\ 1 \end{bmatrix} \tag{17}$$

$\square$

Proposition 1.4.1 (Ill-posed Nature of Single-View Reconstruction)**.**
The inverse projection problem is fundamentally **ill-posed** because:

(a) The depth parameter λ is undetermined

(b) Each pixel $\boldsymbol{p}$ defines a ray of infinitely many possible 3D points

(c) Additional constraints are required for unique reconstruction

1.5 Cues for Solving the Inverse Problem

To achieve unique reconstruction, we require additional information such as:

- **Stereo correspondence**: Multiple viewpoints providing triangulation
- **Depth sensors**: Direct measurement of λ
- **Shadow constraints**: Geometric relationships via light ray intersections

2 Shadows as a Geometric Constraint

We formalize how shadows encode geometric information to constrain 3D reconstruction. By analyzing light transport and occlusion, we derive conditions enabling depth recovery from single-view images, leveraging shadows as powerful geometric cues.

2.1 Light Ray and Shadow Geometry

Definition 2.1.1 (Light Ray)**.**
Given a point light source $\boldsymbol{L} \in \mathbb{R}^3$ and a surface point $\boldsymbol{P} \in \mathbb{R}^3$, the light ray is:

$$\boxed{r(t) = \boldsymbol{L} + t(\boldsymbol{P} - \boldsymbol{L}), \quad t \in [0,1]} \tag{18}$$

Definition 2.1.2 (Shadow Occlusion Test)**.**
A point $\boldsymbol{P}$ is in shadow if there exists $t \in (0,1)$ such that the light ray intersects a surface $\mathscr{S}$:

$$\boxed{r(t) \cap \mathscr{S} \neq \emptyset, \quad t \in (0,1)} \tag{19}$$

Remark 2.1.1 (Physical Interpretation)**.**
The interval $(0,1)$ excludes the light source ($t = 0$) and the target point ($t = 1$), ensuring the test checks for obstructions between $\boldsymbol{L}$ and $\boldsymbol{P}$. This models physical light transport where occlusion by another surface causes a shadow.

2.2 Shadow Boundary and Surface Partitioning

Theorem 2.2.1 (Tangency Condition)**.**
A point $\boldsymbol{Q} \in \mathscr{S}$ lies on the shadow boundary if and only if the light direction is tangent to the surface:

$$\boxed{(\boldsymbol{Q} - \boldsymbol{L}) \cdot \boldsymbol{n}(\boldsymbol{Q}) = 0} \tag{20}$$

where $\boldsymbol{n}(\boldsymbol{Q})$ is the outward unit normal at $\boldsymbol{Q}$.

Proof. At the shadow boundary, the light ray $r(t)$ grazes the surface $\mathscr{S}$ at $\boldsymbol{Q}$, meaning the direction $\boldsymbol{Q} - \boldsymbol{L}$ lies in the tangent plane. Thus, it is perpendicular to the surface normal $\boldsymbol{n}(\boldsymbol{Q})$, yielding $(\boldsymbol{Q} - \boldsymbol{L}) \cdot \boldsymbol{n}(\boldsymbol{Q}) = 0$. □

Definition 2.2.1 (Shadow Boundary Set)**.**
The 3D shadow boundary is:

$$\boxed{\mathscr{B} = \{\boldsymbol{Q} \in \mathscr{S} \mid (\boldsymbol{Q} - \boldsymbol{L}) \cdot \boldsymbol{n}(\boldsymbol{Q}) = 0\}} \tag{21}$$

Proposition 2.2.1 (Surface Illumination Partition).
The surface $\mathscr{S}$ is partitioned into three disjoint regions based on the light direction:

$$\mathscr{S}_{\text{lit}} = \{\boldsymbol{P} \in \mathscr{S} \mid (\boldsymbol{P} - \boldsymbol{L}) \cdot \boldsymbol{n}(\boldsymbol{P}) > 0\} \qquad \text{(illuminated)} \tag{22}$$
$$\mathscr{S}_{\text{shadow}} = \{\boldsymbol{P} \in \mathscr{S} \mid (\boldsymbol{P} - \boldsymbol{L}) \cdot \boldsymbol{n}(\boldsymbol{P}) < 0\} \qquad \text{(attached shadow)} \tag{23}$$
$$\mathscr{B} = \{\boldsymbol{P} \in \mathscr{S} \mid (\boldsymbol{P} - \boldsymbol{L}) \cdot \boldsymbol{n}(\boldsymbol{P}) = 0\} \qquad \text{(shadow boundary)} \tag{24}$$

Remark 2.2.1 (Geometric Interpretation).
The sign of $(\boldsymbol{P} - \boldsymbol{L}) \cdot \boldsymbol{n}(\boldsymbol{P})$ reflects the angle between the light direction and the surface normal, determining whether a point is lit, shadowed, or on the boundary.

2.3 Cast Shadow Regions

Definition 2.3.1 (Cast Shadow Region).
For an occluding surface $\mathscr{S}_1$ and a receiving surface $\mathscr{S}_2$, the cast shadow region is:

$$\boxed{\mathscr{C}_{1\to 2} = \{\boldsymbol{P} \in \mathscr{S}_2 \mid \exists t \in (0,1) : \boldsymbol{L} + t(\boldsymbol{P} - \boldsymbol{L}) \in \mathscr{S}_1\}} \tag{25}$$

Remark 2.3.1 (Cast Shadow Formation).
A point $\boldsymbol{P} \in \mathscr{S}_2$ is in the cast shadow if the light ray from $\boldsymbol{L}$ to $\boldsymbol{P}$ is blocked by $\mathscr{S}_1$. For multiple occluders $\mathscr{S}_1, \mathscr{S}_2, \ldots, \mathscr{S}_k$, a point is shadowed if it lies in $\bigcup_{i=1}^{k} \mathscr{C}_{i\to\text{target}}$. Self-shadowing occurs when $\mathscr{S}_1 = \mathscr{S}_2$, as a surface may occlude itself.

Definition 2.3.2 (Cast Shadow Boundary).
The cast shadow boundary on $\mathscr{S}_2$ is:

$$\boxed{\partial\mathscr{C}_{1\to 2} = \{\boldsymbol{P} \in \mathscr{S}_2 \mid \boldsymbol{P} = \boldsymbol{Q} + s(\boldsymbol{Q} - \boldsymbol{L}), \boldsymbol{Q} \in \mathscr{B}_1, s > 0\}} \tag{26}$$

where $\mathscr{B}_1$ is the shadow boundary on $\mathscr{S}_1$ (Equation (21)).

Proposition 2.3.1 (Multi-Surface Cast Shadows and Boundaries).
For k occluders, the total cast shadow region and its boundary on a target surface are:

$$\boxed{\mathscr{C}_{\text{total}} = \bigcup_{i=1}^{k} \mathscr{C}_{i\to\text{target}}} \tag{27}$$

$$\boxed{\partial\mathscr{C}_{\text{total}} = \bigcup_{i=1}^{k} \partial\mathscr{C}_{i\to\text{target}}} \tag{28}$$

where $\mathscr{C}_{i\to\text{target}}$ is the cast shadow region from $\mathscr{S}_i$ and $\partial\mathscr{C}_{i\to\text{target}}$ is its boundary.

2.4 Shadow Regions in Images

Definition 2.4.1 (Image Shadow and Lit Regions).
Given the projection function $\pi : \mathbb{R}^3 \to \mathbb{R}^2$ (Equation (1)), the image I partitions into:

$$\Omega_{\text{lit}}^{\text{obs}} = \{\pi(\boldsymbol{P}) \mid \boldsymbol{P} \in \mathscr{S}_{\text{lit}}\} \qquad \text{(lit region)} \tag{29}$$
$$\Omega_{\text{shadow}}^{\text{obs}} = \{\pi(\boldsymbol{P}) \mid \boldsymbol{P} \in \mathscr{S}_{\text{shadow}} \cup \mathscr{C}_{\text{total}}\} \qquad \text{(shadow region)} \tag{30}$$
$$\partial\Omega_{\text{shadow}}^{\text{obs}} = \{\pi(\boldsymbol{P}) \mid \boldsymbol{P} \in \mathscr{B} \cup \partial\mathscr{C}_{\text{total}}\} \qquad \text{(shadow boundary)} \tag{31}$$

where $\mathscr{S}_{\text{lit}}$, $\mathscr{S}_{\text{shadow}}$, $\mathscr{B}$, and $\mathscr{C}_{\text{total}}$ are defined in Equations (22), (23), (24), and (27).

Remark 2.4.1 (Relation between shadow region and image region)**.**
The forward projection π maps 3D surface points to 2D image pixels. For example, a 3D point $\boldsymbol{P}$ on a lit surface projects to pixel $\boldsymbol{p} = \pi(\boldsymbol{P})$ in the bright image region $\Omega_{\text{lit}}^{\text{obs}}$. Similarly, shadowed 3D points project to dark pixels in $\Omega_{\text{shadow}}^{\text{obs}}$. This creates a direct correspondence between 3D geometry and what we observe in the image.

Definition 2.4.2 (Shadow Consistency Loss)**.**
The shadow consistency loss measures how well our reconstructed 3D geometry matches the observed shadows:

$$\mathscr{L}_{\text{shadow}} = w_1 \cdot \mathscr{L}_{\text{lit-mismatch}} + w_2 \cdot \mathscr{L}_{\text{shadow-mismatch}} + w_3 \cdot \mathscr{L}_{\text{boundary-mismatch}} \tag{32}$$

where $\mathscr{L}_{\text{lit-mismatch}}$, $\mathscr{L}_{\text{shadow-mismatch}}$, and $\mathscr{L}_{\text{boundary-mismatch}}$ penalize pixels that are misclassified as lit, shadow, or boundary regions respectively, and w_1, w_2, w_3 are weighting coefficients.

Remark 2.4.2 (Purpose of Shadow Consistency Loss)**.**
This loss acts as a geometric constraint: if we observe a shadow at pixel $\boldsymbol{p}$, then the 3D point $\pi^{-1}(\boldsymbol{p})$ must be occluded by some surface. The loss penalizes inconsistencies, forcing the reconstruction to respect shadow boundaries and create geometrically reasonable surfaces.

2.5 Shadows as Cues for 3D Reconstruction

Geometric Information Encoded in Shadows

Shadows provide critical geometric constraints for single-view 3D reconstruction, leveraging light-surface interactions.

Proposition 2.5.1 (Shadow-Derived Geometric Cues)**.**
From observed shadows, we can infer:

(i) **Surface Orientation:** Points in attached shadows satisfy

$$(\boldsymbol{P} - \boldsymbol{L}) \cdot \boldsymbol{n}(\boldsymbol{P}) < 0$$

as in Equation (23), indicating that the surface at $\boldsymbol{P}$ is facing away from the light source.

(ii) **Tangency Constraints at Boundaries:** For points on the shadow boundary, the incoming light direction lies in the surface's tangent plane:

$$(\boldsymbol{Q} - \boldsymbol{L}) \cdot \boldsymbol{n}(\boldsymbol{Q}) = 0$$

(Equation (20)). This geometric constraint localizes the transition between lit and shadowed regions.

(iii) **Relative Depth from Cast Shadows:** If $\boldsymbol{P} \in \mathscr{C}_{1\to 2}$ (cast shadow region), then the occluding surface $\mathscr{S}_1$ must be geometrically in front of $\mathscr{S}_2$ along the light direction, as defined in Equation (25). This gives ordinal depth information.

(iv) **Occlusion Structure:** The presence of a shadow implies the existence of an occluder blocking the light ray from $\boldsymbol{L}$ to $\boldsymbol{P}$, indicating some intermediate geometry between the light and the shadowed surface point.

Remark 2.5.1 (Depth Ordering via Cast Shadows)**.**
Cast shadows provide explicit ordering information. If a point $\boldsymbol{P}$ on surface $\mathscr{S}_2$ is shadowed due to surface $\mathscr{S}_1$, then $\mathscr{S}_1$ must lie between the light source $\boldsymbol{L}$ and $\boldsymbol{P}$ along the ray. This insight can be directly encoded into optimization or inference schemes.

Shadow Boundary Depth Recovery

For a pixel $\boldsymbol{p}$ on the shadow boundary, depth recovery combines the camera ray with the tangency condition:

$$\boxed{\begin{aligned} \boldsymbol{P}(\lambda) &= \boldsymbol{C} + \lambda \boldsymbol{d} \\ (\boldsymbol{P}(\lambda) - \boldsymbol{L}) \cdot \boldsymbol{n}(\boldsymbol{P}(\lambda)) &= 0 \end{aligned}} \tag{33}$$

Theorem 2.5.1 (Depth from Shadow Boundary)**.**
Given surface normal estimate $\hat{\boldsymbol{n}}$, the depth parameter solves to:

$$\boxed{\lambda = \frac{(\boldsymbol{L} - \boldsymbol{C}) \cdot \hat{\boldsymbol{n}}}{\boldsymbol{d} \cdot \hat{\boldsymbol{n}}}} \tag{34}$$

provided $\boldsymbol{d} \cdot \hat{\boldsymbol{n}} \neq 0$.

Remark 2.5.2 (Multiple Boundaries)**.**
For m boundary pixels $\{\boldsymbol{p}_i\}$ with normals $\{\hat{\boldsymbol{n}}_i\}$:

$$\boxed{\lambda_i = \frac{(\boldsymbol{L} - \boldsymbol{C}) \cdot \hat{\boldsymbol{n}}_i}{\boldsymbol{d}_i \cdot \hat{\boldsymbol{n}}_i}, \quad i = 1, \dots, m} \tag{35}$$

Limitations and Computational Challenges

Fundamental Challenges

1. **Circular Dependency:** Equation (34) requires surface normals $\hat{\boldsymbol{n}}$, which depend on unknown surface geometry.

2. **Singular Cases:** Solution undefined when $\boldsymbol{d} \cdot \hat{\boldsymbol{n}} = 0$ (grazing angles).

3. **Non-Unique Solutions:** Multiple depth values may satisfy shadow conditions for complex surfaces.

Computational Challenges

4. **Non-Differentiable Occlusion:** The condition $r(t) \cap \mathscr{S} \neq \emptyset$ prevents gradient-based optimization. This discrete intersection test returns binary values (occluded/not occluded) with undefined gradients at surface boundaries, making automatic differentiation impossible.

5. **Nonlinear Systems:** Equation (33) requires iterative solvers with convergence issues.

6. **Ray Tracing Complexity:** Cast shadow computation scales as $O(n^2)$ for n surface points. Each of the n points requires testing occlusion against all other $n-1$ potential occluders, yielding $n(n-1) = O(n^2)$ ray-surface intersection tests.

Practical Issues

7. **Light Localization:** Unknown light position $\boldsymbol{L}$ must be estimated, introducing error propagation.

8. **Shadow Detection:** Real images have soft shadows, noise, and ambient lighting that blur boundaries.

9. **Multi-Object Complexity:** Overlapping shadows from multiple objects complicate the total cast region $\mathscr{C}_{\text{total}}$.

Remark 2.5.3 (Neural Solutions)**.**
Modern neural approaches address classical limitations through the following key advances:

Challenge	**Classical Limitation**	**Neural Solution**
Differentiability	Binary occlusion tests block gradient flow	Differentiable functions enable backpropagation through entire pipeline
Circular Dependencies	Needs normals to compute depth, needs depth to compute normals	Implicit surfaces (SDF/NeRF) represent geometry and normals jointly
Computational Efficiency	$O(n^2)$ ray tracing for cast shadows	Parallel GPU processing, learned approximations
Robustness	Fails on soft shadows, noise, complex lighting	Handles real-world imperfections through learned representations
Integration	Separate modules for lighting, geometry, materials	Unified model learns all components simultaneously

These methods transform geometric constraints into trainable optimization objectives, bridging classical shadow analysis with neural reconstruction.

3 ShadowNeuS: Neural Shadow-Based 3D Reconstruction

ShadowNeuS addresses single-view 3D reconstruction under specific controlled conditions:

- **Single camera viewpoint**: Fixed camera position and orientation
- **Multiple lighting conditions**: Images captured under various known light source positions $\{\mathbf{L}_i\}$
- **Known light positions**: Light source locations are calibrated or estimated
- **Static scene**: No object movement between lighting conditions
- **Observable shadows**: Clear shadow boundaries and regions in captured images

This setup enables ShadowNeuS to use shadow information to recover complete 3D geometry, including occluded regions, by combining classical geometric insights with neural optimization.

3.1 Classical vs Neural Approaches: Method Comparison

Figures 1 and 2 illustrate the processing pipelines for classical geometric methods and ShadowNeuS respectively.

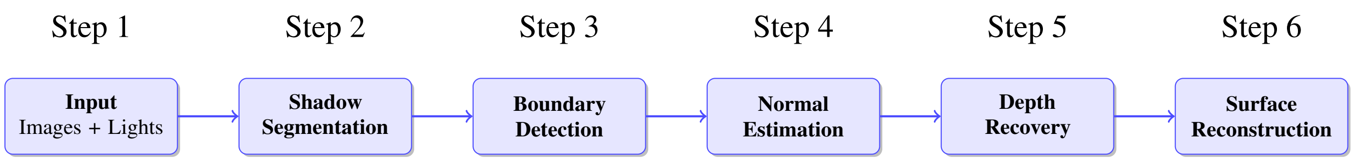

Figure 1: Classical method (Sequential processing pipeline)

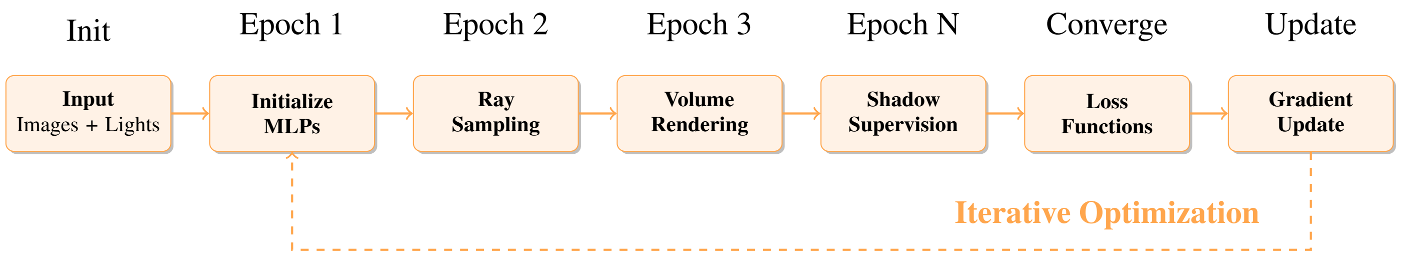

Figure 2: ShadowNeuS (End-to-end neural optimization pipeline)

3.2 From Classical Shadow Analysis to Neural Methods

Classical shadow-based 3D reconstruction, as depicted in Figure 1, relies on a sequential pipeline: segmenting shadows, detecting boundaries, estimating normals, solving for depth, and reconstructing surfaces. This approach faces significant limitations, as outlined in Section 2.5, including non-differentiable occlusion tests, circular dependencies between normals and depth, and computational complexity for multi-surface scenes.

ShadowNeuS, illustrated in Figure 2, overcomes these challenges by representing the 3D scene as a neural Signed Distance Function (SDF) parameterized by a Multi-Layer Perceptron (MLP). It uses differentiable ray tracing and optimization to jointly learn geometry and enforce shadow consistency, leveraging multiple lighting conditions to resolve ambiguities in single-view reconstruction.

Definition 3.2.1 (Neural Signed Distance Function)**.** A neural SDF is a function $f(\mathbf{x};\theta):\mathbb{R}^3\to\mathbb{R}$, parameterized by an MLP with weights θ, where:

- $f(\mathbf{x})=0$: Defines the surface $\mathscr{S}$.
- $f(\mathbf{x})>0$: Outside the object, distance to the nearest surface.
- $f(\mathbf{x})<0$: Inside the object, negative distance to the surface.
- $\nabla f(\mathbf{x})$: Surface normal at $\mathbf{x}\in\mathscr{S}$.

Remark 3.2.1 (Role of Neural SDF)**.**
The neural SDF implicitly represents the entire 3D geometry, enabling differentiable evaluation of surface points and normals, which is critical for ray tracing and shadow computation in ShadowNeuS.

3.2.1 ShadowNeuS Pipeline

ShadowNeuS operates through the following steps, executed iteratively during optimization:

1. **MLP Initialization**: Initialize an MLP with 8 layers and 256 hidden units to represent the SDF $f(\mathbf{x};\theta)$. Positional encoding $\gamma(\mathbf{x})$ is applied to the input $\mathbf{x}$ to capture high-frequency details:

$$\gamma(\mathbf{x})=[\mathbf{x},\sin(2^0\pi\mathbf{x}),\cos(2^0\pi\mathbf{x}),\dots]^T$$

2. **Camera Ray Marching**: For each pixel $\boldsymbol{p}=[p_x,p_y]^T$ in the input image, trace a camera ray:

$$\mathbf{p}(t)=\mathbf{C}+t\mathbf{d},\quad \mathbf{d}=R^{-1}K^{-1}\begin{bmatrix}p_x\\p_y\\1\end{bmatrix}$$

Find the surface point $\hat{\mathbf{x}}$ where $f(\hat{\mathbf{x}};\theta)\approx 0$ using numerical root-finding. Compute a differentiable surface point:

$$\hat{\mathbf{x}}=\mathbf{x}-\frac{\mathbf{d}}{\nabla f(\mathbf{x})\cdot\mathbf{d}}f(\mathbf{x}),\quad \mathbf{x}=\mathbf{p}(t)$$

3. **Shadow Ray Supervision**: For each surface point $\hat{\mathbf{x}}$, trace a shadow ray toward the light source $\mathbf{L}_i$:

   $$\mathbf{p}(t) = \hat{\mathbf{x}} + t(\mathbf{L}_i - \hat{\mathbf{x}}), \quad t \in [0,1]$$

   Sample points along the ray and evaluate the SDF to compute opacity:

   $$\alpha_i = \max\left(1 - \frac{\Phi_s(f(\mathbf{p}(t_{i+1});\theta))}{\Phi_s(f(\mathbf{p}(t_i);\theta))}, 0\right), \quad \Phi_s(z) = \frac{1}{1+e^{-sz}}$$

   Compute radiance:

   $$\hat{C}_{\text{in}}(\hat{\mathbf{x}}, \mathbf{L}_i) = L\prod_{i=1}^{N}(1-\alpha_i)$$

   where L is the light intensity (assumed constant).

4. **RGB Rendering (Optional)**: For RGB images, compute color at $\hat{\mathbf{x}}$:

   $$\hat{C} = (\rho_d + \rho_s)\hat{C}_{\text{in}}(\mathbf{L}_i - \hat{\mathbf{x}}) \cdot \nabla f(\hat{\mathbf{x}})$$

   where ρ_d and ρ_s are diffuse and specular reflectance coefficients.

5. **Loss Computation**: Compute the shadow or RGB loss to enforce consistency with input images:

   $$\mathscr{L}_{\text{shadow}} = \|\hat{C}_{\text{in}} - I_s\|_1, \quad \mathscr{L}_{\text{rgb}} = \|\hat{C} - I_r\|_1$$

   where I_s is the binary shadow image and I_r is the RGB image. Add the Eikonal loss to ensure a valid SDF:

   $$\mathscr{L}_{\text{eik}} = \frac{1}{M}\sum_{i=1}^{M}(\|\nabla f(\mathbf{p}_i;\theta)\|_2 - 1)^2$$

   Combine losses:

   $$\mathscr{L} = \mathscr{L}_{\text{shadow/rgb}} + 0.01 \cdot \mathscr{L}_{\text{eik}}$$

6. **Gradient Update**: Use the Adam optimizer to update MLP weights θ by minimizing $\mathscr{L}$ over 150,000 iterations, with a coarse-to-fine strategy (100x100 to 800x800 image resolution).

Remark 3.2.2 (Differentiable Pipeline)**.** The neural SDF and ray tracing are fully differentiable, allowing gradients to flow from the loss to θ, enabling end-to-end optimization of geometry and appearance.

3.2.2 Addressing Classical Limitations

ShadowNeuS overcomes the challenges listed in Section 2.5 as follows:

1. **Circular Dependency**: The neural SDF provides both geometry ($f(\mathbf{x})$) and normals ($\nabla f(\mathbf{x})$) simultaneously, resolving the need for separate normal estimation.

2. **Singular Cases**: The Eikonal loss ensures $\|\nabla f\|_2 = 1$, stabilizing normals near grazing angles.

3. **Non-Unique Solutions**: Multiple lighting conditions ($\{\mathbf{L}_i\}$) provide diverse shadow cues, reducing ambiguity in depth recovery.

4. **Non-Differentiable Occlusion**: The continuous SDF and sigmoid-based opacity Φ_s enable differentiable shadow ray tracing, replacing discrete intersection tests.

5. **Nonlinear Systems**: The shadow loss $\mathscr{L}_{\text{shadow}}$ implicitly enforces tangency constraints (Equation (20)) through optimization, avoiding explicit solvers.

6. **Ray Tracing Complexity**: GPU-based parallel processing and batched ray sampling reduce computational cost compared to classical $O(n^2)$ methods.
7. **Light Localization**: Known light positions $\{\mathbf{L}_i\}$ eliminate estimation errors, though robust loss functions can mitigate minor inaccuracies.
8. **Shadow Detection**: The L1 loss is robust to soft shadows and noise, as it compares entire shadow maps rather than relying on precise boundary detection.
9. **Multi-Object Complexity**: The neural SDF implicitly represents all surfaces, handling self-shadowing and multiple occluders by evaluating $f(\mathbf{x})$ along rays.

Remark 3.2.3 (Shadow Cues in ShadowNeuS)**.** Shadows provide geometric cues (surface orientation, tangency, depth ordering, occlusion) that constrain the SDF to match observed shadow boundaries and cast shadows across multiple lighting conditions, enabling reconstruction of complex geometries.

3.2.3 Shadow Consistency Loss

Definition 3.2.2 (Shadow Consistency Loss)**.** The shadow consistency loss in ShadowNeuS is:

$$\boxed{\mathscr{L}_{\text{shadow}} = \|\hat{C}_{\text{in}} - I_s\|_1} \tag{36}$$

where $\hat{C}_{\text{in}}$ is the predicted radiance from shadow ray tracing, and I_s is the observed binary shadow image.

Remark 3.2.4 (Comparison to Classical Loss)**.** Unlike the classical shadow consistency loss (Equation 2.4.2), which uses weighted terms for lit, shadow, and boundary mismatches, ShadowNeuS simplifies to a single L1 loss over all pixels. This leverages the neural SDF's ability to implicitly handle boundaries and regions through differentiable rendering.

Proposition 3.2.1 (Geometric Constraints via Shadows)**.** The shadow loss enforces:

- **Surface Orientation**: Points in $\Omega^{\text{obs}}_{\text{shadow}}$ correspond to $\mathscr{S}_{\text{shadow}}$ or $\mathscr{C}_{\text{total}}$, constraining $\nabla f(\mathbf{x})$.
- **Tangency**: Pixels in $\partial\Omega^{\text{obs}}_{\text{shadow}}$ satisfy Equation (20) implicitly through optimization.
- **Depth Ordering**: Cast shadows in $\mathscr{C}_{\text{total}}$ ensure correct relative positioning of surfaces.

Theorem 3.2.1 (Depth Recovery in ShadowNeuS)**.** For a pixel $\boldsymbol{p} \in \partial\Omega^{\text{obs}}_{\text{shadow}}$, the depth λ is constrained by:

$$\boxed{(\mathbf{C} + \lambda\mathbf{d} - \mathbf{L}_i) \cdot \nabla f(\mathbf{C} + \lambda\mathbf{d}) = 0} \tag{37}$$

ShadowNeuS solves this implicitly by optimizing the SDF to match shadow images across multiple lights $\{\mathbf{L}_i\}$.

Proof. For $\boldsymbol{p} \in \partial\Omega^{\text{obs}}_{\text{shadow}}$, the 3D point $\mathbf{P} = \mathbf{C} + \lambda\mathbf{d}$ lies on the shadow boundary $\mathscr{B}$. The tangency condition (Equation (20)) applies, with $\boldsymbol{n}(\mathbf{P}) = \nabla f(\mathbf{P})$, yielding Equation (37). $\square$

Remark 3.2.5 (Neural Optimization Advantage)**.** By optimizing $\mathscr{L}_{\text{shadow}} + 0.01 \cdot \mathscr{L}_{\text{eik}}$ over multiple lighting conditions, ShadowNeuS resolves depth ambiguities without explicit normal estimation or iterative solvers, leveraging the MLP's expressive power and gradient-based optimization.

References

[LWX23] Jingwang Ling, Zhibo Wang, Feng Xu. *ShadowNeuS: Neural SDF Reconstruction by Shadow Ray Supervision*. arXiv: 2211.14086, 2023.