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 20, 2025

Abstract

This paper explores *ShadowNeuS* [LWX23], a neural network that creates 3D models from single-view camera images by using clues from shadows and light. Unlike traditional methods that rely on multiple camera or sensors to build the 3D scene, *ShadowNeuS* uses a neural signed distance field (SDF) to accurately reconstruct 3D geometry. We will start by understanding how the network is trained and connected to geometry with spatial reasoning in 3D space $\mathbb{R}^3$. Finally, we will check our understanding by adapting the network for 2D reconstruction using similar conditions and methods.

Contents

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 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, MLPs

These data structures result from projecting 3D geometry through camera optics, where $C_n(\boldsymbol{p})$ corresponds to visible surface 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:

$$\begin{aligned}
\text{Camera center:} \quad & \boldsymbol{C} = [C_x, C_y, C_z]^\top \in \mathbb{R}^3 && (4)\\
\text{Rotation matrix:} \quad & R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \in \mathrm{SO}(3) && (5)\\
\text{Translation vector:} \quad & \boldsymbol{t} = -R\boldsymbol{C} \in \mathbb{R}^3 && (6)
\end{aligned}$$

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 forward projection involves three sequential transformations of the 3D point $\boldsymbol{P}$ to pixel point $\boldsymbol{p}$:

Step 1: World to camera coordinates

$$\boldsymbol{P}_{\text{cam}} = R(\boldsymbol{P} - \boldsymbol{C}) = R\boldsymbol{P} + \boldsymbol{t} \tag{8}$$

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 $\lambda = z'$:

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

□

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-surface interactions

2 Shadows as a Geometric Constraint

To what extent can shadows constrain 3D geometry from a single image? In this section, we formalize shadow formation, analyze the geometric information it provides, discuss its limitations and set up the motivation for neural solutions.

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

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:

$$\boxed{\mathscr{C}_{1\to2} = \{\boldsymbol{P} \in \mathscr{S}_2 \mid \exists t \in (0,1) : \boldsymbol{L} + t(\boldsymbol{P}-\boldsymbol{L}) \in \mathscr{S}_1\}} \tag{25}$$

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:

$$\boxed{\partial\mathscr{C}_{1\to2} = \{\boldsymbol{P} \in \mathscr{S}_2 \mid \boldsymbol{P} = \boldsymbol{Q} + s(\boldsymbol{Q}-\boldsymbol{L}), \boldsymbol{Q} \in \mathscr{B}_1, s > 0\}} \tag{26}$$

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:

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

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

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 is partitioned as:

$$\Omega_{\text{lit}}^{\text{obs}} = \{\pi(\boldsymbol{P}) \mid \boldsymbol{P} \in \mathscr{S}_{\text{lit}}, \pi(\boldsymbol{P}) \text{ not occluded}\} \qquad \text{(lit region)} \tag{29}$$

$$\Omega_{\text{shadow}}^{\text{obs}} = \{\pi(\boldsymbol{P}) \mid \boldsymbol{P} \in \mathscr{S}_{\text{shadow}} \cup \mathscr{C}_{\text{total}}, \pi(\boldsymbol{P}) \text{ not occluded}\} \qquad \text{(shadow region)} \tag{30}$$

$$\partial\Omega_{\text{shadow}}^{\text{obs}} = \{\pi(\boldsymbol{P}) \mid \boldsymbol{P} \in \mathscr{B} \cup \partial\mathscr{C}_{\text{total}}, \pi(\boldsymbol{P}) \text{ not occluded}\} \qquad \text{(shadow boundary)} \tag{31}$$

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). Only points not occluded by the object contribute to $\Omega_{\text{shadow}}^{\text{obs}}$, as shadows may be blocked by the object itself.

Remark 2.4.1 (Shadow Projection and Occlusion).
The projection π maps 3D points to 2D image pixels. A lit point $\boldsymbol{P} \in \mathscr{S}_{\text{lit}}$ projects to a bright pixel $\pi(\boldsymbol{P}) \in \Omega_{\text{lit}}^{\text{obs}}$ if not occluded by the object. Similarly, a shadowed point $\boldsymbol{P} \in \mathscr{S}_{\text{shadow}} \cup \mathscr{C}_{\text{total}}$ projects to a dark pixel in $\Omega_{\text{shadow}}^{\text{obs}}$ only if not occluded. Shadows cast by the object may be hidden, so not all shadowed points appear in the image.

Definition 2.4.2 (Shadow Region Classification Accuracy).
This accuracy metric evaluates how well the reconstructed 3D geometry correctly predicts the image regions classified as lit, shadowed, or the boundary:

$$\text{Accuracy} = \frac{\text{Number of correctly predicted pixels}}{\text{Total number of pixels}} = \frac{N_{\text{correct}}}{N_{\text{total}}} \tag{32}$$

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 Depth Recovery

For a pixel $\boldsymbol{p}$ on the shadow boundary, depth recovery combines the camera ray (12) with the Surface Illumination Partition (2.2.1):

$$\boxed{\boldsymbol{P}(\lambda) = \boldsymbol{C} + \lambda \boldsymbol{d} \quad \text{and} \quad \begin{cases} (\boldsymbol{P}(\lambda) - \boldsymbol{L}) \cdot \hat{\boldsymbol{n}} > 0 \\ (\boldsymbol{P}(\lambda) - \boldsymbol{L}) \cdot \hat{\boldsymbol{n}} = 0 \\ (\boldsymbol{P}(\lambda) - \boldsymbol{L}) \cdot \hat{\boldsymbol{n}} < 0 \end{cases}} \tag{33}$$

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

$$\boxed{\begin{cases} \lambda > \dfrac{(\boldsymbol{L} - \boldsymbol{C}) \cdot \hat{\boldsymbol{n}}}{\boldsymbol{d} \cdot \hat{\boldsymbol{n}}} & \text{(in attached shadow region)} \\ \lambda = \dfrac{(\boldsymbol{L} - \boldsymbol{C}) \cdot \hat{\boldsymbol{n}}}{\boldsymbol{d} \cdot \hat{\boldsymbol{n}}} & \text{(on shadow boundary)} \\ \lambda < \dfrac{(\boldsymbol{L} - \boldsymbol{C}) \cdot \hat{\boldsymbol{n}}}{\boldsymbol{d} \cdot \hat{\boldsymbol{n}}} & \text{(point is illuminated)} \end{cases}} \tag{34}$$

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

Limitations and Computational Challenges

Fundamental Challenges

1. **Circular Dependency:** Equation (34) requires surface normals $\hat{\boldsymbol{n}}$ at point $P(\lambda)$ to find the value of λ.

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 the shadow information.

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

Remark 2.5.2 (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 a set of known light source positions $\{\mathbf{L}_i\}$.
- **Calibrated light positions**: Light source locations are known or pre-calibrated.
- **Static scene**: No object movement between lighting conditions.
- **Observable shadows**: Clear and distinguishable shadow boundaries in the captured images.

This setup allows ShadowNeuS to leverage shadow cues to recover complete 3D geometry, including occluded or non-visible regions, by combining classical geometric reasoning 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.

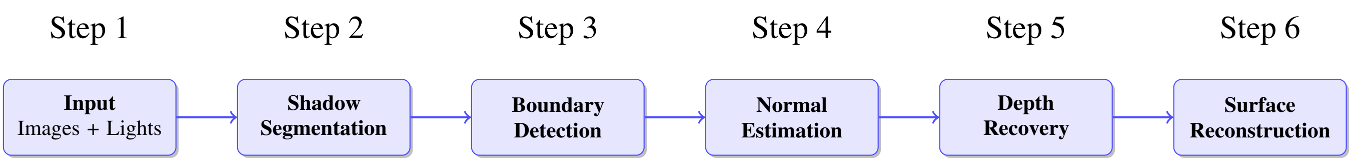

Figure 1: Classical method (Sequential processing pipeline)

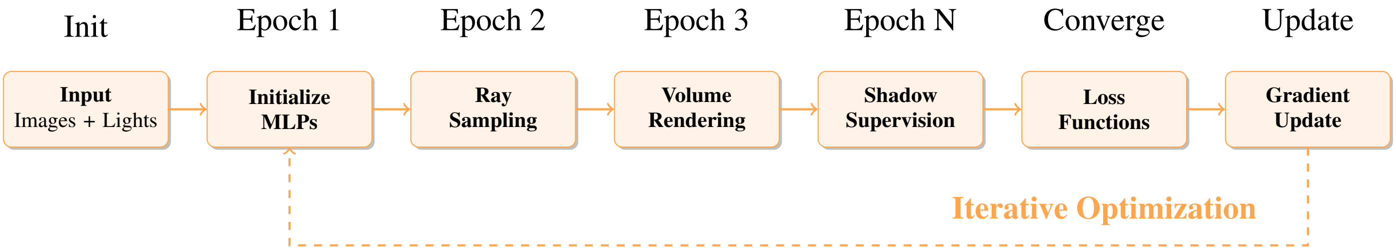

Figure 2: ShadowNeuS (End-to-end neural optimization pipeline)

3.2 ShadowNeuS Pipeline

In this section, we outline the complete pipeline of ShadowNeuS, from the initialization of neural SDFs to optimization using shadow supervision.

3.2.1 Neural Signed Distance Fields (SDF)

Definition 3.2.1 (Neural Signed Distance Field)**.**
A **Neural Signed Distance Field** (Neural SDF), as introduced in [PFS19], is a function $f(\mathbf{P};\theta):\mathbb{R}^3\to\mathbb{R}$ parameterized by a multi-layer perceptron (MLP) with trainable weights θ. It implicitly represents a 3D scene by outputting the signed distance of any spatial point $\mathbf{P}=(p_x,p_y,p_z)^\top$ to the nearest surface:

$$f(\mathbf{P})\begin{cases}<0 & \text{if } \mathbf{P} \text{ is inside the object,}\\ =0 & \text{if } \mathbf{P} \text{ lies on the surface,}\\ >0 & \text{if } \mathbf{P} \text{ is outside the object.}\end{cases}$$

Remark 3.2.1 (Key Mathematical Properties of Neural SDF)**.**
Neural SDFs encode geometric and differential properties useful for 3D reconstruction from shadow cues:

- **Differentiability**: $f(\mathbf{P};\theta)$ is differentiable with respect to both spatial coordinates $\mathbf{P}$ and neural parameters θ, enabling end-to-end gradient-based optimization:

$$\nabla_{\mathbf{P}} f(\mathbf{P};\theta), \quad \nabla_{\theta} f(\mathbf{P};\theta).$$

- **Surface Normals**: At the zero-level set $f(\mathbf{P})=0$, the unit surface normal vector is computed via normalized spatial gradients:

$$\hat{\mathbf{n}}(\mathbf{P})=\frac{\nabla_{\mathbf{P}} f(\mathbf{P})}{\|\nabla_{\mathbf{P}} f(\mathbf{P})\|_2}. \tag{35}$$

  This facilitates shadow-based light visibility computations via the angle between $\hat{\mathbf{n}}$ and the light direction $\mathbf{L}-\mathbf{P}$.

- **Eikonal Regularization (Distance Consistency)**: For $f(\mathbf{P})$ to represent a valid distance function locally, it must satisfy the *Eikonal equation*:

$$\|\nabla_{\mathbf{P}} f(\mathbf{P})\|_2=1, \quad \forall \mathbf{P}\in\mathbb{R}^3. \tag{36}$$

  A first-order Taylor approximation confirms this property:

$$f(\mathbf{P}+\sigma\mathbf{u})\approx f(\mathbf{P})+\sigma\cdot(\nabla_{\mathbf{P}} f(\mathbf{P})\cdot\mathbf{u}), \quad \|\mathbf{u}\|_2=1, |\sigma|\ll 1. \tag{37}$$

- **Geometric Stability**: Deviations from $\|\nabla_{\mathbf{P}} f(\mathbf{P})\|_2=1$ indicate distortion:

$$\|\nabla_{\mathbf{P}} f(\mathbf{P})\|_2\begin{cases}>1 & \text{implies local stretching (distance overestimation),}\\ <1 & \text{implies local compression (distance underestimation).}\end{cases}$$

  Enforcing the Eikonal constraint regularizes the SDF, ensuring stable and consistent geometry during optimization.

3.2.2 Epoch 1: Neural SDF Initialization and MLP Design

The training begins by constructing a neural signed distance field (SDF), modeled by an eight-layer multi-layer perceptron (MLP) to implicitly represent 3D geometry from shadow cues.

- **Network Structure**: ShadowNeuS uses an 8-layer fully-connected MLP with ReLU activation functions, a single output head producing scalar signed distances $f(\mathbf{P};\theta) \in \mathbb{R}$.

  Remark 3.2.2 (Why MLP Architecture?).
  The MLP design offers several advantages:

  - **Universal Approximation**: By the universal approximation theorem, MLPs can approximate any continuous function on compact subsets of $\mathbb{R}^3$.
  - **End-to-End Differentiability**: Gradients $\nabla_{\mathbf{P}} f$, $\nabla_{\theta} f$ can be computed throughout the pipeline, enabling training directly from shadow observations.
  - **Joint Integrative Learning**: Both surface positions ($f(\mathbf{P}) = 0$) and surface normals ($\nabla_{\mathbf{P}} f$) are jointly encoded in f, simplifying the learning process compared to classical pipelines which handle these separately.
  - **Circular Dependency Resolution**: Neural SDF jointly models geometry and visibility, avoiding the iterative or circular inversion problems faced by classical shadow methods.

- **Input Encoding**: To enhance the ability of the MLP to capture high-frequency geometric features, ShadowNeuS applies a sinusoidal positional encoding $\gamma(\mathbf{P})$ to the input 3D coordinates:

  $$\gamma(\mathbf{P}) = \left(\sin(2^0\pi p_x), \cos(2^0\pi p_x), \ldots, \sin(2^{k-1}\pi p_z), \cos(2^{k-1}\pi p_z)\right). \tag{38}$$

  Remark 3.2.3 (Why Positional Encoding?).
  Raw coordinates $\mathbf{P} = (p_x, p_y, p_z)$ do not provide sufficient high-frequency variation for MLPs, leading to overly smooth approximations (spectral bias). Positional encoding introduces Fourier feature mappings [TSS20], allowing the network to represent both coarse and fine geometric structures revealed by shadows.

- **Output**: A scalar SDF value $f(\mathbf{P})$ per point.

- **Random Initialization**: The MLP weights θ are initialized using standard techniques such as Xavier initialization, ensuring well-scaled activations in early epochs.

3.2.3 Epoch 2→N: Ray Sampling and Shadow Rendering

From the second epoch onward, ShadowNeuS iteratively performs **ray sampling** to locate surface points and **shadow rendering** to predict shadow values at image pixels. The procedure proceeds as follows:

- **Camera Ray Sampling**:
  For each pixel $\mathbf{p}$, a camera ray is constructed:

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

  The intersection point $\mathbf{P}$ is found by sphere tracing along $\mathbf{r}(t)$ to locate t^* satisfying

  $$f_\theta(\mathbf{r}(t^*)) = 0, \quad \mathbf{P} = \mathbf{r}(t^*).$$

- **Light Ray Sampling and Visibility Estimation**:
  For each $\mathbf{P}$, a light ray towards the light source $\mathbf{L}$ is defined:

  $$\mathbf{l}(s) = \mathbf{P} + s(\mathbf{L} - \mathbf{P}), \quad s \in [0,1].$$

  The **opacity** along $\mathbf{l}(s)$ is computed at discrete points $\mathbf{l}(s_j)$ using:

  $$a_j = \max\left(1 - \frac{\phi(f_\theta(\mathbf{l}(s_{j+1})))}{\phi(f_\theta(\mathbf{l}(s_j)))}, 0\right)$$

  where $\phi(\cdot)$ is an active function, and $a_j \in [0,1]$ represents the **local opacity** between adjacent samples.

  The accumulated transmittance (incoming light intensity) is then:

  $$C_{\text{in}}(\mathbf{P}) = \prod_{j=1}^{N}(1 - a_j)$$

  This idea is inspired by Neural Radiance Fields (NeRF) [MST20], which uses volume rendering to synthesize views from radiance fields.

3.2.4 Converge: Loss Functions

The overall objective of the training process is to minimize a weighted sum of several loss terms that enforce both geometric and appearance-based constraints. The total loss function is defined as:

$$\mathcal{L}_{\text{total}} = w_{\text{shadow}} \cdot \mathcal{L}_{\text{shadow}} + w_{\text{eik}} \cdot \mathcal{L}_{\text{eik}} + w_{\text{app}} \cdot \mathcal{L}_{\text{appearance}} + \dots \tag{39}$$

- **Shadow Supervision Loss:**

  $$\mathcal{L}_{\text{shadow}} = \sum_{\mathbf{p}} |C_{\text{in}}(\mathbf{p}) - S(\mathbf{p})| \tag{40}$$

  This term measures the difference between the predicted incoming light intensity $C_{\text{in}}(\mathbf{p})$ and the observed shadow value $S(\mathbf{p})$ over all valid image pixels $\mathbf{p}$.

- **Eikonal Loss:**

  $$\mathcal{L}_{\text{eik}} = \frac{1}{M}\sum_{i=1}^{M}(\|\nabla_{\mathbf{P}} f(\mathbf{P}_i)\|_2 - 1)^2 \tag{41}$$

  This regularization ensures the predicted signed distance function (SDF) f has unit gradient norm almost everywhere, encouraging smooth and well-defined surfaces.

- **Appearance or Consistency Loss:**

  $$\mathcal{L}_{\text{appearance}} = \sum_{\mathbf{p}} \left\|C_{\text{pred}}(\mathbf{p}) - C_{\text{gt}}(\mathbf{p})\right\|_2^2 \tag{42}$$

  This term enforces color or texture consistency by minimizing the error between predicted colors C_{pred} and ground-truth colors C_{gt} at pixel locations $\mathbf{p}$. Depending on the dataset, it can also include feature or texture-based losses.

- **Other Loss Terms (optional):**

  Additional loss terms such as normal supervision, depth alignment, or regularization losses can be added when extra supervision signals are available.

3.2.5 Gradient Descent and Adam Optimizer

To minimize the total loss $\mathscr{L}_{\text{total}}(\theta)$, neural networks typically rely on gradient-based optimization methods. Below, we summarize both the basic gradient descent method and the more advanced Adam optimizer used in our implementation. Details of Adam optimizer are refered to paper [KB14].

- **Gradient Descent**: The simplest optimization method updates the parameters θ along the direction of steepest descent, defined by the negative gradient of the loss:

$$\theta_{t+1} = \theta_t - \alpha \nabla_\theta \mathscr{L}_{\text{total}}(\theta_t),$$

where θ_t denotes the parameters at iteration t, $\alpha > 0$ is the **learning rate**, and $\nabla_\theta \mathscr{L}_{\text{total}}$ is the gradient of the total loss with respect to θ.

Remark 3.2.4 (Limitations of Simple Gradient Descent)**.** Despite its simplicity, basic gradient descent exhibits several limitations:

  - **Learning Rate Sensitivity**: A fixed α may result in slow convergence (if too small) or divergence (if too large).
  - **Oscillation**: In regions with steep or curved loss landscapes, updates can oscillate and hinder convergence.
  - **Uniform Step Size**: All parameters are updated using the same learning rate, without accounting for variations in gradient magnitude.

- **Adam Optimizer**: Adaptive Moment Estimation (Adam) addresses these issues by combining:

  - **Momentum (First Moment Estimate)**: Adam maintains an exponentially decaying average of past gradients:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t, \quad g_t = \nabla_\theta \mathscr{L}_{\text{total}}(\theta_t),$$

  where $0 < \beta_1 < 1$ controls the momentum decay. This term reduces oscillations and stabilizes updates.

  - **Adaptive Scaling (Second Moment Estimate)**: Adam also tracks the exponentially decaying average of squared gradients:

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2,$$

  where $0 < \beta_2 < 1$ governs the decay rate for squared gradients. This adjusts learning rates per parameter based on historical gradient magnitudes.

  - **Bias Correction**: To correct for initialization bias (since $m_0, v_0 = 0$), Adam computes bias-corrected estimates:

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1-\beta_2^t}.$$

- **Adam Update Rule**: The final parameter update rule is given by:

$$\theta_{t+1} = \theta_t - \alpha \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon},$$

where:

  - α is the learning rate (e.g., $\alpha = 10^{-3}$),
  - β_1 is the first moment decay rate (typically $\beta_1 = 0.9$),
  - β_2 is the second moment decay rate (typically $\beta_2 = 0.999$),
  - ε is a small constant (e.g., $\varepsilon = 10^{-8}$) to ensure numerical stability and prevent division by zero.

References

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

[PFS19] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, Steven Lovegrove. *DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation*. arXiv: 1901.05103, 2019.

[TSS20] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, Ren Ng. *Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains*. arXiv: 2006.10739, 2020.

[MST20] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, Ren Ng. *NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis*. arXiv: 2003.08934, 2020.

[KB14] Diederik P. Kingma, Jimmy Ba. *Adam: A Method for Stochastic Optimization*. arXiv: 1412.6980, 2014.