Electronics and Computer Science
Faculty of Engineering and Physical Sciences
University of Southampton

Daniel Hoang
October 27, 2026

Reconstructing 3D Geometry from Fisheye Photos

Project Supervisor: Dr. Andersen Ang
Second Examiner: Dr. David Millard

Contents

1 Introduction **5**

2 Camera models, dewarping and literature **6**
2.1 Pinhole Cameras . . . 6
2.1.1 Paraxial Refraction Model . . . 7
2.1.2 Image Space . . . 7
2.1.3 Intrinsic and Extrinsic Parameters . . . 7
2.2 Fisheye Cameras . . . 8
2.3 The Dewarping Algorithm . . . 9

3 Report on Technical Progress **10**
3.1 Dewarping . . . 10
3.2 Optimisation . . . 11
3.2.1 Single Image Parameter Ternary Search . . . 11
3.2.2 Cross Validation . . . 12

4 3D reconstruction from 2D images **14**
4.1 Planar Face Detection from a Single Greyscale Image . . . 14
4.1.1 Stage 1: Pre-Filtering and Canny Edge Detection . . . 15
4.1.2 Stage 2: Sequential RANSAC Line Fitting . . . 16
4.1.3 Stage 2b: Line Extension Toward Intersections and Image Edges . . . 17
4.1.4 Stage 3: Face Construction via Boundary Rasterisation and Region Growing . . . 17
4.1.5 Stage 4: Vertex Detection via Line-Line Intersection . . . 22
4.1.6 Stage 5: Vertex-to-Face Assignment . . . 23
4.1.7 Planar detection results . . . 23
4.1.8 Summary of Parameters . . . 23
4.2 Surface Normal Estimation via Piecewise-Planar Shape-from-Shading . . . 24
4.2.1 Reflectance Model and Structural Assumptions . . . 24
4.2.2 Stage N1: Face Adjacency Graph and Boundary Pixel Collection . . . 25
4.2.3 Stage N2: Anchor Face Identification . . . 25
4.2.4 Stage N3: Image-Plane Edge Directions . . . 26
4.2.5 Stage N4: Visible-Hemisphere Parameterisation . . . 26
4.2.6 Stage N5: Cost Function . . . 26
4.2.7 Stage N6: Initialisation by Light Grid Search . . . 28
4.2.8 Stage N7: Joint Refinement by Alternating Coordinate Descent . . . 28
4.2.9 Stage N8: Output and Diagnostics . . . 30
4.2.10 Minimal Improvements . . . 30
4.2.11 Normal Estimation Results . . . 31
4.2.12 Structural Assumptions and Problem Limitations . . . 31
4.2.13 Summary of Parameters . . . 32
4.3 Polyhedron Reconstruction from Estimated Normals . . . 32
4.3.1 World Frame and Orthographic Lift . . . 34
4.3.2 Pass 1: Plane Offsets via Breadth-First Propagation . . . 34
4.3.3 Pass 2: Vertex Depths from Incident Plane Equations . . . 35
4.3.4 Reconstruction Results . . . 36
4.3.5 Limitations and Sources of Error . . . 37

5 Complete Pipeline **39**

6 Reflection **39**

7 Future Work **40**

8 Appendix: Planar Face Detection Stage Visualisations **41**

9 Appendix: Project Management **46**

Abstract Fisheye cameras provide a wide field of view, making them desirable for many applications, including surveillance and robotics. These cameras, however, introduce non-linear distortions complicating geometric analysis. This thesis presents a complete pipeline for fisheye image processing for 3D geometry reconstruction. We first model the fisheye lens with a projective framework to dewarp and rectify the image, producing metrically consistent views. The rectified images are then integrated into a multi-view reconstruction system. We evaluate the proposed approach on real-world datasets to maximise accuracy and confirm that rectified fisheye imagery can be reliably used for precise 3D reconstruction.

Symbol and terminology

Symbol / Abbreviation	Meaning / Definition
Span *Camera Model & Geometry*	
P	Point in 3D space $[X, Y, Z]^T$, relative to the camera
P_w	Coordinates of object in real-world coordinate system
r	Point on image plane $[u, v]^T$
O	Aperture point (pinhole / lens centre)
f	Focal length
$[u_0, v_0]^T$	Principal point
θ	Angle of incidence between optical axis and ray to point P
K	Intrinsic camera matrix
R, T	Extrinsic rotation matrix and translation vector
Span *Fisheye Model & Dewarping*	
r_d, r_u	Distorted and undistorted radii on the image plane
$a_1, \ldots, a_4$	Distortion coefficients (WoodScape polynomial model)
Θ	Optimisation parameter vector $\{f_t, f_{src}, a_1, \ldots, a_4\}$
I_{src}, I_{gt}	Source (fisheye) image and ground-truth image
RMSE	Root Mean Squared Error (loss metric)
Span *Planar Face Detection*	
I	Greyscale image $I : \Omega \to [0, 1]$
$\mathcal{L}$	Set of fitted line segments
(a, b, c)	Line parameters in implicit form $ax + by + c = 0$
L	Label map assigning each pixel to a face
N_f, N_v	Number of detected faces and vertices
$\mathcal{F}_k$	Polygon vertex list of face k
$\mathcal{B}_e$	Boundary pixel cloud for adjacency edge e
Span *Normal Estimation*	
$\mathbf{n}_f$	Unit surface normal of face f
$\mathbf{l}$	Global light direction
ρ	Scalar albedo
I_f	Median intensity of face f
$\mathbf{v}$	Camera view direction
$\mathbf{n}_g$	Ground (anchor) normal $(0, 0, 1)^T$
g_0	Anchor face index
(θ_f, ϕ_f)	Polar / azimuth angles parameterising $\mathbf{n}_f$
$\mathcal{C}$	Total cost function (intensity + four priors)
$\lambda_c, \lambda_v, \lambda_a, \lambda_e$	Weights for convexity, visibility, coplanar, edge priors
Span *3D Reconstruction*	
$\mathbf{P}_i$	Recovered 3D world position of vertex i
d_k	Plane offset of face k in $\mathbf{n}_k \cdot \mathbf{P} = d_k$
$\mathcal{V}_g$	Set of ground-anchored vertices
$\mathcal{K}_i$	Set of wall planes incident to vertex i
Span *Abbreviations*	
RANSAC	Random Sample Consensus
BFS	Breadth-First Search
PCA / SVD	Principal Component Analysis / Singular Value Decomposition
DOF	Degrees of Freedom

1 Introduction

Project Overview Modern cameras capture detailed photos of the world, yet turning these flat images into accurate 3D models remains challenging. This project uses fisheye cameras, which have an extremely wide field of view, producing distorted photos, hindering reconstruction of accurate 3D geometry of objects. We aim to take these heavily curved images, correct the distortion, and reconstruct the captured real-world geometry.

Scope and Motivation The project aims to create the full pipeline, from fisheye image capture to final 3D geometry reconstruction. Fisheye cameras have become increasingly common in robots, security systems, autonomous vehicles, and virtual reality. The wide field of view simplifies covering large areas with fewer cameras. However, the heavy distortion hinders current standard computer vision techniques. Reliably using fisheye images for accurate 3D reconstruction would significantly improve mapping and navigation systems.

Technical Background Overview This project builds on concepts and theory from camera geometry, projection models, lens distortion, and multi-view 3D reconstruction. Pinhole camera models describe how 3D points project onto a flat image sensor, whereas fisheye models describe the mapping of incoming light angles. Fisheye images must be undistorted and rectified to resemble a pinhole camera photo before meaningful 3D reconstruction can occur.

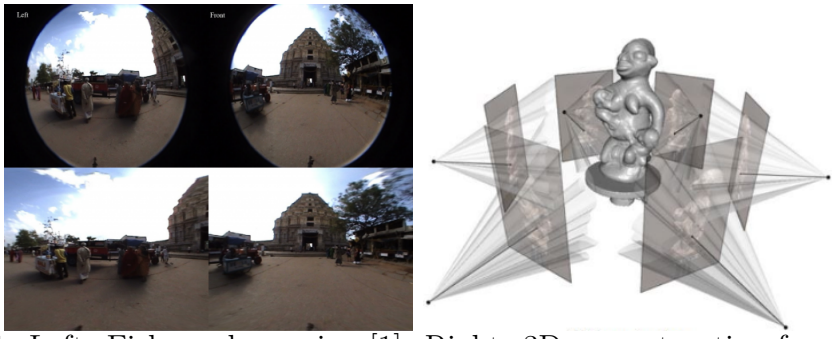


Figure 1: Left: Fisheye dewarping [1]. Right: 3D reconstruction from 2D [2].

Project Goals

- Develop a complete pipeline for fisheye image 3D reconstruction, from capture to final model generation.
- Calibrate an equidistant fisheye camera and estimate its intrinsics and extrinsics.
- Model and correct fisheye lens distortion with established polynomial models.
- Dewarp and rectify fisheye images to metrically consistent perspective views.
- Integrate rectified images into a Structure from Motion pipeline for camera pose estimation and sparse 3D reconstruction.
- Evaluate reconstruction accuracy using reprojection error, depth consistency, and visual fidelity.
- Compare fisheye reconstruction results against standard pinhole camera baselines.
- Demonstrate practical applicability of fisheye 3D reconstruction in real-world environments.

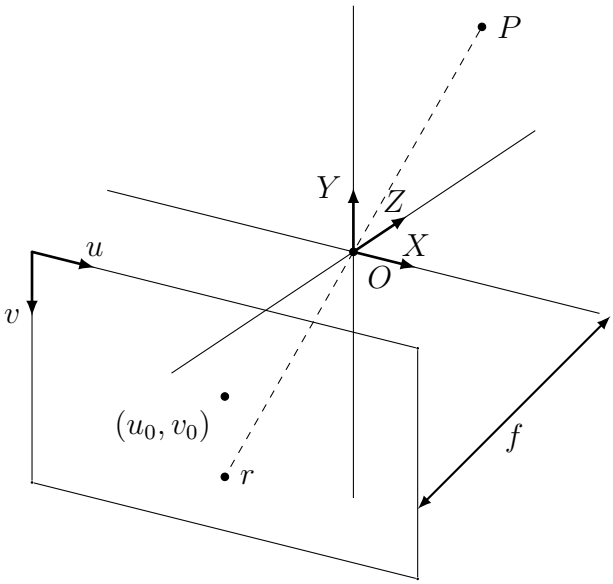
2 Camera models, dewarping and literature

2.1 Pinhole Cameras

Pinhole cameras underlie the standard model for modern cameras. They are the simplest camera, allowing light through a tiny hole in a plane; light passing through projects an inverted image onto the opposite surface (image plane). Pinhole cameras demonstrate the fundamental principles of image projection and are essential to dewarp fisheye photos.

In 3D camera model, a 3D point $P = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$ is mapped to the 2D coordinate of the pixel on the image $r = \begin{bmatrix} u \\ v \end{bmatrix}$, where the x-axis is horizontal, y-axis is vertical and z-axis is depth from the camera's focal point [3].

The light rays are projected through the aperture point O onto the image plane, which is a distance f away. The principal point, usually in the centre of the image plane, defines the origin of the coordinate system and coordinates $(0, 0)$ are defined to be the top-left pixel of the image plane.



P is being transformed to point r, and thus we can define a matrix K as our camera model that maps point $P = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$ to $r = \begin{bmatrix} u \\ v \end{bmatrix}$ such that $Zr' = KP$ with $r' = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$ and $K = \begin{bmatrix} f & 0 & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{bmatrix}$, where r' are the **homogenous coordinates** of r as the equation wouldn't work with a 2×1 vector.

This matrix is a special case of a projective transformation that scales by focal length f and translates by the **principal point** $[u_0, v_0]$ (usually the centre of the image plane) [3]. Also, we can have different focal lengths for the x and y coordinates ($K_{00} \neq K_{11}$); however, it's always possible to stretch the image so that $K_{00} = K_{11}$.

Rearranging the equation, we can get the pixel coordinates on the image plane from the 3D coordinates of the object. From $Zr' = KP$,

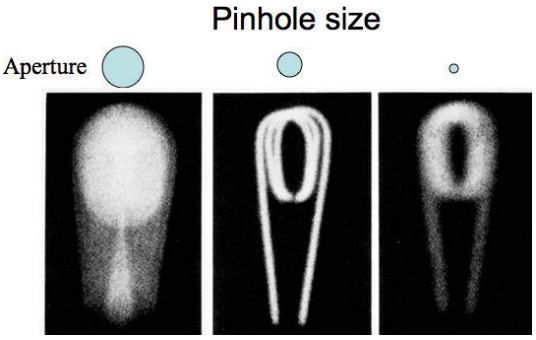
$$Z \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} fX + Zu_0 \\ fY + Zv_0 \\ Z \end{bmatrix} \implies u = f\tfrac{X}{Z} + u_0, \quad v = f\tfrac{Y}{Z} + v_0$$

We can define normalised coordinates x, y as

$$x = \tfrac{X}{Z}, \qquad y = \tfrac{Y}{Z} \qquad \implies \qquad u = fx + u_0, \quad v = fy + v_0$$

2.1.1 Paraxial Refraction Model

The size of the aperture O influences how sharp an image is on the image plane; the larger the aperture, the more light rays are let through, which can distort the image. We cannot assume an infinitely small aperture, and modern cameras use lenses to focus the light rays onto a single point on the image plane.



How aperture size affects image quality in a pinhole camera [4].

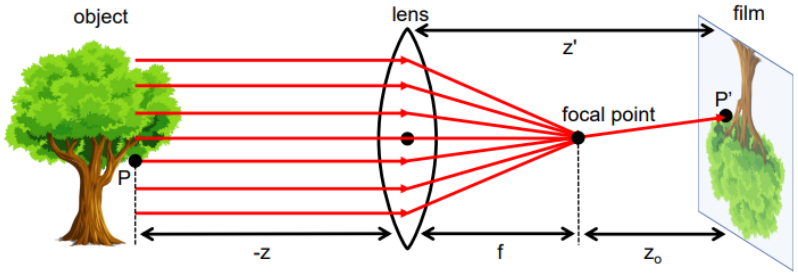
This is the **paraxial refraction model**, we introduce $z' = f + z_0$ (image distance). We arrive at a similar construction as the pinhole model, to map the 3D coordinates to the image plane.

By similar triangles, we derive the image point $P' = (X', Y', z')$:

$$\frac{X'}{X} = \frac{Y'}{Y} = \frac{z'}{Z} \implies X' = \frac{z'}{Z}X, Y' = \frac{z'}{Z}Y$$

Including the principal point as an offset, its sensor coordinates $r = (u, v)$ are

$$u = \frac{z'}{Z}X + u_0, \quad v = \frac{z'}{Z}Y + v_0$$

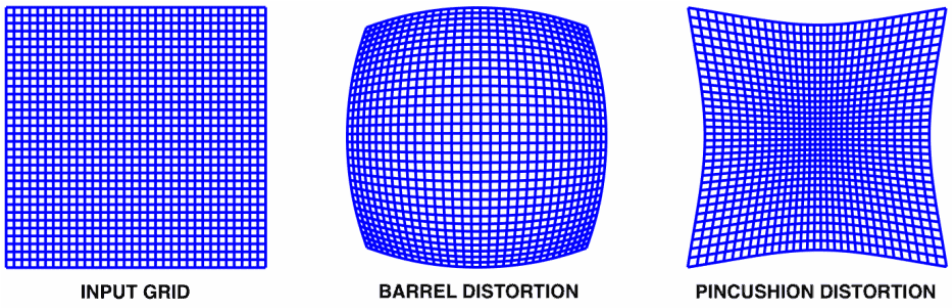


Lens focuses rays parallel to the optical axis onto the focal point [5]

Assume the camera uses a **thin lens**, we use the approximation to relate the **focal length** f, **depth** z and image distance z' [6]: $\frac{1}{f} = \frac{1}{Z} + \frac{1}{z'}, z' = \frac{fZ}{Z-f}$. Substituting to get rid of z':

$$\boxed{u = \frac{f}{Z-f}X + u_0, \qquad v = \frac{f}{Z-f}Y + v_0}$$

We do not move forward with these equations; however, they're important if we want to account for any **radial distortion** in our images later in the process. All subsequent derived results will still hold true for the paraxial refraction model.



Example of radial distortion due to lens [7]

2.1.2 Image Space

We consider digital coordinates are measured in pixels, and points in the image plane are represented using physical measurements [5]. We introduce parameters k and l that has units similar to $\frac{\text{pixels}}{\text{cm}}$. $u = fkx + u_0 = \alpha x + u_0, v = fly + v_0 = \beta y + v_0$. Thus we update our camera matrix model slightly:

$$K = \begin{bmatrix} \alpha & 0 & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix}.$$

2.1.3 Intrinsic and Extrinsic Parameters

Our camera matrix model doesn't account for skew or distortion. An image is skewed when the camera coordinate system is skewed, meaning the angle between two axes is slightly larger or smaller

than 90 degrees [5]. Our complete camera matrix model can be derived as $K = \begin{bmatrix} \alpha & -\alpha \cot \theta_O & u_0 \\ 0 & \frac{\beta}{\sin \theta_O} & v_0 \\ 0 & 0 & 1 \end{bmatrix}$, where θ_O represents the angle between the x- and y-axes of the camera's image plane, the variables in the matrix model are called **intrinsic parameters** since they are unique and inherent to each camera and relate to the essential properties of it [5].

Extrinsic parameters represent the camera's position and orientation in the 3D world. We use a rotation matrix R to describe the camera's rotation in space relative to the world coordinate system and a translation vector T to describe the position of the camera in the world. A point in a world reference system P_w, we compute it's coordinates relative to the camera [5]: $P = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} P_w$. Getting a complete equation from world position to image plane position $Zr' = K[R \ T]P_w = MP_w$.

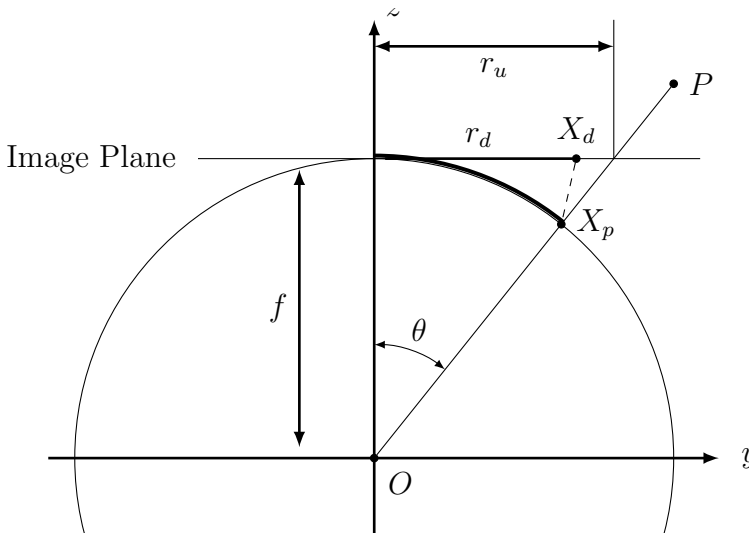
2.2 Fisheye Cameras

Fisheye lenses were made to capture a vast field of view by mapping angles, not depth, onto the image plane. Unlike pinhole cameras, fisheye lenses can extend beyond 90°, designed for applications where the direction of incoming light matters more than the reconstruction of straight lines or metric depth. There are 4 main types of fisheye camera models: equidistant, equisolid, stereographic and orthographic. We will mainly focus on equidistant, as it's the standard.



Circular fisheye view of Amsterdam [8]

Projection The equidistant fisheye lens model shows the 2D view of the projective geometry of an equidistant fisheye lens camera [9]. r_d represents the projected pixel from the fisheye lens, whereas r_u is where the pixel would have been projected if it was a pinhole camera. X_p represents the point of intersection between the light ray from point P and the **camera plane**. r_d is the arc length between the z-axis and X_p. r_d is then directly proportional to the angle θ ($r_d = f\theta$). All derivations can and will be extended to 3D using the same concepts.



In a fisheye camera, everything depends on the angle between the ray and the optical axis:

$$\theta = \text{atan2}(\sqrt{X^2 + Y^2}, Z)$$

This is called the **angle of incidence**. In an equidistant fisheye camera model, the 3D to 2D projection is defined as [3]:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} Z = \begin{bmatrix} \alpha & 0 & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{XZ}{\sqrt{X^2+Y^2}}\theta \\ \frac{YZ}{\sqrt{X^2+Y^2}}\theta \\ Z \end{bmatrix}, \quad \theta = \text{atan2}(\sqrt{X^2 + Y^2}, Z)$$

Rearrange for u and v with $\alpha = fk$, $\theta = \frac{r}{f}$, we get:

$$\begin{aligned} u &= \alpha \left(\frac{X}{\sqrt{X^2+Y^2}}\right)\theta + u_0 = fk \left(\frac{X}{\sqrt{X^2+Y^2}}\right)\frac{r}{f} + u_0 = rk \left(\frac{X}{\sqrt{X^2+Y^2}}\right) + u_0 \\ v &= rl \left(\frac{Y}{\sqrt{X^2+Y^2}}\right) + v_0 \end{aligned}$$

Extrinsic parameters aren't camera-specific; thus, we have $P = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} P_w$ and $\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix}$.

Distortion In the equidistant fisheye camera model, the distance between the pixel in the image and the principal point is directly proportional to the angle of incidence: $r = f\theta$. Fisheye cameras often deviate from the ideal and thus require to be undistorted to an ideal fisheye camera model to work with it further. [1]

2.3 The Dewarping Algorithm

Rather than creating a direct mapping, that 'moves' pixels from the fisheye source to a flat plane, we opt to create an inverse mapping. In this process, we treat the desired dewarped image as an ideal equidistant pinhole camera. We iterate through every pixel in the new target image and project it back into 3D space to determine which pixel it would have ended up as under our fisheye camera assumptions. We then use the colour data from that area in the source image and apply it to the pixel in our target pinhole image.

In this algorithm we ignore intrinsic and extrinsic parameters, assume an equidistant fisheye lens model with distortion that can be modelled using WoodScape's model coefficients.

Algorithm 1: Fisheye Dewarping via Inverse Mapping

Input: Fisheye image I_{src}, Target dimensions (W, H), Target focal length f_t, Principal points $(u_0, v_0)_t$ and $(u_0, v_0)_{src}$, Distortion coefficients $a_1, \ldots, a_n$

Initialise a blank target image (I_{dest}) of size $W \times H$
for *each pixel* (u, v) *in* I_{dest} **do**
  $x \leftarrow \frac{u-u_{0,t}}{f_t}$, $y \leftarrow \frac{v-v_{0,t}}{f_t}$ `// Convert pixel to a ray relative to focal length`
  $r_u \leftarrow \sqrt{x^2+y^2}$ `// Calculate radius with no distortion`
  $\theta \leftarrow \text{atan2}(r_u, 1)$ `// Calculate angle of incidence`
  $r_d \leftarrow a_1\theta + a_2\theta^2 + a_3\theta^3 + a_4\theta^4$ `// Calculate distorted radius`
  if $r_u > 0$ **then**
    $u_{src} \leftarrow (\frac{x}{r_u})r_d + u_{0,src}$, $v_{src} \leftarrow (\frac{y}{r_u})r_d + v_{0,src}$ `// Convert to img plane coordinate`
  else if $r_u = 0$ **then**
    $(u_{src}, v_{src}) \leftarrow (u_{0,src}, v_{0,src})$ `//` $r_u = 0$ `is the exact centre of the image`
  if (u_{src}, v_{src}) *falls within the boundaries of the source image* **then**
    $I_{\text{dest}}(u, v) \leftarrow \text{INTERPOLATE}(I_{src}, u_{src}, v_{src})$
  else
    $I_{\text{dest}}(u, v) \leftarrow 0$ `// out-of-bounds colour`
return Dewarped rectilinear image I_{dest}

We need an interpolate algorithm as it's likely the $(u, v)_{src}$ are decimals whereas we need integer pixels to get colour data from. We interpolate by getting the 4 neighbouring pixels and combining their colour data into one value by how close $(u, v)_{src}$ is to each neighbour.

[1] In OpenCV, the distortion is added by replacing θ with [3, 10]: $\theta_d = \theta + k_1\theta^3 + k_2\theta^5 + k_3\theta^7 + k_4\theta^9$.

In WoodScape, distortion is given in terms of $r(\theta)$ s.t. f is hidden within the coefficients that make up r [3]: $r = a_1\theta + a_2\theta^2 + a_3\theta^3 + a_4\theta^4$.

Algorithm 2: Interpolate

Input: Image I, decimal coordinates (u, v)
Output: Interpolated colour data of 4 neighbouring pixels

$u_1 \leftarrow \text{floor}(u), \quad v_1 \leftarrow \text{floor}(v), \quad u_2 \leftarrow u_1 + 1, \quad v_2 \leftarrow v_1 + 1$
if u_1, v_1, u_2, v_2 *within the image boundaries* **then**
  $du \leftarrow u - u_1, \quad dv \leftarrow v - v_1$
  return
  $(1 - du)(1 - dv)I(u_1, v_1) + du(1 - dv)I(u_2, v_1) + (1 - du)dvI(u_1, v_2) + dudvI(u_2, v_2)$
else
  return 0

3 Report on Technical Progress

3.1 Dewarping

Implementing the dewarping algorithm with arbitrary model parameters yields us promising results. By running this on a mesh image, we see how the algorithm maps the original pixels to the dewarped.

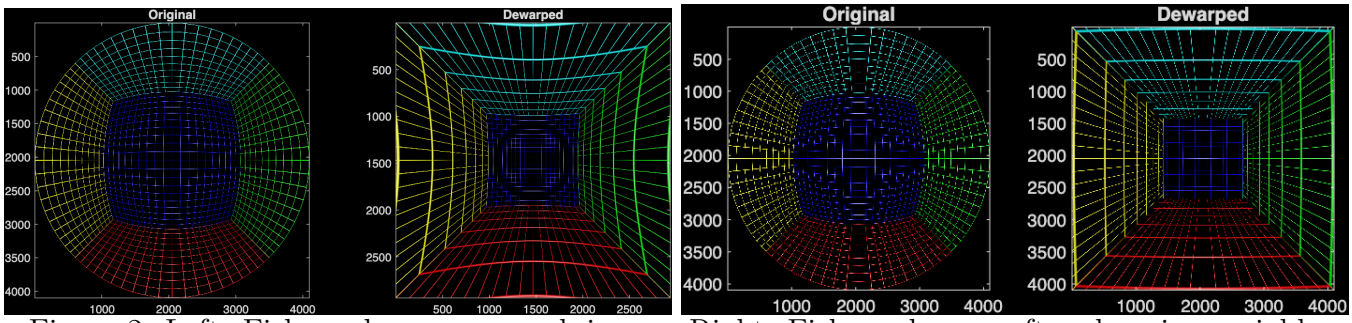


Figure 2: Left: Fisheye dewarp on mesh image. Right: Fisheye dewarp after changing variables.

By manually tweaking the camera model parameters, we get a much better output, indicating we can optimise this fully using other methods.

To successfully continue optimising my fisheye camera model I needed to find a dataset. Using FIORD: A Fisheye Indoor-Outdoor Dataset with LIDAR Ground Truth for 3D Scene Reconstruction and Benchmarking [11], an open source dataset containing thousands of fisheye photos as well as a corresponding ground truth pinhole camera photo. This now lets us calculate a loss value between our models dewarped photos and the ground truth examples.

Calculating Loss Mean Squared Error (MSE) sums the square of the difference of pixel between two images. RMSE is the square root of MSE. Let N be the total scalar elements (all pixels and channels): $\text{RMSE}(I_{\text{pred}}, I_{\text{gt}}) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(I_{\text{pred}}^{(i)} - I_{\text{gt}}^{(i)}\right)^2}$

Algorithm 3: RMSE Computation

Input: Predicted image I_{pred}, Ground truth image I_{gt}
If size$(I_{pred}) \neq$ size(I_{gt}) then Resize I_{pred} to match the spatial dimensions of I_{gt}
Compute squared error: $S \leftarrow (I_{pred} - I_{gt})^2$
Flatten all pixels and channels of S into a vector
return RMSE $L \leftarrow \sqrt{\text{mean}(S)}$

Using arbitrary model parameters, we plot the loss for a batch of 40 images for these initial model parameters and gauge the average RMSE, how noisy the data is, how much the RMSE changes from image to image.

The figure on the right shows the RMSE of first 40 images with initial model parameters, average RMSE: 42.81706.

We take a look at an example of a 'bad' dewarping (image at index 20)

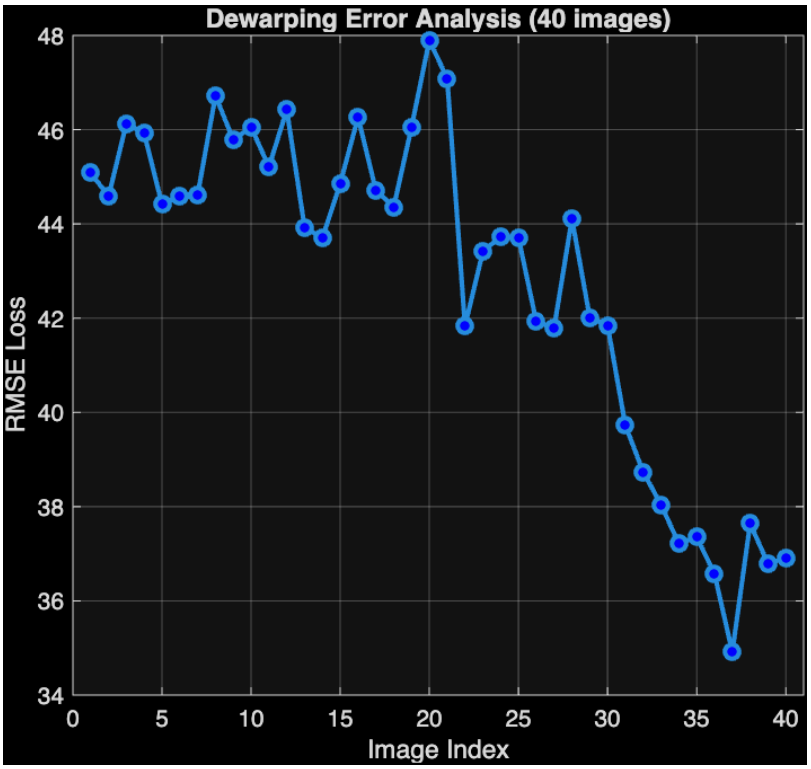




Figure 3: Image at index 20 with loss of 47.9

3.2 Optimisation

3.2.1 Single Image Parameter Ternary Search

We can start by optimising the model on one image to see whether we decrease the loss on just one image for now. We use a ternary search algorithm to find a minima in the loss landscape.

Ternary search is an iterative optimisation method used to locate the minimum of a unimodal function over a closed interval (L, R). At each iteration, the interval is divided into three equal parts using two interior points m_1 and m_2. The function is evaluated at both points, and the portion of the interval that cannot contain the minimum is discarded. Specifically, if $f(m_1) < f(m_2)$, the minimum must lie in $[L, m_2]$; otherwise, it lies in $[m_1, R]$. Repeating this process progressively shrinks the interval around the minimiser. After a fixed number of iterations, the midpoint of the remaining interval is returned as an approximation of the optimal value [12].

Algorithm 4: Ternary Search for Unimodal Function Minimisation

Input: Objective function f, Search interval (L, R), Number of iterations S

for $s \leftarrow 1$ **to** S **do**
  $m_1 \leftarrow L + \frac{R-L}{3}, \quad m_2 \leftarrow R - \frac{R-L}{3}$
  if $f(m_1) < f(m_2)$ **then**
    $R \leftarrow m_2$
  else
    $L \leftarrow m_1$
return Approximate minimiser $x^\star \leftarrow \frac{L+R}{2}$

Using this algorithm and passing the RMSE function as the objective function, we set initial intervals for our model. The parameters that need to be changed are: focal length of source image f_{src}, focal length of the target image f_{t}, parameter coefficients (a_1, a_2, a_3, a_4).

We search each of these parameters individually to to move towards a lower loss, essentially performing a simple form of coordinate descent. Since these variables are not independent, we do multiple iterations, changing the bounds for each variable in each iteration to move to an optimal solution.

Algorithm 5: Coordinate Descent Optimisation of Fisheye Parameters

Input: Source image I_{src}, Ground truth image I_{gt}
Initialise parameter vector Θ with initial guesses
Define search bounds for f_t, f_{src}, and each a_i
Initialise target parameters T and calibration parameters C
for *each optimisation pass* $k = 1$ *to* K **do**
  $f_t \leftarrow \textsc{TernarySearch}(\text{RMSE}(f_t), \text{bounds}_{f_t})$
  Update T with new f_t
  $f_{src} \leftarrow \textsc{TernarySearch}(\text{RMSE}(f_{src}), \text{bounds}_{f_{src}})$
  Update C with new f_{src}
  for $i = 1$ *to* 4 **do**
    Update C with new $a_i \leftarrow \textsc{TernarySearch}(\text{RMSE}(a_i), \text{bounds}_{a_i})$
  Compute and report current RMSE
return Optimised parameters $\Theta = \{f_t, f_{src}, a_1, \ldots, a_4\}$

Running this algorithm on an image we see the before and after:

We rerun and chart our loss with these new parameters to see how well parameters fare with a test set.

The model now performs significantly better with a lower average RMSE, a lower range of loss. We can also take a look at the old 'worst' dewarping image from the initial parameters.

3.2.2 Cross Validation

We can begin to trial cross validation (CV) using ternary search on the dataset to see if we get consistent results on the folds. For now we only use 2 folds. Each fold has multiple images, so the new objective function given to the ternary search is the average RMSE of all the images in a fold, we now attempt to optimise using a set of images rather than just one by using this average RMSE.

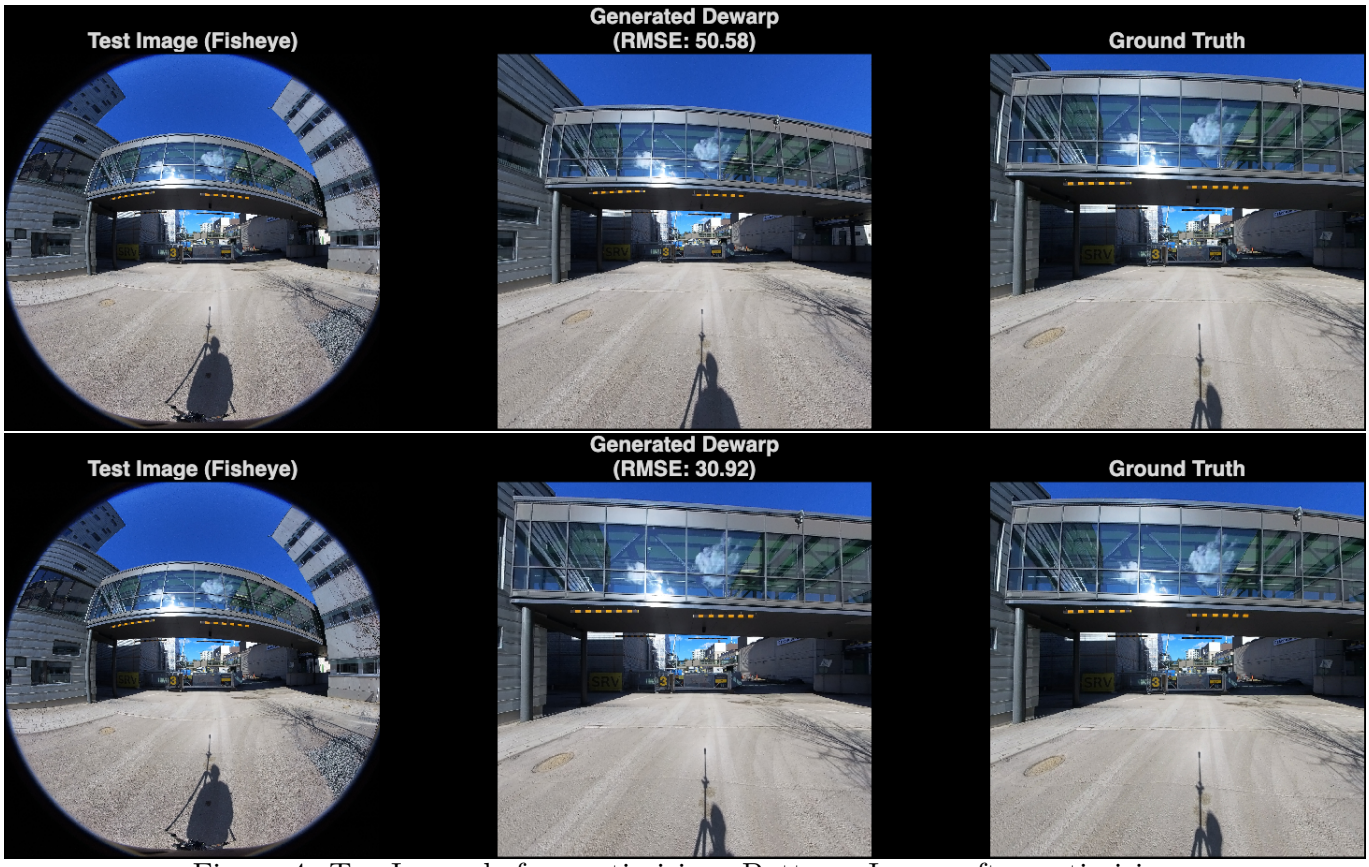


Figure 4: Top:Image before optimising. Bottom: Image after optimising

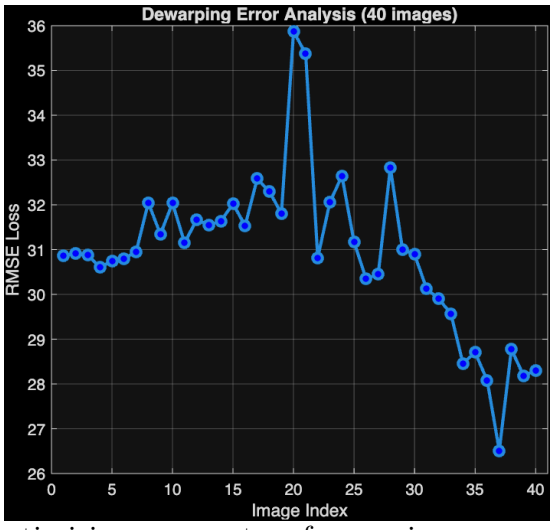


Figure 5: Loss after optimising parameters for one image, average RMSE: 30.93743

Algorithm 6: Two-Fold Cross-Validation for Fisheye Parameter Optimisation

Input: Dataset of paired fisheye and ground truth images

1 Select $N = 40$ valid image pairs, split dataset into two folds: $\mathcal{D}_1$ and $\mathcal{D}_2$

2 **Fold 1:** $\Theta_1 \leftarrow \textsc{Train}(\mathcal{D}_1)$ and $\text{RMSE}_1 \leftarrow \textsc{Test}(\mathcal{D}_2, \Theta_1)$

3 **Fold 2:** $\Theta_2 \leftarrow \textsc{Train}(\mathcal{D}_2)$ and $\text{RMSE}_2 \leftarrow \textsc{Test}(\mathcal{D}_1, \Theta_2)$

4 **return** Computed average cross-validation error: $\text{RMSE}_{\text{avg}} = \frac{\text{RMSE}_1 + \text{RMSE}_2}{2}$



Figure 6: Test loss has improved from optimising off of one image

Algorithm 7: Training via Coordinate Descent on Average RMSE

Input: Training set $\mathcal{D}$

Initialise parameters $\Theta = \{f_t, f_{src}, a_1, \ldots, a_4\}$

for *each parameter in* Θ **do**

  Optimise parameter using TernarySearch to minimise average RMSE over $\mathcal{D}$

return Optimised parameters Θ

Algorithm 8: Testing Procedure with Fixed Fisheye Parameters

Input: Test set $\mathcal{D}$, Fixed parameters $\Theta = \{f_t, f_{src}, a_1, \ldots, a_4\}$

Initialise total error $E \leftarrow 0$

for *each image pair* $(I_{src}, I_{gt}) \in \mathcal{D}$ **do**

  Construct target parameters T using f_t

  Construct calibration parameters C using $f_{src}, a_1, \ldots, a_4$

  $I_{res} \leftarrow \text{Dewarp}(I_{src}, T, C)$

  $E \leftarrow E + \text{RMSE}(I_{res}, I_{gt})$ %$e \leftarrow \text{RMSE}(I_{res}, I_{gt})$

return Average RMSE over $\mathcal{D}$: $\text{RMSE}_{\text{avg}} = \frac{E}{|\mathcal{D}|}$

Metric	Value
Elapsed Time (s)	254.30
Fold 1 Test RMSE	31.70
Fold 2 Test RMSE	34.48
Average Cross-Validation RMSE	33.09

CV Results for Fisheye Parameter Optimisation on a batch of 40 images.

These results indicate that the optimisation procedure successfully found fisheye parameters that generalise reasonably well across two folds of the dataset. The RMSE values for the two folds are comparable, suggesting stability of the learned parameters.

4 3D reconstruction from 2D images

4.1 Planar Face Detection from a Single Greyscale Image

A central requirement of the reconstruction pipeline is identifying the planar faces of a polyhedral object from a single greyscale photograph. Unlike depth-based approaches relying on RGB-D sensors, deep-learning, or multi-view stereo, this method operates exclusively on an intensity image $I : \Omega \subset \mathbb{Z}^2 \to [0, 1]$ of dimensions $H \times W$. This constraint reflects practical scenarios such as rapid prototyping or inspection of manufactured parts, where only a single calibrated view is available.

The algorithm exploits the fact that, under approximately uniform lighting, each visible face of a polyhedron projects to a region of nearly constant intensity, separated from neighbours by sharp

discontinuities at dihedral edges. The pipeline proceeds through five stages:

1. Edge-preserving pre-filtering followed by Canny edge detection;
2. Sequential RANSAC line fitting with cluster splitting and photometric validation;
3. Face polygon construction via boundary rasterisation, connected-component labelling, intensity-based merging, and region growing;
4. Vertex detection through pairwise line intersection with agglomerative clustering;
5. Vertex-to-face assignment with angular sorting.

The complete set of tuneable parameters is summarised in Table 1.

4.1.1 Stage 1: Pre-Filtering and Canny Edge Detection

Two feature types matter for 3D reconstruction: *keypoint features* (e.g. corners, distinctive structures) and *edges*, which indicate object boundaries and enable vanishing-point estimation for extrinsic parameter recovery [13].

The Canny (1986) algorithm computes image gradients to highlight regions of high spatial derivative through four stages [14]:

1. **Noise Reduction** via Gaussian filtering.
2. **Gradient Magnitude and Direction**: gradients computed in x and y are combined per pixel.
3. **Non-maximum Suppression** thins edges by suppressing all gradient values except local maxima.
4. **Hysteresis Thresholding** produces a binary edge map using high/low thresholds.

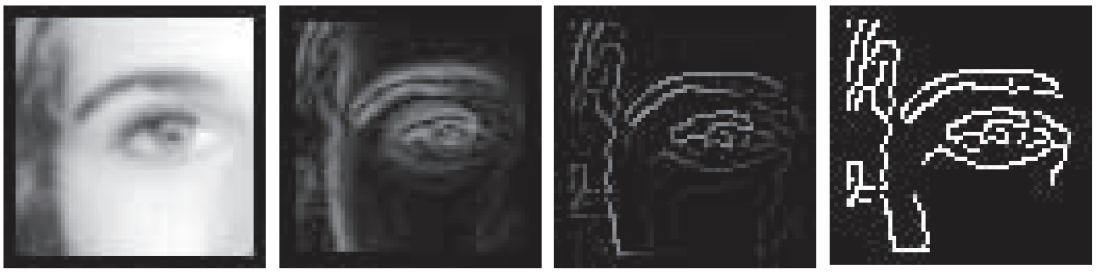


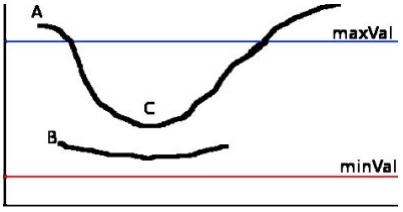
Figure 7: Canny edge detection process [15]

Gaussian Filter. A Gaussian filter applies a weighted average where weights follow a Gaussian distribution, producing smoother results than a uniform mean filter while still preserving edges.

Gradient Computation. The gradient $\nabla I(x,y) = \left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right)$ can be combined directly with Gaussian smoothing: $J_\sigma(x) = [\nabla G_\sigma](x) \cdot I(x), \nabla G_\sigma(x) = \begin{bmatrix} -x & -y \end{bmatrix} \frac{1}{\sigma^2} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$.
The gradient magnitude and direction are [14]: $||\nabla I(x,y)|| = \sqrt{I_x^2 + I_y^2}, \theta(x,y) = \arctan2(I_y, I_x)$

Non-maximum Suppression. A single response per edge is obtained by zeroing all gradient values except local maxima. These maxima correspond to zero crossings of the directional derivative along the gradient direction, equivalent to the Laplacian of Gaussian [13]: $S_\sigma(x) = [\nabla^2 G_\sigma](x) \cdot I(x)$. Sub-pixel peak location uses linear interpolation between adjacent pixels where $S(x)$ changes sign: $x_z = \frac{x_i S(x_j) - x_j S(x_i)}{S(x_j) - S(x_i)}$.

Hysteresis Thresholding. Two thresholds remove noisy low-strength segments [13]: pixels above the high threshold are accepted; pixels between thresholds are accepted only if recursively connected to a strong edge; pixels below the low threshold are rejected.



Edge A is strong, Edge C is connected to A so accepted, Edge B is rejected [14].

4.1.2 Stage 2: Sequential RANSAC Line Fitting

Canny binary edge map E carries no information about which pixels belong to which line. Stage 2 imposes geometric structure by fitting straight-line segments using the RANSAC algorithm [16].

Sequential extraction. Rather than fitting all lines simultaneously, computationally expensive for multi-model estimation [17] , lines are extracted greedily one at a time, with inliers removed after each fit. This works well here because the line count is low and each line accounts for a substantial fraction of remaining edge pixels.

Line model and inlier selection. Two edge pixels $\mathbf{p}_1, \mathbf{p}_2$ are sampled uniformly from the remaining pool $\mathcal{R}$ (rejecting samples closer than 5 px). The line parametrised in implicit normal form:

$$ax + by + c = 0, \qquad a = -(y_2 - y_1), \quad b = x_2 - x_1, \quad c = -(ax_1 + by_1), \tag{1}$$

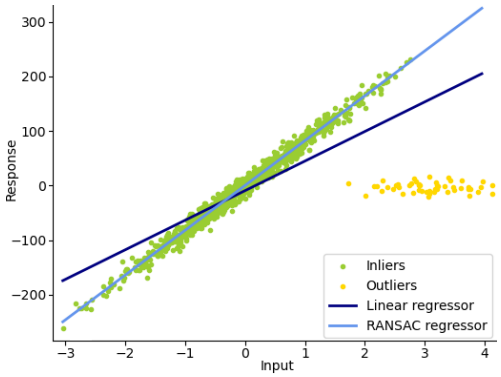
normalised so $\sqrt{a^2 + b^2} = 1$. The perpendicular distance becomes $d(\mathbf{q}) = |aq_x + bq_y + c|$, and inliers are defined: $\mathcal{I} = \{\mathbf{q} \in \mathcal{R} : d(\mathbf{q}) < d_{\text{RANSAC}}\}$.

Adaptive iteration count. Following RANSAC [16], the iteration count is:

$$N_{\text{iter}} = \left\lceil \frac{\ln(1 - p_{\text{conf}})}{\ln\left(1 - w^2\right)} \right\rceil, \qquad w = \frac{n_{\min}}{|\mathcal{R}|}, \tag{2}$$

with $p_{\text{conf}} = 0.99$ (using $n_{\min}$, the minimum acceptable inlier count). The count is recomputed adaptively as $|\mathcal{R}|$ shrinks and clamped to $[N_{\text{user}}, 50{,}000]$.

PCA-based line refinement. The two-point line is approximate. Once the best inlier set $\mathcal{I}^*$ is found, [18] refines the fit: with centred coordinate matrix $\mathbf{X}$, the SVD $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ yields the smallest singular vector $\mathbf{v}_2$ (the line normal). Refined parameters are $(a, b) = \mathbf{v}_2^\top$ and $c = -(a\mu_x + b\mu_y)$. This minimises squared-sum perpendicular distances to inliers.



RANSAC vs a linear regressor [19].

Cluster splitting. When two physically separate but collinear face boundaries exist, RANSAC may bridge them with a phantom segment. To prevent this, inliers are projected onto the line direction $\mathbf{d} = (b, -a)$ and sorted; gaps exceeding $g_{\max}$ (default 30 px) split the inlier set into independent clusters.

Photometric contrast validation. Texture edges, shadows, and noise can yield geometrically valid but spurious line fits. Sampling pixels at intervals along the segment, intensities are measured at perpendicular offsets ± 5 px. The contrast metric is

$$\kappa = \Big|\text{median}\big\{I(\mathbf{s} + 5\hat{\mathbf{n}})\big\} - \text{median}\big\{I(\mathbf{s} - 5\hat{\mathbf{n}})\big\}\Big|, \tag{3}$$

with $\hat{\mathbf{n}} = (a, b)$. Medians are robust against outliers. Segments are accepted only if $\kappa \geq \kappa_{\min}$ and length exceeds $L_{\min}$. Together, we maximise the chance that the line represents a physical intensity boundary between faces.

The full RANSAC line fitting algorithm can be found in algorithm 9



Figure 8: A RANSAC line not contributed to a face boundary: rejected by photometric validation.

Note that all inliers are removed regardless of whether segments pass validation - otherwise the same collinear points would be repeatedly selected (infinite loop).

4.1.3 Stage 2b: Line Extension Toward Intersections and Image Edges

The Canny detector frequently terminates edge chains several pixels short of true vertices, since Gaussian smoothing attenuates response near corners where gradient orientation changes abruptly [14]. To close these gaps, each segment endpoint is extended along its line direction by up to d_{ext} pixels, snapping to the closest of:

1. **Line-line intersection.** For each pair (i, j), the intersection is computed via Cramer's rule:

$$\Delta_{ij} = a_i b_j - a_j b_i, \qquad x_{\text{int}} = \frac{b_i c_j - b_j c_i}{\Delta_{ij}}, \qquad y_{\text{int}} = \frac{a_j c_i - a_i c_j}{\Delta_{ij}}, \tag{4}$$

   treating $|\Delta_{ij}| < 10^{-6}$ as parallel. The intersection is accepted only if it lies within d_{ext} along the extension direction and is geometrically close to segment j, preventing snaps to unrelated distant lines.

2. **Image boundary.** The endpoint ray $\mathbf{r}(t) = \mathbf{p}_{\text{end}} + t\hat{\mathbf{u}}$ is intersected with each image edge; the smallest positive $t \leq d_{\text{ext}}$ wins.

If no target is found within d_{ext}, the endpoint is unchanged. This step can be disabled when face boundaries are clearly contrasted.

4.1.4 Stage 3: Face Construction via Boundary Rasterisation and Region Growing

Stage 3 converts the wireframe into a partition of the image into closed labelled face regions. Because the wireframe is not guaranteed to form a perfectly closed planar graph, the classic planar subdivision algorithm [20] is replaced by a more robust rasterise and flood-fill strategy.

Algorithm 9: Sequential RANSAC Line Fitting with Cluster Splitting

Input: Edge pixel set $\mathcal{E}$, parameters d_{RANSAC}, $n_{\min}$, $L_{\min}$, $\kappa_{\min}$, $K_{\max}$, $g_{\max}$
Output: Set of line segments $\mathcal{L} = \{(\mathbf{p}_1^{(k)}, \mathbf{p}_2^{(k)}, a_k, b_k, c_k)\}_{k=1}^{N_l}$

$\mathcal{R} \leftarrow \mathcal{E}$ `// Remaining edge pixels`
$\mathcal{L} \leftarrow \emptyset$
for $k = 1, 2, \ldots, K_{\max}$ **do**
  if $|\mathcal{R}| < n_{\min}$ **then**
    break
  Compute adaptive iteration count N_{iter} via Equation (2)
  $\mathcal{I}^* \leftarrow \emptyset$ `// Best inlier set found so far`
  for $t = 1, 2, \ldots, N_{\text{iter}}$ **do**
    Sample $\mathbf{p}_1, \mathbf{p}_2 \in \mathcal{R}$ uniformly at random
    if $\|\mathbf{p}_2 - \mathbf{p}_1\| < 5$ **then**
      continue `// Reject degenerate sample`
    Compute normalised line (a, b, c) through $\mathbf{p}_1, \mathbf{p}_2$ via Equation (1)
    $\mathcal{I} \leftarrow \{\mathbf{q} \in \mathcal{R} : |aq_x + bq_y + c| < d_{\text{RANSAC}}\}$
    if $|\mathcal{I}| > |\mathcal{I}^*|$ **then**
      $\mathcal{I}^* \leftarrow \mathcal{I}$
  if $|\mathcal{I}^*| < n_{\min}$ **then**
    break `// No further lines detectable`
  Refit (a, b, c) to all points in $\mathcal{I}^*$ via PCA (SVD of centred coordinates)
  Project $\mathcal{I}^*$ onto line direction $\mathbf{d} = (b, -a)$; sort projections in ascending order
  Split into contiguous clusters $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_m$ at gaps exceeding $g_{\max}$
  for *each cluster* $\mathcal{C}_i$, $i = 1, \ldots, m$ **do**
    if $|\mathcal{C}_i| < n_{\min}$ **then**
      continue
    Compute segment endpoints from extremal projections within $\mathcal{C}_i$
    if *segment length* $< L_{\min}$ **then**
      continue
    Compute photometric contrast κ via Equation (3)
    if $\kappa \geq \kappa_{\min}$ **then**
      Append segment (endpoints and line parameters) to $\mathcal{L}$
  $\mathcal{R} \leftarrow \mathcal{R} \setminus \mathcal{I}^*$ `// Remove all inliers regardless of acceptance`
return $\mathcal{L}$

Step 3a: Object mask construction. A binary mask $\mathcal{M}_{\text{obj}}$ is built by thresholding ($I > 0.03$) to separate object from dark background. Morphological closing (3 px dilation then erosion) refines the mask, and BFS flood-fill from the image border classifies background pixels; the complement is the filled object mask.

Step 3b: Silhouette boundary extraction. The outer boundary of the object mask is the set difference between the mask and its erosion:

$$\mathcal{B}_{\text{sil}} = \mathcal{M}_{\text{obj}} \setminus \text{erode}\big(\mathcal{M}_{\text{obj}}, \mathbf{D}_2\big), \tag{5}$$

where $\mathbf{D}_2$ is a disk of radius 2.

Step 3c: Boundary mask rasterisation. Each line in $\mathcal{L}$ is rasterised onto $\mathcal{B}_{\text{lines}} \in \{0,1\}^{H\times W}$ by sampling along the segment, rounding to integer grid, and stamping a 5×5 px square. Endpoints are extended 8 px beyond. Thick rasterisation prevents flood-fill leakage at vertex misalignments. The final boundary is:

$$\mathcal{B} = \big[\text{dilate}\big(\mathcal{B}_{\text{lines}},\, \mathbf{D}_2\big) \;\cup\; \mathcal{B}_{\text{sil}}\big] \;\cap\; \mathcal{M}_{\text{obj}}. \tag{6}$$

The intersection with $\mathcal{M}_{\text{obj}}$ ensures the boundary mask does not extend beyond the object region.

Step 3d: Connected-component labelling. The fillable region $\mathcal{F} = \mathcal{M}_{\text{obj}} \setminus \mathcal{B}$ is partitioned via 4-connected flood-fill labelling [21], producing a label map $L : \Omega \to \{0, 1, \ldots, N_r\}$ where $L(\mathbf{p}) = 0$ for pixels on the boundary or outside the object mask and N_r is the total number of connected components. Each region is a candidate face, with an unlabelled boundary band ~ 9 px wide between adjacent regions.

Step 3e: Intensity-based region merging. Texture gradients can fragment a single face into multiple regions. An iterative merging step [22] builds an adjacency graph by scanning 4-connected pixel pairs, then merges adjacent region pairs (r_i, r_j) whose median-intensity ($|\bar{I}_{r_i} - \bar{I}_{r_j}|$) difference falls below τ_m. Median intensities provide outlier robustness. The process repeats until no merges occur. Algorithm 10 formalises the procedure.

Algorithm 10: Iterative Merging of Similar Adjacent Regions

Input: Label map L, image I, intensity threshold τ_m
Output: Updated label map L with over-segmented regions consolidated

repeat
  changed $\leftarrow$ FALSE
  Compute median intensity $\bar{I}_r \leftarrow \text{median}\big\{I(\mathbf{p}) : L(\mathbf{p}) = r\big\}$ for each region $r > 0$
  Construct adjacency set
    $\mathcal{A} = \big\{(r_i, r_j) : \exists\, \mathbf{p}, \mathbf{q}$ 4-adjacent with $L(\mathbf{p}) = r_i,\, L(\mathbf{q}) = r_j,\, r_i \neq r_j\big\}$
  for *each pair* $(r_i, r_j) \in \mathcal{A}$ *with* $r_i < r_j$ **do**
    if $\big|\bar{I}_{r_i} - \bar{I}_{r_j}\big| < \tau_m$ **then**
      $L(\mathbf{p}) \leftarrow r_i$ for all $\mathbf{p}$ with $L(\mathbf{p}) = r_j$ `// Absorb` r_j `into` r_i
      Recompute $\bar{I}_{r_i}$
      changed $\leftarrow$ TRUE
until *changed* = FALSE
Relabel regions consecutively as $\{1, 2, \ldots, N_f\}$
return L

Step 3f: Boundary pixel assignment via simultaneous region growing. The thick boundary mask $\mathcal{B}$ leaves an unlabelled band ~ 9 px wide between every adjacent face pair. To produce a seamless tiling, all labelled regions are dilated by one pixel per iteration with the 4-connected element $\mathbf{S}_4$, claiming only pixels within $\mathcal{M}_{\text{obj}}$ that are currently unlabelled. When multiple regions reach the same pixel in one iteration, processing order resolves the conflict.

Algorithm 11: Boundary Pixel Assignment via Simultaneous Region Growing

Input: Label map L, object mask $\mathcal{M}_{\text{obj}}$, structuring element $\mathbf{S}_4$
Output: Updated label map L with all pixels in $\mathcal{M}_{\text{obj}}$ assigned to a face

for $t = 1, 2, \ldots, T_{\max}$ **do**
  $\mathcal{U} \leftarrow \{\mathbf{p} \in \mathcal{M}_{\text{obj}} : L(\mathbf{p}) = 0\}$ `// Currently unlabelled pixels`
  if $\mathcal{U} = \emptyset$ **then**
    break `// All pixels assigned`
  $C \leftarrow \mathbf{0}_{H \times W}$ `// Claim map for this iteration`
  for *each region label* $r \in \{1, 2, \ldots, N_f\}$ **do**
    $D_r \leftarrow \text{dilate}(\{L = r\}, \mathbf{S}_4)$ `// Expand region` r `by one pixel`
    $\mathcal{N}_r \leftarrow D_r \cap \mathcal{U} \cap \{C = 0\}$ `// Newly reachable and unclaimed pixels`
    $C(\mathcal{N}_r) \leftarrow r$ `// Claim these pixels for region` r
  if $\{C > 0\} = \emptyset$ **then**
    break `// No progress; isolated pixels remain`
  $L(\{C > 0\}) \leftarrow C(\{C > 0\})$ `// Commit all claims to the label map`
return L

At most ~ 5 iterations are needed since the band is ≤ 9 px wide.

Step 3g: Face extraction and boundary tracing. Regions with area below $A_{\min}$ are discarded as noise. For each surviving face f, the median intensity ($\bar{I}_f = \text{median}\{I(\mathbf{p}) : L(\mathbf{p}) = f\}$) is recorded (needed for normal estimation), and the boundary polygon is extracted by Moore neighbourhood tracing [23], yielding boundary pixels in clockwise order.

Step 3h: Whole-region absorption of sub-threshold components. Area-thresholding leaves orphaned pixels where small components were discarded, re-opening interior gaps. A naive application of Algorithm 11 would let arbitrary neighbours dilate into these gaps. This errors when small components represent decorative features, e.g. a door that should be absorbed by only the wall, not split between wall and floor.

The absorption step instead snapshots the pre-filter label map as L_{full} and absorbs each rejected component *as a single unit* into the surviving face that shares the longest 4-connected perimeter with it. The rejected set is identified by overlap:

$$\mathcal{R} = \{L_{\text{full}}(\mathbf{p}) : \mathbf{p} \in \mathcal{M}_{\text{obj}},\ L(\mathbf{p}) = 0,\ L_{\text{full}}(\mathbf{p}) > 0\}. \tag{7}$$

For each $r \in \mathcal{R}$ with mask $M_r = \{L_{\text{full}} = r\}$, the neighbour band is

$$\mathcal{B}_r = \big(\text{dilate}(M_r, \mathbf{S}_4) \setminus M_r\big) \cap \{L > 0\}, \tag{8}$$

and the receiving face is

$$f^*(r) = \underset{f}{\text{argmax}} \big|\{\mathbf{p} \in \mathcal{B}_r : L(\mathbf{p}) = f\}\big|. \tag{9}$$

Components whose neighbours are all rejected ($\mathcal{B}_r = \emptyset$) are deferred until a neighbour becomes absorbed, iterating until no progress. A fallback to Algorithm 11 handles isolated orphans touching only the silhouette.

Because absorption is whole-region, the boundary between an absorbed component and its non-receiving neighbours follows the original RANSAC line, which is preserved by the line-pruning step of Stage 3j. The vertical lines that previously separated, e.g. a door from the wall correctly become interior to the merged wall and are dropped.

Algorithm 12: Whole-Region Absorption of Sub-Threshold Components

Input: Filtered label map L, full label map L_{full}, object mask $\mathcal{M}_{\text{obj}}$, structuring element $\mathbf{S}_4$

Compute $\mathcal{R}$ via Equation (7) `// CC IDs whose pixels are unlabelled in` L
progress $\leftarrow$ TRUE
while *progress* ***and*** $\mathcal{R} \neq \emptyset$ **do**
  progress $\leftarrow$ FALSE
  $\mathcal{R}_{\text{next}} \leftarrow \emptyset$
  for *each* $r \in \mathcal{R}$ **do**
    $M_r \leftarrow \{\mathbf{p} : L_{\text{full}}(\mathbf{p}) = r\}$
    Compute $\mathcal{B}_r$ via Equation (8)
    if $\mathcal{B}_r = \emptyset$ **then**
      $\mathcal{R}_{\text{next}} \leftarrow \mathcal{R}_{\text{next}} \cup \{r\}$ `// Defer: neighbours all still rejected`
      continue
    $f^* \leftarrow \operatorname{argmax}_f \big|\{\mathbf{p} \in \mathcal{B}_r : L(\mathbf{p}) = f\}\big|$
    $L(\mathbf{p}) \leftarrow f^*$ for all $\mathbf{p} \in M_r$
    progress $\leftarrow$ TRUE
  $\mathcal{R} \leftarrow \mathcal{R}_{\text{next}}$
if $\{\mathbf{p} \in \mathcal{M}_{\text{obj}} : L(\mathbf{p}) = 0\} \neq \emptyset$ **then**
  Apply Algorithm 11 to L restricted to $\mathcal{M}_{\text{obj}}$ `// Mop up isolated orphans`
return label map L with every rejected component absorbed into one surviving face

Step 3i: Absorption of fully-enclosed faces. A closed loop of segments around an interior feature (e.g. a window on a wall) can produce a small region completely enclosed by a larger face. Unlike Step 3e, the intensity contrast may correctly exceed τ_m (a dark window on a light wall), so intensity merging declines. However, such islands don't represent independent polyhedron faces and corrupt subsequent normal estimation.

$$\text{touches_exterior}(f) = \big[\text{dilate}(\{L = f\}, \mathbf{S}_4) \cap \neg\mathcal{M}_{\text{obj}} \neq \emptyset\big]. \tag{10}$$

A face is *enclosed* iff:

$$\text{enclosed}(f) = \neg\,\text{touches_exterior}(f) \wedge |\mathcal{N}(f)| = 1. \tag{11}$$

Each enclosed face is absorbed into its sole neighbour. Because nested enclosures may exist, the test runs as a fixed-point loop, terminating in at most N_f passes.

Algorithm 13: Fixed-Point Absorption of Enclosed Faces

Input: Label map L, object mask $\mathcal{M}_{\text{obj}}$, structuring element $\mathbf{S}_4$

repeat
  changed $\leftarrow$ FALSE
  Compute neighbour sets $\mathcal{N}(f) \leftarrow \{f' \neq f : \exists\, \mathbf{p}, \mathbf{q}$ 4-adjacent with $L(\mathbf{p}) = f$, $L(\mathbf{q}) = f'\}$ for every face f
  Compute touches_exterior(f) via Equation (10) for every face f
  for *each face* f **do**
    if $\neg$ touches_exterior(f) ***and*** $|\mathcal{N}(f)| = 1$ **then**
      $f' \leftarrow$ the unique element of $\mathcal{N}(f)$
      $L(\mathbf{p}) \leftarrow f'$ for all $\mathbf{p}$ with $L(\mathbf{p}) = f$ `// Absorb island into surround`
      changed $\leftarrow$ TRUE
until *changed* = FALSE

Stage 3j: Label-Driven Line Pruning After Steps 3e and 3i merge regions, several RANSAC lines may now lie entirely *inside* a single labelled face. Each such redundant line contributes ghost vertices when intersected with other lines. To eliminate them, every line is classified against the final L.

For each line ℓ_i, the unit normal $\hat{\mathbf{n}} = (a_i, b_i)$ is sampled along the segment at parametrised positions:

$$\mathbf{s}(t) = \mathbf{p}_1 + t(\mathbf{p}_2 - \mathbf{p}_1), \quad t \in \text{linspace}(0.1, 0.9, n_s), \tag{12}$$

with $n_s = \text{clamp}(\lceil \|\mathbf{p}_2 - \mathbf{p}_1\|/3 \rceil, 5, 80)$. At each sample, labels $L^{\pm} = L(\mathbf{s} \pm \delta\hat{\mathbf{n}})$ are read with $\delta = \max(3, \lfloor d_{\text{snap}}/3 \rfloor)$, large enough to clear the boundary band but small enough to stay inside neighbour faces. Counters are updated:

$$\begin{aligned} n_{\text{diff}} \mathrel{+}= & \mathbb{1}\big[L^+ > 0 \wedge L^- > 0 \wedge L^+ \neq L^-\big], \\ n_{\text{sil}} \mathrel{+}= & \mathbb{1}\big[(L^+ = 0) \oplus (L^- = 0)\big], \\ n_{\text{same}} \mathrel{+}= & \mathbb{1}\big[L^+ > 0 \wedge L^- > 0 \wedge L^+ = L^-\big]. \end{aligned} \tag{13}$$

The line is retained iff:

$$\frac{n_{\text{diff}} + n_{\text{sil}}}{n_{\text{diff}} + n_{\text{sil}} + n_{\text{same}}} \geq \tau_b. \tag{14}$$

The default $\tau_b = 0.30$ is lenient; corner overshoots and boundary-band overlaps reduce the ratio, so stricter thresholds drop true boundaries.

Algorithm 14: Label-Driven Line Classification

Input: Line set $\mathcal{L}$, label map L, side offset δ, keep threshold τ_b
Output: Pruned line set $\mathcal{L}' \subseteq \mathcal{L}$

$\mathcal{L}' \leftarrow \emptyset$
for *each line* $\ell_i \in \mathcal{L}$ **do**
  $n_s \leftarrow \text{clamp}(\lceil \|\mathbf{p}_2 - \mathbf{p}_1\|/3 \rceil, 5, 80)$
  $n_{\text{diff}}, n_{\text{sil}}, n_{\text{same}} \leftarrow 0$
  for $t \in \text{linspace}(0.1, 0.9, n_s)$ **do**
    $\mathbf{s} \leftarrow \mathbf{p}_1 + t(\mathbf{p}_2 - \mathbf{p}_1), \quad \mathbf{s}^{\pm} \leftarrow \text{round}(\mathbf{s} \pm \delta(a_i, b_i))$
    if $\mathbf{s}^+ \notin \Omega$ ***or*** $\mathbf{s}^- \notin \Omega$ **then**
      continue
    $L^+ \leftarrow L(\mathbf{s}^+), \quad L^- \leftarrow L(\mathbf{s}^-)$
    Update $(n_{\text{diff}}, n_{\text{sil}}, n_{\text{same}})$ via Equation (13)
  $n_{\text{tot}} \leftarrow n_{\text{diff}} + n_{\text{sil}} + n_{\text{same}}$
  if $n_{\text{tot}} > 0$ ***and*** $(n_{\text{diff}} + n_{\text{sil}})/n_{\text{tot}} \geq \tau_b$ **then**
    $\mathcal{L}' \leftarrow \mathcal{L}' \cup \{\ell_i\}$
return $\mathcal{L}'$

The pruned set $\mathcal{L}'$ replaces $\mathcal{L}$ in all downstream stages: the diagnostic visualisations, the vertex detection of Stage 3, and the result struct returned to the caller all see only the boundary-confirmed lines, so the output is internally consistent.

Empirically, on the synthetic test polyhedra the classifier reduces the line count from typically 20–25 to between 8 and 14, and on indoor photographs from 40–60 down to roughly 20, with no observed loss of face boundaries.

4.1.5 Stage 4: Vertex Detection via Line-Line Intersection

All $\binom{N_{l'}}{2}$ pairwise intersections of the pruned line set are computed via Equation (4). A candidate is accepted if (i) its perpendicular distance to each contributing finite segment is within $2\,d_{\text{snap}}$ (with endpoint tolerance), and (ii) it falls within 5 px of the image bounds.

Nearby candidates are merged by *union-find* single-linkage clustering with threshold d_{snap}. Each candidate starts in its own component; pairs within d_{snap} are unioned with path compression. Component centroids form the final N_v vertex positions.

Union-find is necessary because greedy linkage is non-transitive: three candidates with pairwise gaps 5, 5, 10.1 px under threshold 12 would collapse into two clusters greedily, but should form one.

4.1.6 Stage 5: Vertex-to-Face Assignment

This stage assigns the detected vertices to face regions. For each face f, the binary mask $\{L = f\}$ is dilated by d_{snap} pixels (since vertices lie on boundaries, not face interiors). Vertices whose rounded coordinates fall within the dilated mask are assigned to f.

Vertices are then sorted by angle relative to the face's vertex centroid:

$$\alpha_v = \text{atan2}(y_v - \bar{y}_f, x_v - \bar{x}_f),$$

producing counter-clockwise winding. This assumes face convexity in projection, which holds for visible faces of a convex polyhedron.

4.1.7 Planar detection results

We can show this method generalises with many different examples, including, such as the house used above. By changing different parameters, typically only requiring to change the minimum face area, we can generate different planar detection results for different scenarios, including synthetic polyhedron data that follow the structural assumptions we will put in place in section 4.2

The full pipeline of stages can be visualised and viewed in the appendix (section 8)

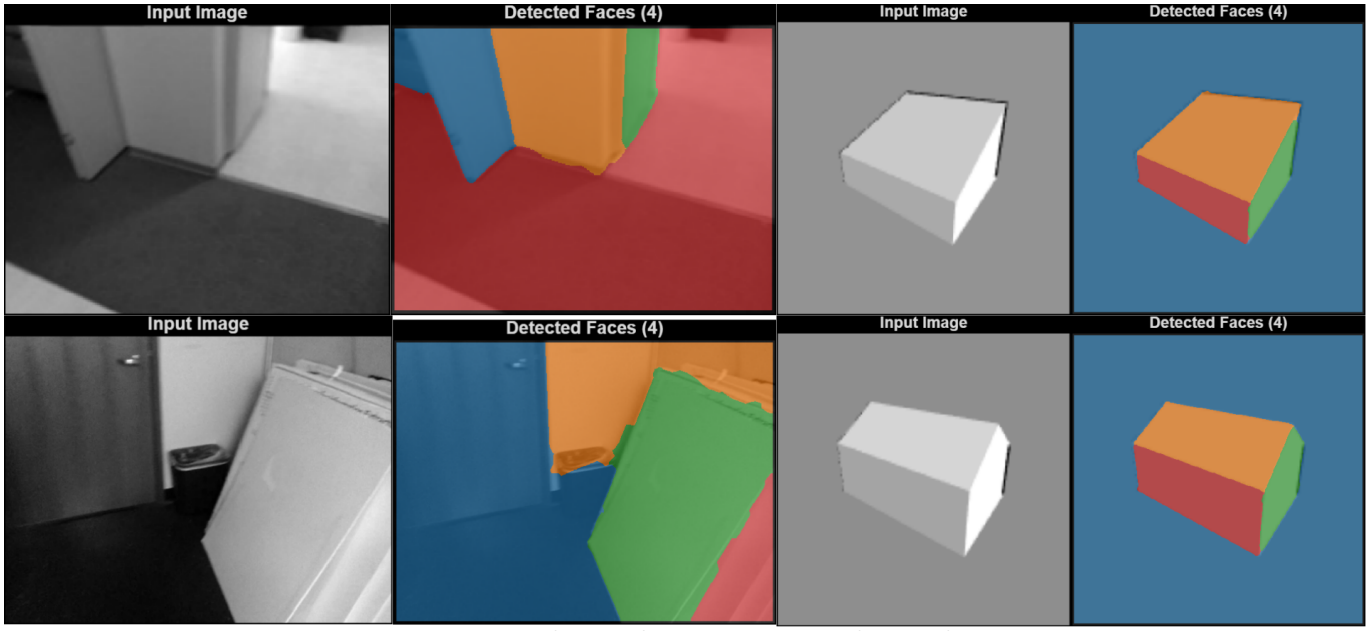


Figure 9: Planar detection example results

4.1.8 Summary of Parameters

The algorithm exposes a number of tuneable parameters that control the behaviour of each stage. Table 1 provides a complete listing together with the default values used for real-image experiments in this work. These defaults were determined empirically on a development set of indoor room photographs and synthetic polyhedron renderings.

Table 1: Complete set of tuneable parameters for the planar face detection algorithm, together with the default values used for real-image experiments.

Parameter	Symbol	Default	Description
Canny Gaussian σ	σ	2.0	Scale of gradient smoothing kernel
Canny low threshold	τ_L	0.06	Hysteresis low threshold
Canny high threshold	τ_H	0.18	Hysteresis high threshold
RANSAC inlier distance	d_{RANSAC}	2.0 px	Maximum point-to-line distance for inliers
Minimum inlier count	$n_{\min}$	50	Fewest inliers to accept a line
Maximum line count	$K_{\max}$	30	Upper bound on lines extracted
Minimum segment length	$L_{\min}$	50 px	Shortest acceptable segment
Minimum contrast	$\kappa_{\min}$	0.01	Min cross-edge intensity difference
Line extension maximum	d_{ext}	40 px	Max endpoint extension distance
Vertex snap distance	d_{snap}	20 px	Vertex clustering and assignment radius
Region merge threshold	τ_m	0.08	Max intensity difference for merging
Minimum face area	$A_{\min}$	3000 px^2	Smallest valid face region
Line-keep threshold	τ_b	0.30	Min fraction of differently-labelled samples per line

4.2 Surface Normal Estimation via Piecewise-Planar Shape-from-Shading

Given the labelled partition of the image into planar faces produced by Section 4.1, the second module recovers, for each detected face $f \in \{1, \dots, N_f\}$, a unit surface normal $\mathbf{n}_f \in \mathbb{S}^2$, a single global light direction $\mathbf{l} \in \mathbb{S}^2$, and a scalar albedo $\rho \in \mathbb{R}_{>0}$. This is a piecewise-planar instance of the classical shape-from-shading problem [24], severely ill-posed in general but tractable here because the dense per-pixel normal field collapses to N_f unknowns, constrained by the assumption that the visible faces all belong to a *single convex polyhedron viewed from outside*. Visibility, plane separation, non-coplanar dihedrals, and shared-edge coherence are all encoded as differentiable penalties in a smooth global cost.

4.2.1 Reflectance Model and Structural Assumptions

The photometric forward model is Lambertian [25]: for a face with outward normal $\mathbf{n}_f$ illuminated by a single distant source along $\mathbf{l}$,

$$I_f \;=\; \rho \, \max\big(\mathbf{n}_f \cdot \mathbf{l},\, 0\big), \tag{15}$$

where the $\max(\cdot, 0)$ encodes that faces turned away from the source emit no reflected light.

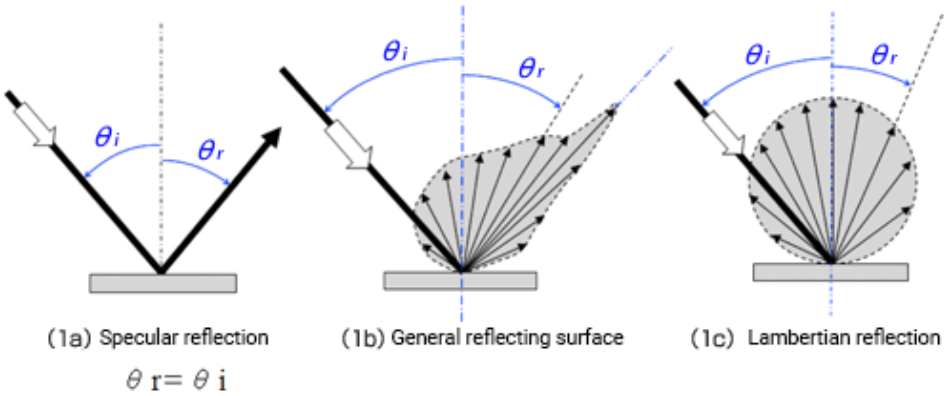


Figure 10: Lambertian model surface scatters light equally in all directions. Image from [26]

The implementation defaults to `KnownAlbedo` mode ($\rho = 1$ fixed); when disabled, ρ is constrained to $[0.01, 10]$ during the lighting update. We further assume a single distant source on the upper hemisphere ($l_z > 0$), purely diffuse uniform reflectance, no cast or self-shadows, orthographic projection along a known view direction $\mathbf{v} \in \mathbb{S}^2$ (default $\mathbf{v} = (0, 1, 0)^\top$), and a planar horizontal ground with

normal $\mathbf{n}_g = (0, 0, 1)^\top$. The ground anchor breaks the bas-relief ambiguity [27] and fixes the global scale of $\rho\mathbf{l}$. With $\mathbf{u} = \mathbf{n}_g$ as world-up, the estimator constructs an orthonormal camera-frame triple $(\mathbf{r}, \mathbf{u}_c, \mathbf{v})$:

$$\mathbf{r} \;=\; \tfrac{\mathbf{v}\times\mathbf{u}}{\|\mathbf{v}\times\mathbf{u}\|}, \qquad \mathbf{u}_c \;=\; \mathbf{r}\times\mathbf{v}, \tag{16}$$

where $\mathbf{r}$ spans the image-plane horizontal axis and $\mathbf{u}_c$ the vertical axis (opposite to image row index). If $\mathbf{v} \parallel \mathbf{u}$, $\mathbf{u}$ is replaced by $\hat{\mathbf{x}}$, then $\hat{\mathbf{y}}$ if needed. Image-plane displacements $(\Delta x, \Delta y)$ lift to 3D as $\Delta x\,\mathbf{r} - \Delta y\,\mathbf{u}_c$ (the negative sign converts row-down to height-up).

The single convex polyhedron assumption is encoded as four structural relations on the recovered normals; these are turned into smooth penalties in Section 4.2.6:

(P_1) **Visibility.** Every detected face is oriented toward the camera: $\mathbf{n}_f \cdot \mathbf{v} \le -\varepsilon_v$ for a margin ε_v (default 0.05). Back-facing surfaces cannot be captured.

(P_2) **Plane separation.** For distinct faces $f \neq g$, $\mathbf{n}_f \cdot (\mathbf{c}_g - \mathbf{c}_f) \le 0$, where $\mathbf{c}_f, \mathbf{c}_g$ are face centroids. The anchor is excluded from the role of f: in this codebase the ground is a separate floor surface, and the polyhedron lies on the same side as $\mathbf{n}_g$.

(P_3) **Non-coplanar dihedrals.** Adjacent faces (i, j) satisfy $|\mathbf{n}_i \cdot \mathbf{n}_j| \le \cos\theta_{\min}$ for $\theta_{\min}$ (default 10°), excluding both $\theta = 0$ and the antiparallel $\theta = \pi$ degeneracy while leaving $(\theta_{\min}, \pi - \theta_{\min})$ unconstrained.

(P_4) **Shared-edge coherence.** The 3D edge direction shared by adjacent faces i, j is parallel to $\mathbf{n}_i \times \mathbf{n}_j$; its image-plane projection must be parallel to the observed boundary direction $\mathbf{d}_e^{(2D)}$. Under orthographic projection, an isolated edge has a free depth slope and so does not reduce a face's degrees of freedom.

An earlier prototype additionally assumed ground-adjacent faces are vertical walls and that dihedrals come from a discrete architectural set $\{45°, 60°, 90°, 120°, 135°, 150°, 180°\}$. Although seemingly restrictive in a helpful way, (P_1)-(P_4) together with (15) and the ground anchor already suffice to identify the normal field up to numerical noise; the extra priors limited applicability without improving accuracy.

4.2.2 Stage N1: Face Adjacency Graph and Boundary Pixel Collection

From the integer label map $L : \Omega \to \{0, 1, \ldots, N_f\}$, an adjacency matrix $A \in \{0,1\}^{N_f \times N_f}$ and boundary-pixel lists are built by a 4-connected scan: for each pair of 4-neighbours $(\mathbf{p}, \mathbf{q})$ with $L(\mathbf{p}) = a > 0$, $L(\mathbf{q}) = b > 0$, $a \neq b$, the pair $(\min(a,b), \max(a,b))$ is recorded as adjacent and the subpixel midpoint appended to $\mathcal{B}_{ab}$. The upper triangle of A yields the ordered adjacent-pair set $\mathcal{P} = \{(f_i, f_j)\}_{e=1}^{N_e}$, each with boundary cloud $\mathcal{B}_e \subset \mathbb{R}^2$.

4.2.3 Stage N2: Anchor Face Identification

One face is designated as the *anchor*, with normal fixed to $\mathbf{n}_g$; this fixes the scale of $\rho\mathbf{l}$ and is the base for the convex-polyhedron priors. For the scenes considered, the most reliable anchor is the ground plane, typically the largest face and lowest in the image. Unless the caller supplies an explicit index, the anchor is the argmax of

$$s_f \;=\; \tfrac{|\Omega_f|}{\max_{f'} |\Omega_{f'}|} \cdot \tfrac{\bar{r}_f}{H}, \tag{17}$$

where $|\Omega_f|$ is the pixel count, $\bar{r}_f$ the mean image row, and H the image height. The product rewards faces simultaneously large and low in the image.

4.2.4 Stage N3: Image-Plane Edge Directions

For each adjacent pair e, $\mathcal{B}_e$ is summarised by its sub-pixel midpoint

$$\mathbf{m}_e \;=\; \tfrac{1}{|\mathcal{B}_e|}\sum_{\mathbf{p}\in\mathcal{B}_e}\mathbf{p}, \tag{18}$$

and its dominant 2D direction $\mathbf{d}_e^{(2\mathrm{D})} = (d_u, d_v)^\top$, the leading right singular vector of the centred coordinate matrix. Edges with $|\mathcal{B}_e| < 2$ are excluded from (P_4). Under orthographic projection an isolated image edge leaves the depth slope of the 3D edge unconstrained, and any deterministic lift introduces structural bias; the constraint is therefore used as a coupling cost in Section 4.2.6.

4.2.5 Stage N4: Visible-Hemisphere Parameterisation

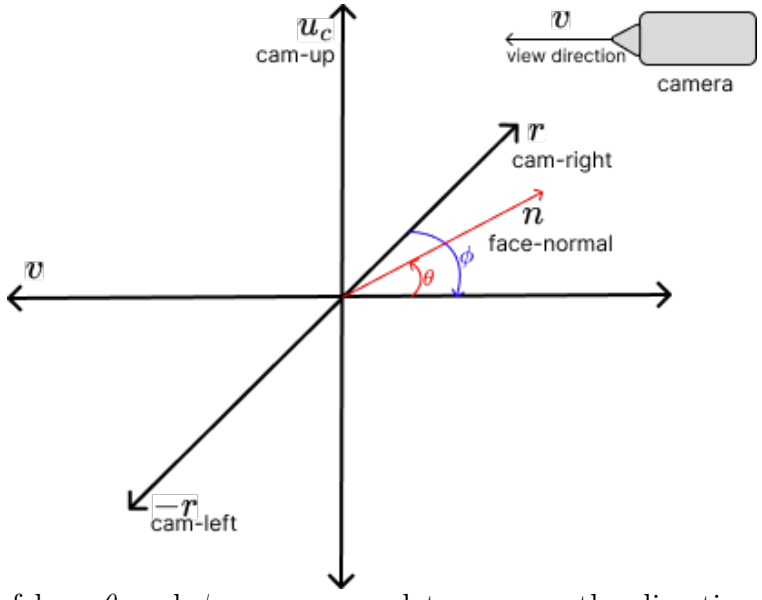


Figure 11: Example of how θ and ϕ are measured to express the direction of the face-normal in context of the camera view angle

The anchor normal is fixed to $\mathbf{n}_\mathrm{g} = (0, 0, 1)^T$. Every other face is parametrised on the open visible hemisphere via a polar angle $\theta_f \in [0, \theta_{\max}]$ and azimuth $\phi_f \in [0, 2\pi)$,

$$\mathbf{n}_f(\theta_f, \phi_f) \;=\; \sin\theta_f\,\big(\cos\phi_f\,\mathbf{r} + \sin\phi_f\,\mathbf{u}_\mathrm{c}\big) \;-\; \cos\theta_f\,\mathbf{v}, \tag{19}$$

with $\theta_{\max} = \arccos\varepsilon_\mathrm{v}$. At $\theta_f = 0$ the normal is exactly $-\mathbf{v}$ (face full-on to the camera); as θ_f grows the normal tilts toward the silhouette while remaining visible. The map (19) is a smooth bijection between $[0, \theta_{\max}] \times [0, 2\pi)$ and the open visible hemisphere; unit norm holds by construction, and (P_1) becomes the box constraint $\theta_f \le \theta_{\max}$. The inverse is

$$\theta_f \;=\; \arccos\big(-\mathbf{n}_f \cdot \mathbf{v}\big), \qquad \phi_f \;=\; \operatorname{atan2}\big(\mathbf{n}_f \cdot \mathbf{u}_\mathrm{c},\, \mathbf{n}_f \cdot \mathbf{r}\big); \tag{20}$$

if $\mathbf{n}_f$ falls outside the visible hemisphere it is reflected across the plane normal to $\mathbf{v}$ first. Every non-anchor face thus has exactly two scalar DOF, regardless of edge count; edge information enters via the soft term (P_4) rather than reducing the parametrisation.

4.2.6 Stage N5: Cost Function

The estimator minimises the global cost

$$\mathcal{C}\big(\{\mathbf{n}_f\}, \mathbf{l}, \rho\big) \;=\; \mathcal{C}_\mathrm{int} \;+\; \lambda_\mathrm{c}\,\mathcal{C}_\mathrm{conv} \;+\; \lambda_\mathrm{v}\,\mathcal{C}_\mathrm{vis} \;+\; \lambda_\mathrm{a}\,\mathcal{C}_\mathrm{cop} \;+\; \lambda_\mathrm{e}\,\mathcal{C}_\mathrm{edge}, \tag{21}$$

with default weights $(\lambda_\mathrm{c}, \lambda_\mathrm{v}, \lambda_\mathrm{a}, \lambda_\mathrm{e}) = (0.5, 1.0, 0.5, 0.2)$. The five terms encode the photometric model and the four priors of Section 4.2.1.

Intensity residual. The squared error between observed and Lambertian-predicted intensity:

$$\mathcal{C}_{\text{int}} \;=\; \sum_{f=1}^{N_f} \big(I_f - \rho\, \max(\mathbf{n}_f \cdot \mathbf{l}, 0)\big)^2. \tag{22}$$

Plane-separation prior (P_2)**.** Letting (c_f^x, c_f^y) denote the image centroid of face f, the camera-frame-lifted offset between f and g is

$$\boldsymbol{\Delta}_{fg} \;=\; (c_g^x - c_f^x)\,\mathbf{r} \;-\; (c_g^y - c_f^y)\,\mathbf{u}_{\text{c}}. \tag{23}$$

(P_2) demands $\mathbf{n}_f \cdot \boldsymbol{\Delta}_{fg} \le 0$ for every $g \neq f$ (anchor excluded as f). Violations are penalised quadratically and normalised by the squared image diagonal $D^2 = (c^x_{\max} - c^x_{\min})^2 + (c^y_{\max} - c^y_{\min})^2$:

$$\mathcal{C}_{\text{conv}} \;=\; \tfrac{1}{D^2} \sum_{f \neq g_0} \sum_{g \neq f} \big[\max\big(\mathbf{n}_f \cdot \boldsymbol{\Delta}_{fg},\, 0\big)\big]^2, \tag{24}$$

where g_0 is the anchor index. The max ensures only inequality violations are penalised, not satisfied configurations. Because $\boldsymbol{\Delta}_{fg}$ has no $\mathbf{v}$ component, this term cannot distinguish front from back along the optical axis; the visibility prior covers that direction independently.

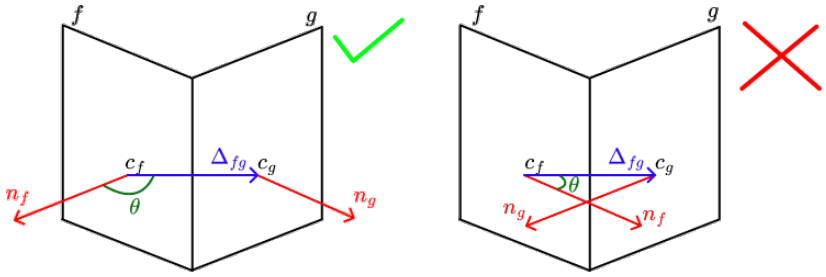


Figure 12: Visual representation of the plane separation prior

Visibility prior (P_1)**.** Although (19) already restricts $\theta_f \le \theta_{\max}$, a smooth penalty is also retained:

$$\mathcal{C}_{\text{vis}} \;=\; \sum_{f \neq g_0} \big[\max\big(\mathbf{n}_f \cdot \mathbf{v} + \varepsilon_{\text{v}},\, 0\big)\big]^2. \tag{25}$$

Anti-coplanar prior (P_3)**.** For each adjacent pair $e = (f_i, f_j) \in \mathcal{P}$,

$$\mathcal{C}_{\text{cop}} \;=\; \sum_{e \in \mathcal{P}} \big[\max\big(|\mathbf{n}_{f_i} \cdot \mathbf{n}_{f_j}| - \cos\theta_{\min},\, 0\big)\big]^2. \tag{26}$$

The absolute value penalises both $\theta = 0$ and $\theta = \pi$ symmetrically; no specific dihedral is preferred inside $(\theta_{\min}, \pi - \theta_{\min})$.

Edge-coherence prior (P_4)**.** For each adjacent pair with valid boundary cloud, the projection of $\mathbf{n}_{f_i} \times \mathbf{n}_{f_j}$ onto the image plane is

$$\boldsymbol{\pi}_e \;=\; \big((\mathbf{n}_{f_i} \times \mathbf{n}_{f_j}) \cdot \mathbf{r},\; -(\mathbf{n}_{f_i} \times \mathbf{n}_{f_j}) \cdot \mathbf{u}_{\text{c}}\big)^{\top}, \tag{27}$$

which after normalisation must be parallel to $\mathbf{d}_e^{(2\text{D})}$. Misalignment is captured by the 2D cross-product magnitude:

$$\mathcal{C}_{\text{edge}} \;=\; \sum_{e \in \mathcal{P}_{\text{val}}} \big(\hat{\pi}_{e,1}\, d_v - \hat{\pi}_{e,2}\, d_u\big)^2, \qquad \hat{\boldsymbol{\pi}}_e = \boldsymbol{\pi}_e / \|\boldsymbol{\pi}_e\|, \tag{28}$$

where $\mathcal{P}_{\text{val}} \subset \mathcal{P}$ collects pairs with $|\mathcal{B}_e| \ge 2$; degenerate cases ($\|\boldsymbol{\pi}_e\|$ or $\|\mathbf{n}_{f_i} \times \mathbf{n}_{f_j}\|$ below 10^{-8}) contribute zero.

4.2.7 Stage N6: Initialisation by Light Grid Search

Because $\mathcal{C}$ is non-convex in $\mathbf{l}$, a coarse spherical grid search produces an initial guess. The upper hemisphere is parametrised by elevation $\varphi \in [5°, 85°]$ and azimuth $\psi \in [0°, 360°)$ on an $N_g \times N_g$ grid (default $N_g = 24$),

$$\mathbf{l}(\varphi, \psi) \;=\; \big(\cos\varphi\cos\psi,\; \cos\varphi\sin\psi,\; \sin\varphi\big)^{\top}. \tag{29}$$

Each candidate $\mathbf{l}$ is evaluated by: (i) computing ρ from the anchor intensity (under `KnownAlbedo` the candidate is rejected unless $|\mathbf{n}_{\mathrm{g}} \cdot \mathbf{l} - I_{\mathrm{g}}| \le 0.25$; otherwise rejected when $\mathbf{n}_{\mathrm{g}} \cdot \mathbf{l} < 0.05$ to avoid near-singular division, else $\rho = I_{\mathrm{g}}/(\mathbf{n}_{\mathrm{g}} \cdot \mathbf{l})$ clipped to $[0.01, 10]$); (ii) sweeping faces in BFS order from the anchor, picking (θ_f, ϕ_f) on an $N_{gf} \times N_{gf}$ grid (default 24) that minimises

$$\widetilde{\mathcal{C}}_f(\mathbf{n}_f) \;=\; \big(I_f - \rho\,\max(\mathbf{n}_f \cdot \mathbf{l}, 0)\big)^2 \;+\; \lambda_{\mathrm{c}}\widetilde{\mathcal{C}}_{\mathrm{conv},f} \;+\; \lambda_{\mathrm{v}}\widetilde{\mathcal{C}}_{\mathrm{vis},f} \;+\; \lambda_{\mathrm{a}}\widetilde{\mathcal{C}}_{\mathrm{cop},f} \;+\; \lambda_{\mathrm{e}}\widetilde{\mathcal{C}}_{\mathrm{edge},f}, \tag{30}$$

the restriction of each term in $\mathcal{C}$ to contributions involving $\mathbf{n}_f$ (f's row of the plane-separation sum, f's visibility margin, and edges incident to f); (iii) evaluating full $\mathcal{C}$ on the resulting triple; (iv) keeping the lowest-$\mathcal{C}$ triple as initialisation. Equation (30) is also the per-face objective used during refinement (Section 4.2.8).

The complete top-level procedure is summarised in Algorithm **??**.

4.2.8 Stage N7: Joint Refinement by Alternating Coordinate Descent

The initialisation is refined by alternating block-coordinate descent that iterates between (i) a normal update with $(\rho, \mathbf{l})$ fixed, and (ii) a joint $(\rho, \mathbf{l})$ update with all $\mathbf{n}_f$ fixed - a Gauss-Seidel scheme [28].

Normal update (per-face golden-section search). Golden-section search [29] is a derivative-free 1D method that brackets the minimum of a unimodal function and shrinks the bracket by the inverse golden ratio per step.

Each non-anchor face f is refined by coordinate descent on (θ_f, ϕ_f). Five sweeps of (i) full-circle ϕ minimisation - first by 36-sample sampling, then golden-section refinement on the bracket spanning the best sample's neighbours - followed by (ii) bounded θ minimisation on $[0, \theta_{\max}]$ via golden-section search are performed. Each evaluation calls (30), costing $O(N_f)$ for the plane-separation row and $O(\deg f)$ for incident edges. Faces are visited sequentially within each outer iteration, so (P_3, P_4) are evaluated against the current snapshot of the other faces and each per-face problem is exactly two-dimensional.

The full algorithm can be viewed in algorithm (**??**)

Lighting update. With all normals fixed, the linear least-squares problem

$$\min_{\mathbf{s}\in\mathbb{R}^3} \big\|\mathbf{I} - N\mathbf{s}\big\|_2^2, \qquad N = \big[\mathbf{n}_1 \mid \mathbf{n}_2 \mid \ldots \mid \mathbf{n}_{N_f}\big]^{\top} \in \mathbb{R}^{N_f \times 3}, \tag{31}$$

is solved for $\mathbf{s} = \rho\mathbf{l}$ via the Moore-Penrose pseudoinverse $\mathbf{s}^* = N^{+}\mathbf{I}$. Under `KnownAlbedo`, only $\mathbf{l} = \mathbf{s}^*/\|\mathbf{s}^*\|$ is updated; otherwise $\rho = \|\mathbf{s}^*\|$ is also updated and clipped to $[0.01, 10]$. The $l_z > 0$ constraint is enforced: if $l_z \le 0$ the component is replaced by $|l_z| + 0.05$ and the result renormalised. The step is well-posed when the rows of N span $\mathbb{R}^3$; for rank-deficient configurations (e.g. several near-parallel walls plus the ground) the pseudoinverse remains stable but $\mathbf{l}$ along the missing direction is unconstrained.

Convergence. $\mathcal{C}$ is recomputed at the end of each outer iteration; the procedure terminates when the relative change falls below $\varepsilon_{\mathrm{conv}} = 10^{-6}$ or after `NumAltIter` (default 30) iterations. Convergence to the global minimum is not guaranteed, but the grid initialisation places the iterate in the basin of the solution and the smooth priors keep it there. Per-face block updates are monotone (golden-section search cannot increase the marginal cost) and the overall cost decreases until the lighting update introduces a coupled change.

Algorithm 15: Convex-polyhedron normal estimation

Input: Detection struct $\mathcal{D}$ (label map L, face intensities $\{I_f\}$); view direction $\mathbf{v}$; ground normal $\mathbf{n}_{\mathrm{g}}$

Output: Per-face unit normals $\{\mathbf{n}_f\}$, light direction $\mathbf{l}$, albedo ρ

Build adjacency graph A and boundary-pixel lists $\{\mathcal{B}_e\}$ from the label map (Stage N1)

Identify anchor face g_0 via score s_f of (17) (Stage N2); set $\mathbf{n}_{g_0} \leftarrow \mathbf{n}_{\mathrm{g}}$

Compute edge midpoints $\mathbf{m}_e$ and 2D directions $\mathbf{d}_e^{(2\mathrm{D})}$ via PCA over $\mathcal{B}_e$ (Stage N3)

Build cost context: face centroids (c_f^x, c_f^y), neighbour lists, incident-edge tables

`/* Stage N6: light grid search initialisation */`

$\mathcal{C}_{\mathrm{best}} \leftarrow +\infty$

for *each* (φ, ψ) *on the* $N_g \times N_g$ *light grid* (29) **do**

  Compute ρ_c from the anchor intensity (or $\rho_c = 1$ in `KnownAlbedo` mode); reject if outside the acceptance window

  for *each face* f *in BFS order from* g_0 *(excluding* g_0*)* **do**

    $(\theta_f, \phi_f) \leftarrow \arg\min_{(\theta,\phi)\in\mathrm{grid}} \widetilde{\mathcal{C}}_f\big(\mathbf{n}_f(\theta,\phi)\big)$

    $\mathbf{n}_f \leftarrow \mathbf{n}_f(\theta_f, \phi_f)$ via (19)

  if $\mathcal{C}\big(\{\mathbf{n}_f\}, \mathbf{l}, \rho_c\big) < \mathcal{C}_{\mathrm{best}}$ **then**

    Store $(\{\mathbf{n}_f\}, \mathbf{l}, \rho_c)$ as the current best

`/* Stage N7: alternating refinement */`

for $k \leftarrow 1$ **to** *`NumAltIter`* **do**

  for *each non-anchor face* f **do**

    `/* RefineFace: per-face golden-section coordinate descent */`

    $\theta \leftarrow \mathrm{clip}(\theta_f,\, 0,\, \theta_{\max})$; $\phi \leftarrow \phi_f \bmod 2\pi$

    for sweep $\leftarrow 1$ **to** 5 **do**

      `/* φ-pass: bracket by 36-sample sweep, then golden-section refine */`

      for $i \leftarrow 1$ **to** 36 **do**

        $\phi_i \leftarrow 2\pi(i-1)/36$

        $\mathbf{n}_i \leftarrow \mathbf{n}_f(\theta, \phi_i)$ via (19)

        $C_i \leftarrow \widetilde{\mathcal{C}}_f(\mathbf{n}_i,\, f,\, \{\mathbf{n}_g\},\, \mathbf{l},\, \rho)$

      $i^\star \leftarrow \arg\min_i C_i$

      $a \leftarrow \phi_{i^\star-1}$, $b \leftarrow \phi_{i^\star+1}$ (with wrap-around)

      $\phi \leftarrow \textsc{GoldenSection}\big(\phi \mapsto \widetilde{\mathcal{C}}_f(\mathbf{n}_f(\theta,\phi),\, \ldots),\, a,\, b,\, 10^{-4}\big)$

      `/* θ-pass on [0, θ_max] */`

      $\theta \leftarrow \textsc{GoldenSection}\big(\theta \mapsto \widetilde{\mathcal{C}}_f(\mathbf{n}_f(\theta,\phi),\, \ldots),\, 0,\, \theta_{\max},\, 10^{-4}\big)$

    $(\theta_f, \phi_f) \leftarrow (\theta, \phi)$; $\mathbf{n}_f \leftarrow \mathbf{n}_f(\theta_f, \phi_f)$

  Solve $\mathbf{s} \leftarrow N^{+}\mathbf{I}$ for the scaled lighting vector via (31)

  Update $(\rho, \mathbf{l})$ from $\mathbf{s}$, enforcing $l_z > 0$ by reflecting the third component as $l_z \leftarrow |l_z| + 0.05$ when non-positive, and clipping $\rho \in [0.01, 10]$

  if $|\mathcal{C}_k - \mathcal{C}_{k-1}|/\mathcal{C}_{k-1} < \varepsilon_{\mathrm{conv}}$ **then**

    break

Compute dihedral angles $\theta_e = \arccos(\mathbf{n}_{f_i} \cdot \mathbf{n}_{f_j})$ for diagnostics

return $\{\mathbf{n}_f\}$, $\mathbf{l}$, ρ

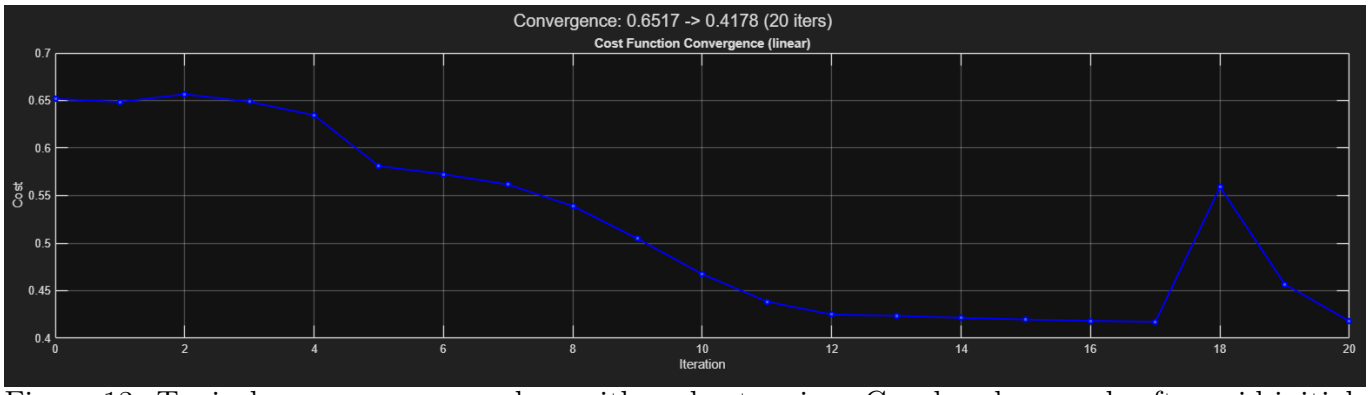


Figure 13: Typical convergence on values with early stopping. Graph only records after grid initialisation

4.2.9 Stage N8: Output and Diagnostics

After convergence, normals are renormalised defensively and dihedral angles $\theta_e = \arccos(\mathbf{n}_{f_i} \cdot \mathbf{n}_{f_j})$ are reported in degrees. The output struct exposes $\{\mathbf{n}_f\}$, $\mathbf{l}$, ρ, the anchor index, the cost trajectory, the adjacency matrix, the adjacent-pair list, the boundary-pixel lists, and per-pair dihedrals. The camera-frame triple is exposed so downstream functions can verify frame compatibility. When ground-truth normals and intensities are supplied (synthetic mode), a detected-to-ground-truth face mapping is computed by iterative greedy nearest-intensity matching - at each round the unmatched pair with the smallest intensity gap is committed - and used to compute per-face angular errors against $\{\mathbf{n}_f^{\mathrm{gt}}\}$.

```
============================================================
  NORMAL ESTIMATION ACCURACY SUMMARY
============================================================
Shape         GT   Det   Match  MeanErr   MedErr   MaxErr   LightE   Albedo     RMSE
------------------------------------------------------------------------------------
house          6     4     4/6    13.4°    15.6°    22.6°    14.9°    1.000   0.0251
shed           4     4     4/4    13.4°    16.2°    21.3°    16.6°    1.000   0.0488
tower          5     4     4/5    16.3°    17.6°    30.1°    35.5°    1.000   0.0913
pyramid        4     4     4/4    10.0°     9.3°    21.5°     4.2°    1.000   0.1653
wedge          3     3     3/3    17.9°    23.3°    30.3°    16.7°    1.000   0.1977
------------------------------------------------------------------------------------
AVERAGE                          14.2°    16.4°    25.2°    17.6°    1.000   0.1056
```

Figure 14: Normal estimation results on the synthetic polyhedron data

4.2.10 Minimal Improvements

Levenberg-Marquardt joint refinement. A natural improvement on the alternating block-coordinate descent of Section 4.2.8 is to replace it with a single second-order solver acting on the full parameter vector $(\{\theta_f, \phi_f\}_{f \neq g_0}, \varphi, \psi, \log \rho)$. Such a variant was implemented using `lsqnonlin` with the `levenberg-marquardt` algorithm [30] from MATLAB's Optimization Toolbox: the scalar cost (21) is rewritten as a residual vector whose squared norm reproduces $\mathcal{C}$ exactly; the θ-bounds and the $l_z > 0$ constraint are absorbed into a tanh reparametrisation, and $\rho > 0$ via $\rho = \exp(\log \rho)$, so LM operates on an unconstrained vector. The kinks in $\max(\mathbf{n} \cdot \mathbf{l}, 0)$, $\max(\cdot, 0)^2$, and $|\cdot|$ are replaced by their C^∞ smoothings $\frac{1}{2}(x + \sqrt{x^2 + \varepsilon^2})$, $\frac{1}{4}(x + \sqrt{x^2 + \varepsilon^2})^2$, and $\sqrt{x^2 + \varepsilon^2}$ respectively (with $\varepsilon^2 = 10^{-6}$, which perturbs each term by less than 5×10^{-4}). The grid initialisation of Section 4.2.7 is retained, since LM remains a local solver on a non-convex objective.

Empirically the mean per-face angular error did not improve over the alternating refinement on the development set, which we assume is the grid initialisation already placing the iterate inside the

data-consistent basin. The residual error is dominated by modelling assumptions rather than by optimiser quality so we opt to stick with our original optimisation solution.

Hyperparameter optimisation by random search and coordinate descent. The five cost weights together with ε_{v}, $\theta_{\min}$, and (N_g, N_{gf}) form an eight-dimensional parameter vector originally fixed by inspection. An outer optimiser was implemented to minimise $J = \overline{\theta_{\mathrm{ang}}} + 0.1\,\overline{\theta_\ell}$ - the mean per-face angular error plus a small light-direction penalty - averaged over a development set of four cached synthetic polyhedra (`house`, `shed`, `tower`, `pyramid`). The search has two phases: (i) random search of 40 uniform draws inside $\lambda_\bullet \in [0, 5]$, $\varepsilon_{\mathrm{v}} \in [0.001, 0.30]$, $\theta_{\min} \in [1°, 45°]$, $(N_g, N_{gf}) \in \{12, 18, 24, 36\}^2$, to localise the basin; (ii) coordinate descent of up to three sweeps over the eight axes, each replacing one coordinate with the best of a five-point line search, terminating when a sweep yields no improvement. Crashing trials are caught and assigned a 90° penalty. The optimum is $(\lambda_{\mathrm{c}}, \lambda_{\mathrm{v}}, \lambda_{\mathrm{a}}, \lambda_{\mathrm{e}}) = (1.25, 3.5016, 3.75, 0.0868)$, $\varepsilon_{\mathrm{v}} = 0.3$, $\theta_{\min} = 23°$. The calibrated weights are roughly an order of magnitude larger than the inspection values, indicating the priors needed more leverage relative to the intensity residual than initially apparent. These weights also reduce mean error on held-out shapes (e.g. a wedge).

4.2.11 Normal Estimation Results

Due to our structural assumptions, it makes it difficult to find specific training data and images that follow all of the assumptions our imaging model has put in place. To combat this, we opt to generate synthetic data that follows our assumptions. Empirically, we achieve a mean error of 13° on the normal estimations, an accuracy comparable to the per-face dihedral spread of the ground truth, indicating that the convex-polyhedron priors and Lambertian lighting model together carry enough information to recover the normal field to well within the tolerance required for the downstream 3D reconstruction stage to produce a coherent polyhedron. The residual error is consistent with the modelling assumptions from 4.2.12.

4.2.12 Structural Assumptions and Problem Limitations

Structural assumptions. The estimator inherits several assumptions from the imaging and reflectance setup. These are conditions of the problem statement and bound the regimes in which the recovered normals are meaningful:

1. **Level-ground assumption.** Fixing $\mathbf{n}_{\mathrm{g}} = (0, 0, 1)^\top$ assumes a perfectly horizontal world ground; sloped terrain introduces a systematic rotation of the entire normal field.
2. **Single-albedo Lambertian reflectance.** A scalar ρ conflates colour and reflectance variation with shape: a dark roof on a light wall is reconstructed as more steeply pitched, since the solver explains lower intensity by changing $\mathbf{n}\cdot\mathbf{l}$ rather than ρ. Equation (15) also has no ambient term and ignores shadowing; shadowed faces violate the model and the optimiser then rotates the normal sharply away from $\mathbf{l}$.
3. **Distant point light source.** A single directional $\mathbf{l}$ assumes one source at infinity. Multiple sources, area sources, or near-field lighting all violate the model.

Fundamental limitations of the ill-posed input. Even with a perfect algorithm, a single greyscale image under orthographic projection cannot resolve the following ambiguities - they are properties of the input data, not of the estimator:

1. **Two DOF per face are irreducible.** Reducing this further would require a true 3D edge-perpendicularity constraint, but a single image edge under orthographic projection determines only a 2D direction; the depth component is unobservable from a single view.

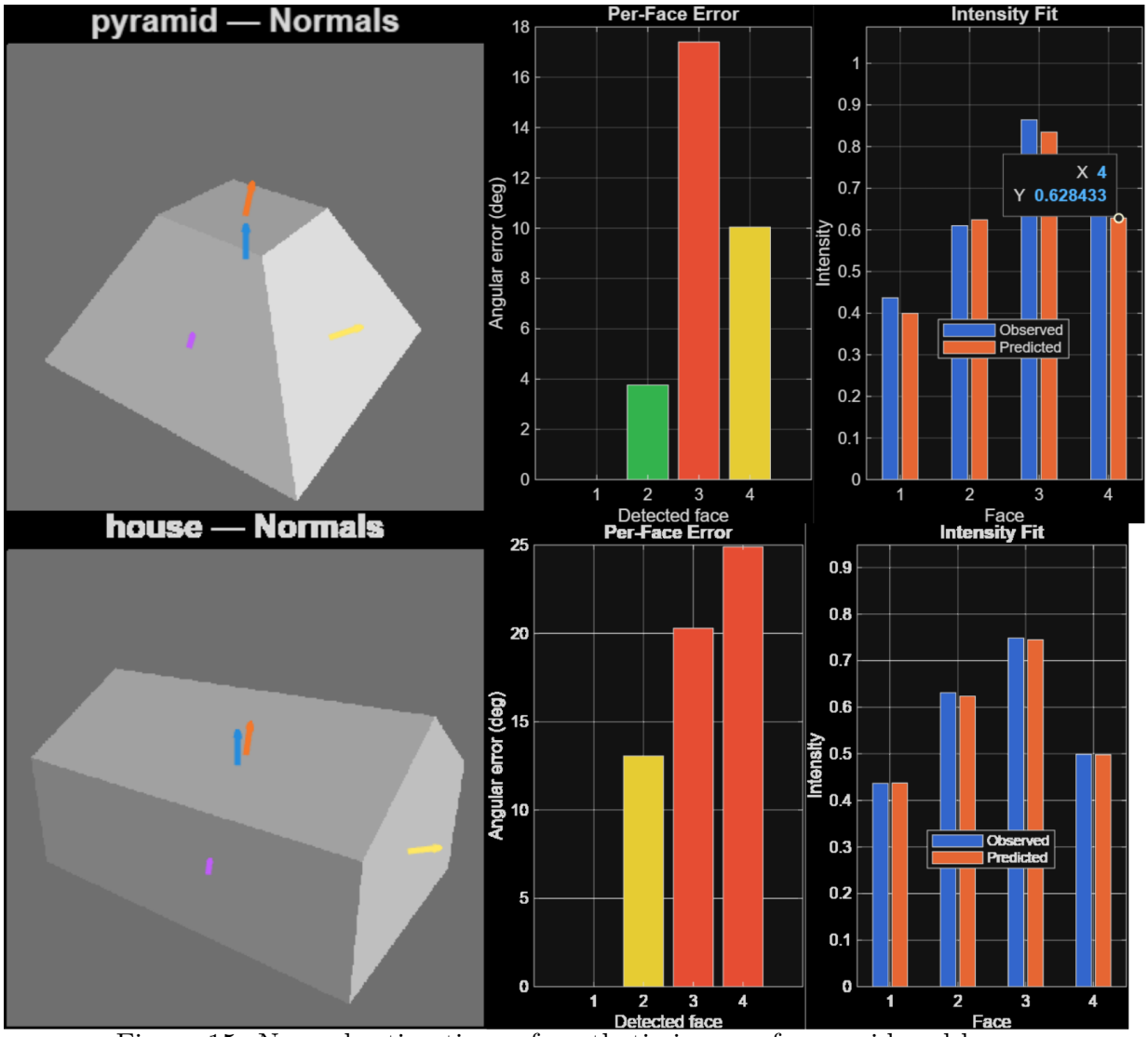


Figure 15: Normal estimations of synthetic image of pyramid and house

2. **Plane-separation collapses to the image plane.** Equation (23) has no $\mathbf{v}$ component, so two faces whose image centroids coincide but whose true centroids differ along the optical axis contribute zero to $\mathcal{C}_{\text{conv}}$. Under the convex-polyhedron assumption this is geometrically impossible from a generic viewpoint, so the limitation is rarely hit in practice.
3. **Bas-relief residual.** The Lambertian model admits the gauge $\mathbf{n} \mapsto G\mathbf{n}$, $\mathbf{l} \mapsto G^{-\top}\mathbf{l}$ leaving images unchanged [27]. The ground anchor removes the depth-scale component, and the convex priors generically remove the rest. On highly symmetric shapes, gauges coinciding with the shape's own symmetries leave every prior unchanged, producing a finite residual ambiguity.

4.2.13 Summary of Parameters

4.3 Polyhedron Reconstruction from Estimated Normals

Given the per-face unit normals $\{\mathbf{n}_f\}_{f=1}^{N_f}$ recovered by the shape-from-shading estimator of Section 4.2, together with the labelled face partition and the detected vertex set $\{\mathbf{v}_i^{(2D)}\}_{i=1}^{N_v}$ from Section 4.1, the final stage of the pipeline is to recover the 3D positions of every vertex in a common world frame. The estimator returns oriented planes but no information about where each plane sits along its own normal direction; the reconstruction stage fixes these offsets and then back-projects the detected 2D vertices onto the resulting plane arrangement. The procedure trusts the estimated normals completely, so any error in $\{\mathbf{n}_f\}$ propagates directly into the recovered geometry. The

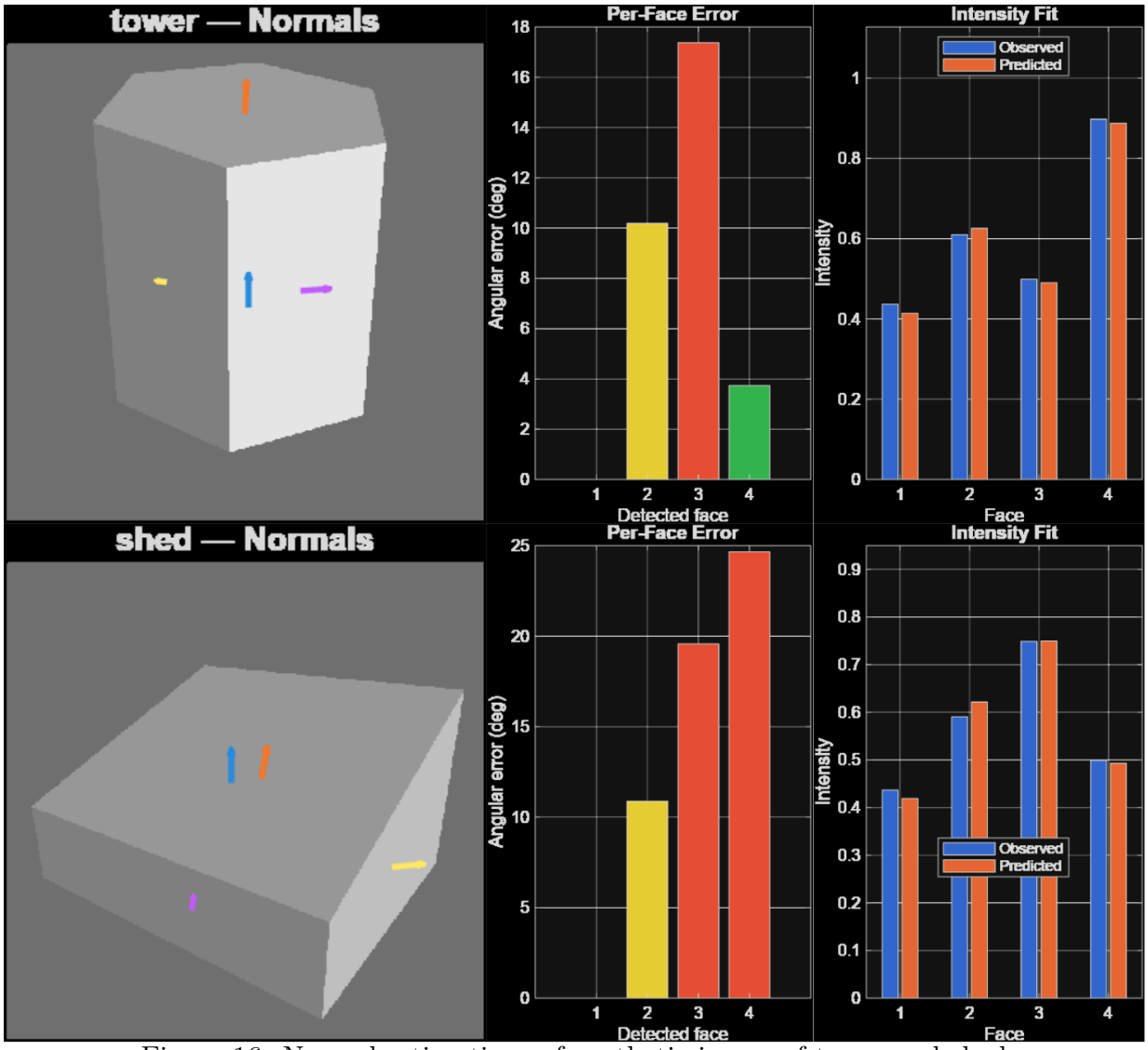


Figure 16: Normal estimations of synthetic image of tower and shed

Table 2: Complete set of tuneable parameters for the convex-polyhedron normal estimation algorithm, together with the default values used for real-image experiments.

Parameter	**Symbol**	**Default**	**Description**
Ground normal	$\mathbf{n}_{\mathrm{g}}$	$(0,0,1)^{\top}$	World up direction (anchor face normal)
Camera view direction	$\mathbf{v}$	$(0,1,0)^{\top}$	Camera-to-scene direction in world frame
Anchor face index	g_0	auto	Index of fixed-normal face (0 = score-based)
Maximum iterations	$K_{\max}$	30	Outer alternating-descent iterations
Light grid resolution	N_g	24	Samples per axis on light hemisphere grid
Face grid resolution	N_{gf}	24	Samples per axis on per-face (θ, ϕ) grid
Convergence tolerance	$\varepsilon_{\mathrm{conv}}$	10^{-6}	Relative cost change for early termination
Patience	K_{pat}	3	Iterations without improvement before revert
Known albedo flag	–	true	If true, fix $\rho = 1$; else fit $\rho \in [0.01, 10]$
Visibility margin	ε_{v}	0.05	Min $\lvert\mathbf{n}_f \cdot \mathbf{v}\rvert$ for visible faces
Min dihedral separation	$\theta_{\min}$	10°	Min angular separation from coplanar
Plane-separation weight	λ_{c}	0.5	Weight of convex-polyhedron prior (P_2)
Visibility weight	λ_{v}	1.0	Weight of camera-facing prior (P_1)
Anti-coplanar weight	λ_{a}	0.5	Weight of non-coplanar dihedral prior (P_3)
Edge-coherence weight	λ_{e}	0.2	Weight of shared-edge coherence prior (P_4)

implications of this are discussed in Section 4.3.5.

4.3.1 World Frame and Orthographic Lift

The world frame is fixed by the same conventions used by the normal estimator (Section 4.2.1): the ground normal is locked to $\mathbf{n}_{\mathrm{g}} = (0, 0, 1)^{\top}$, and the camera looks along the world $+Y$ axis under orthographic projection. Image-plane coordinates (u, v) thus lift to world coordinates (X, Z) directly, with the depth axis Y unobservable from a single image. Concretely, for a detected vertex $\mathbf{v}_i = (u_i, v_i)$,

$$X_i = u_i, \qquad Z_i = v_{\mathrm{ref}} - v_i, \qquad Y_i = ? \tag{32}$$

where v_{ref} is the largest image row among the vertices of the ground face g_0, chosen as the depth-zero point so that ground vertices land exactly on the plane $Z = 0$. The negative sign on v converts image-row-down convention to world-up. The reconstruction problem reduces to solving for the missing Y_i at every vertex, plus a single scalar offset d_k for every plane.

Each face plane is written in the implicit form

$$\mathbf{n}_k \cdot \mathbf{P} = d_k, \qquad \mathbf{P} = (X, Y, Z)^{\top} \in \mathbb{R}^3, \tag{33}$$

with the ground plane locked at $d_{g_0} = 0$ to fix the gauge of the depth-scale ambiguity inherent in orthographic reconstruction. The remaining unknowns are the $N_f - 1$ offsets $\{d_k\}_{k \neq g_0}$ and the depth coordinates $\{Y_i\}_{i=1}^{N_v}$, recovered in two sequential passes.

4.3.2 Pass 1: Plane Offsets via Breadth-First Propagation

Ground anchoring. A subset of detected vertices is identified as physically lying on the floor through structural assumption that walls meet the floor along a single polygon edge, and the two endpoints of that edge are the only wall vertices at $Z = 0$. For each non-ground face k, the polygon edge with the largest mean image row is selected as the candidate floor edge. The pair (v_1, v_2) is accepted as a floor edge iff no other non-ground face shares both endpoints; otherwise, the edge is a roof-line shared with a roof, this helps massively in reducing the number of occlusion edges being considered plane-connecting edges. The union of all accepted endpoint pairs forms the ground-vertex set $\mathcal{V}_{\mathrm{g}}$, and Z_i is overwritten as zero for every $i \in \mathcal{V}_{\mathrm{g}}$.

Seed wall and gauge fixing. We select one wall as the starting point and assign its offset directly. The 'seed' is the first non-ground face k whose vertex list intersects $\mathcal{V}_{\mathrm{g}}$. Picking any such anchor vertex $v_{\mathrm{seed}} \in \mathcal{F}_k \cap \mathcal{V}_{\mathrm{g}}$ and fixing its world coordinates as $\mathbf{P}_{\mathrm{seed}} = (u_{v_{\mathrm{seed}}}, 0, 0)^{\top}$ yields

$$d_{\mathrm{seed}} \;=\; \mathbf{n}_{\mathrm{seed}} \cdot \mathbf{P}_{\mathrm{seed}} \;=\; n_{\mathrm{seed},X}\, u_{v_{\mathrm{seed}}}. \tag{34}$$

Every other plane offset is determined relative to this anchor.

BFS propagation. The remaining offsets are propagated outward from the seed by BFS of the face adjacency graph, restricted to wall-wall edges. Suppose face a has known offset d_a and shares an adjacency edge $e = (a, b)$ with an unknown face b. The boundary-pixel cloud $\mathcal{B}_e$ from the normal-estimator output (Section 4.2.2) lies along the shared 3D edge; taking any point $\mathbf{p}_e \in \mathcal{B}_e$ and lifting it to world coordinates fixes a single point on plane a, from which d_b follows by evaluating plane b's equation at that point. The implementation uses the median to suppress outliers:

$$X_e = \mathrm{median}\{u : (u, v) \in \mathcal{B}_e\}, \qquad Z_e = v_{\mathrm{ref}} - \mathrm{median}\{v : (u, v) \in \mathcal{B}_e\}. \tag{35}$$

The depth coordinate at this point is recovered by solving plane a's equation (33) for Y:

$$Y_e \;=\; \tfrac{d_a - n_{a,X} X_e - n_{a,Z} Z_e}{n_{a,Y}}, \tag{36}$$

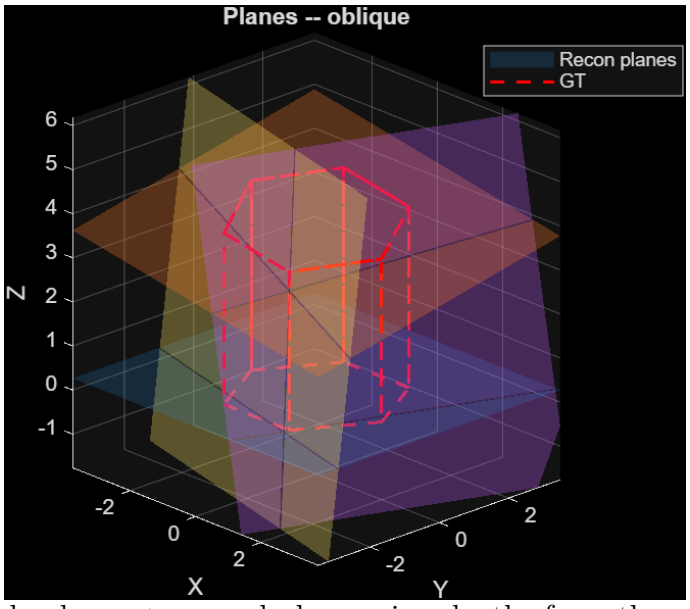


Figure 17: 3 visible planes + ground plane using depths from the estimated normals

which requires $|n_{a,Y}| \geq 10^{-3}$ - that is, plane a must not be edge-on to the camera, since an edge-on plane projects to a line and provides no depth information. The unknown offset is then

$$d_b \;=\; \mathbf{n}_b \cdot (X_e, Y_e, Z_e)^\top. \tag{37}$$

Face b is marked visited and its neighbours are enqueued. The process terminates when the queue empties; all ground faces will be visited, our structural assumption enforces that the scene contains only one polyhedron.

4.3.3 Pass 2: Vertex Depths from Incident Plane Equations

With every plane fully determined as $(\mathbf{n}_k, d_k)$, each vertex i is associated with a set $\mathcal{K}_i \subset \{1, \ldots, N_f\} \setminus \{g_0\}$ of incident wall planes, obtained from the face polygon adjacencies in from planar detection (4.1). For a vertex i incident to wall plane k, the plane equation (33) evaluated at (X_i, Y_i, Z_i) gives a single scalar candidate for the unknown depth:

$$Y_i^{(k)} \;=\; \tfrac{d_k - n_{k,X} X_i - n_{k,Z} Z_i}{n_{k,Y}}, \qquad k \in \mathcal{K}_i,\ |n_{k,Y}| \geq 10^{-3}. \tag{38}$$

For a geometrically consistent reconstruction with exact normals, all $|\mathcal{K}_i|$ candidates would coincide; in practice they differ slightly due to noise in the estimated normals and offsets, and the depth is reported as the arithmetic mean:

$$Y_i \;=\; \tfrac{1}{|\mathcal{K}_i^*|} \sum_{k \in \mathcal{K}_i^*} Y_i^{(k)}, \qquad \mathcal{K}_i^* = \{k \in \mathcal{K}_i : |n_{k,Y}| \geq 10^{-3}\}. \tag{39}$$

Vertices with $i \in \mathcal{V}_\mathrm{g}$ retain $Z_i = 0$ regardless of the value implied by (38) - the ground anchor is a hard constraint rather than as another plane equation.

Incidence augmentation. The face-polygon incidence is occasionally incomplete: the apex of a pyramid, or the four upper corners of a tower, may be reachable through fewer plane equations than required to determine Y_i uniquely. To repair these gaps, the endpoints of every adjacency edge's boundary cloud (computed by PCA on $\mathcal{B}_e$, 4.2.2) are matched to detected vertices by Euclidean nearest-neighbour search with tolerance $\tau_\mathrm{vert} = 18\,\mathrm{px}$. When an endpoint matches a vertex, the two faces sharing that adjacency are added to that vertex's incidence set $\mathcal{K}_i$ if not already present.

The complete two-pass procedure is summarised in Algorithm 16.

Algorithm 16: Polyhedron Reconstruction from Estimated Normals

Input: Detection $\mathcal{D}$ (vertices $\{\mathbf{v}_i^{(2D)}\}$, faces $\{\mathcal{F}_k\}$); normals $\{\mathbf{n}_k\}$; ground index g_0; adjacencies $\mathcal{P}$ with boundary clouds $\{\mathcal{B}_e\}$
Output: World vertices $\{\mathbf{P}_i\}$ and plane offsets $\{d_k\}$

Lock $\mathbf{n}_{g_0} \leftarrow (0,0,1)^\top$ and $d_{g_0} \leftarrow 0$
$v_{\text{ref}} \leftarrow \max\{v : (u,v) = \mathbf{v}_j^{(2D)},\ j \in \mathcal{F}_{g_0}\}$
For every detected vertex set $X_i \leftarrow u_i$, $Z_i \leftarrow v_{\text{ref}} - v_i$, $Y_i \leftarrow \text{NaN}$
`/* Ground anchoring (Section 4.3.2) */`
$\mathcal{V}_{\text{g}} \leftarrow \emptyset$
for *each non-ground face* k **do**
  Identify polygon edge $(v_1, v_2) \in \mathcal{F}_k$ with largest mean image row
  if *no other non-ground face contains both* v_1 *and* v_2 **then**
    $\mathcal{V}_{\text{g}} \leftarrow \mathcal{V}_{\text{g}} \cup \{v_1, v_2\}$
$Z_i \leftarrow 0$ for all $i \in \mathcal{V}_{\text{g}}$
`/* Pass 1: plane offsets */`
Find seed face s with $\mathcal{F}_s \cap \mathcal{V}_{\text{g}} \neq \emptyset$; pick $v_{\text{seed}} \in \mathcal{F}_s \cap \mathcal{V}_{\text{g}}$
$d_s \leftarrow n_{s,X} \cdot u_{v_{\text{seed}}}$ `// Equation` (34)
Mark s visited; enqueue s
while *queue not empty* **do**
  Dequeue a
  for *each wall neighbour* b *of* a *via edge* e*, with* b *unvisited* **do**
    if $|\mathcal{B}_e| < 2$ ***or*** $|n_{a,Y}| < 10^{-3}$ **then**
      continue
    Compute (X_e, Z_e) via Equation (35); Y_e via Equation (36)
    $d_b \leftarrow \mathbf{n}_b \cdot (X_e, Y_e, Z_e)^\top$ `// Equation` (37)
    Mark b visited; enqueue b
`/* Pass 2: vertex depths */`
Build incidence sets $\mathcal{K}_i$ from face polygons (excluding ground)
Augment $\mathcal{K}_i$ via PCA-endpoint matching with tolerance τ_{vert}
for *each vertex* i *with* $\mathcal{K}_i \neq \emptyset$ **do**
  $\mathcal{K}_i^* \leftarrow \{k \in \mathcal{K}_i : |n_{k,Y}| \geq 10^{-3}\}$
  if $\mathcal{K}_i^* \neq \emptyset$ **then**
    $Y_i \leftarrow \frac{1}{|\mathcal{K}_i^*|} \sum_{k \in \mathcal{K}_i^*} (d_k - n_{k,X} X_i - n_{k,Z} Z_i)/n_{k,Y}$
return $\{\mathbf{P}_i = (X_i, Y_i, Z_i)\}$, $\{d_k\}$

4.3.4 Reconstruction Results

When supplied with ground-truth normals and exact face polygons, Algorithm 16 recovers the polyhedron geometry exactly: every plane offset solves to its exact value, every vertex depth is consistent across all incident planes, and the recovered mesh is congruent to the input shape. This is expected, the algorithm is exact in its arithmetic and introduces no approximations. The reconstruction is best understood as a deterministic decoder: given the right normals, it produces the right geometry.

The normal estimation error is multiplicative in nature, since (36) divides by $n_{a,Y}$, faces oriented near edge-on amplify the noise in d_a when computing Y_e, which is then carried into d_b. On simple shapes such as a tower or a pyramid, where every visible face is well-oriented, the reconstruction is largely correct. On more complex shapes, where some faces are estimated with substantial angular

error, the propagated offsets drift and the resulting mesh exhibits walls that fail to meet at their shared edges, roof faces tilted away from their true pitch, or corners floating off the floor.

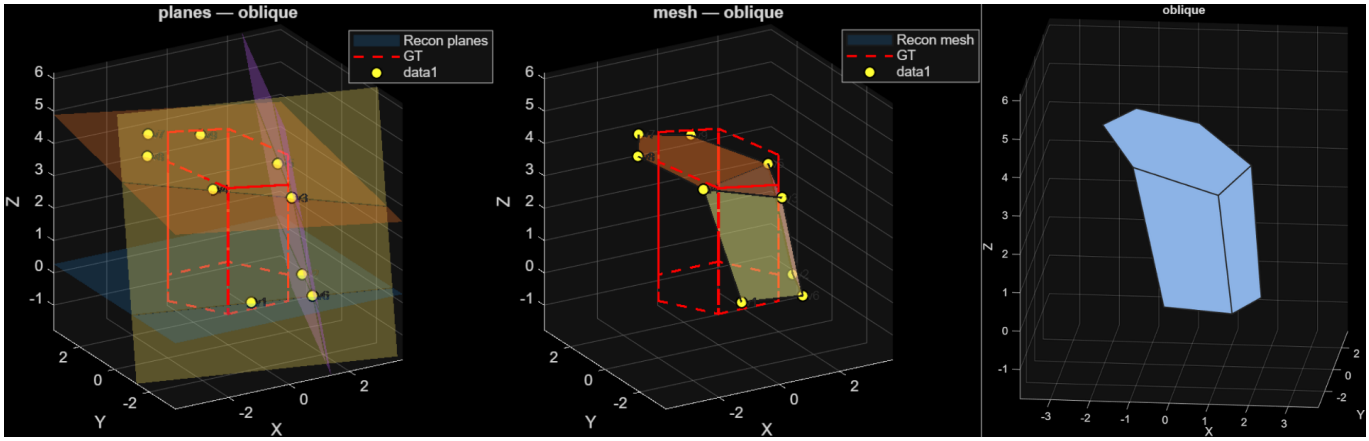


Figure 18: Reconstruction results of the tower image

4.3.5 Limitations and Sources of Error

The reconstruction stage inherits every limitation of the upstream stages and adds a few of its own. They fall into three categories.

Sensitivity to normal-estimation error. The reconstruction is fully determined by the estimated normals; it has no degrees of freedom of its own beyond the gauge fix at the seed. As a consequence, the algorithm cannot 'correct' a poorly estimated normal - it can only propagate the consequences of that error through the BFS tree. Two effects are common in practice:

1. **Edge-on amplification.** When the normal estimator returns a face whose recovered normal has small $|n_{f,Y}|$, equation (36) divides by a near-zero quantity and any noise in d_a blows up the recovered Y_e. The implementation guards against the catastrophic case ($|n_{f,Y}| < 10^{-3}$) by skipping the edge and waiting for a different propagation path.
2. **BFS path dependence.** Because the propagation is a tree rooted at the seed, the offset of a distant face is a sum of error contributions along the path connecting it to the root. Faces deeper in the BFS tree inherit accumulated error from every face on their path, and the choice of seed affects which faces accumulate the most error. This strategy doesn't look to improve this; a least-squares variant is a natural extension.

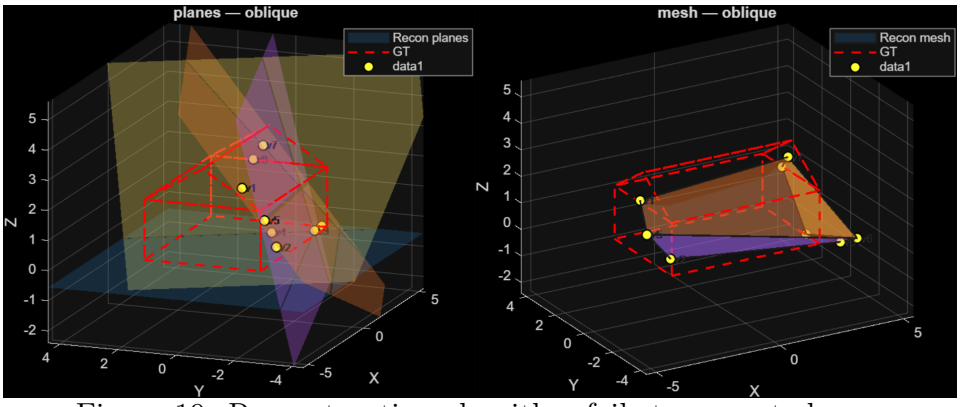


Figure 19: Reconstruction algorithm fails to recreate house

Per-vertex depth ambiguity. Even with perfect normals, a vertex incident to only one wall plane has a one-parameter family of consistent depths along the projection ray. Equation (38) returns a single value in that case only because the orthographic lift of X_i and Z_i together with the plane equation form a determined system. If the plane is itself slightly mis-estimated, the vertex moves with it. Vertices with multiple incident planes are over-determined and the mean of equation (39) provides some smoothing, but the smoothing is unweighted and treats points equally no matter their accuracy. A weighted average proportional to $|n_{k,Y}|$ would be a small but principled improvement.

Inherited orthographic and structural assumptions. The world frame relies on the level-ground assumption, the single-view depth ambiguity, and the convex-polyhedron structural priors discussed in Section 4.2.12. Photographs taken with a tilted camera or of non-convex objects violate these assumptions and produce distortions that no amount of tuning in the reconstruction stage can correct.

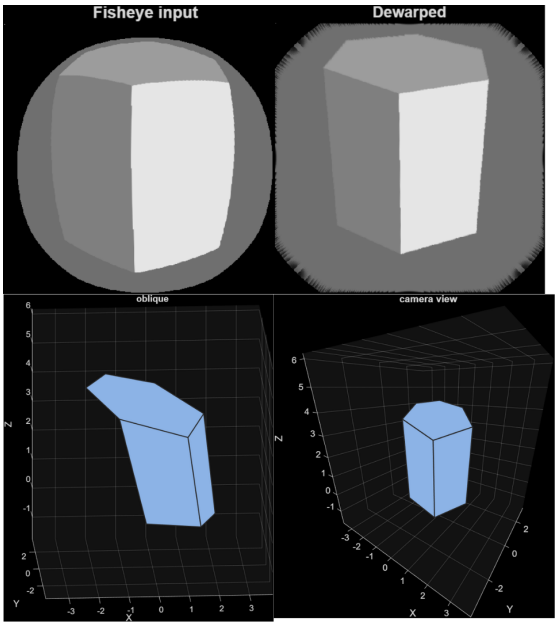
In summary, the reconstruction algorithm is a faithful but unforgiving decoder: it produces the exact polyhedron when given exact normals, and produces a recognisable but imperfect polyhedron when given the noisy normals of the current shape-from-shading estimator. The dominant source of reconstruction error in the current pipeline is the residual angular error in the upstream normal estimation. A natural direction for future work is to replace the BFS-tree offset propagation with a global least-squares fit over all adjacency edges simultaneously, which would distribute normal-estimation error more evenly across the mesh rather than letting it accumulate along a single path.

5 Complete Pipeline

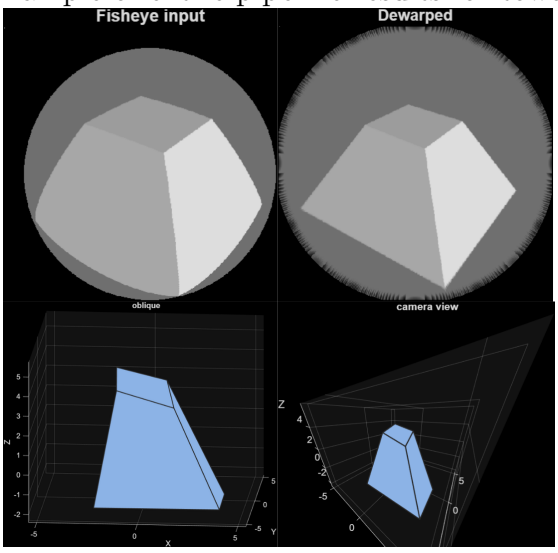
All that's left is to take the synthetic polyhedron renderer 4.2, distort it with a fisheye lens filter, and combine our test files into one pipeline, dewarp, plane detection, normal estimation, reconstruction.
Two main properties of the integrated pipeline are worth testing:

1. **End-to-end geometric fidelity.** Whether the recovered mesh is congruent to the input polyhedron after the full chain of fisheye distortion and dewarping.
2. **Robustness to dewarping imperfection.** Whether residual distortion left by the optimised fisheye parameters degrades downstream planar detection in a graceful or catastrophic manner.

Reconstruction propagation behaves as predicted: simple shapes such as `tower` and `pyramid`, where every visible face has well-conditioned $|n_{f,Y}|$, produce visually correct meshes. More complex shapes such as `house` show the expected drift. These results confirm that the dominant error source in the integrated pipeline is normal estimation, not dewarping, and motivate the global least-squares offset solver proposed in Section 7 as the highest-impact next step.



Example of entire pipeline results for tower



Example of entire pipeline results for pyramid

6 Reflection

This project built a complete pipeline from raw fisheye photographs to reconstructed 3D geometry, and the journey through dewarping, planar detection with normal estimation, and reconstruction was instructive both for what worked and what proved harder than anticipated.

Dewarping was a smooth process. Modelling the equidistant projection from first principles and optimising calibration by coordinate descent over ternary searches gave a measurable improvement (RMSE 42.8 to 30.9 on FIORD, with consistent two-fold cross-validation). Ternary search is unsophisticated, but combined with multiple sweeps it handled parameter coupling well at this scale.

Planar detection, while having many small intricate steps, also proved to go largely as planned. Knowing the clean plan - 1. Edge detection, 2. RANSAC, 3. plane detection - helped a lot, and after each stage, it was easy to visually see how I could improve and the next step to detect structured planes.

Normal estimation was a very tricky process to try and optimise. Reducing a famously ill-posed problem to $2(N_f - 1) + 3$ unknowns is tractable, and the achieved 13° mean angular error is consistent with what those assumptions allow. It took lots of small tweaks, playing with different priors and cost functions to get discernable results from the normal estimator.

Reconstruction was extremely challenging to retrieve a good looking result. What initially started as a least-squares solution, quickly became too complicated and no high quality results were being discovered through this. Instead, pivoting to trusting the normals wholly and reconstructing the depths exactly off of the previous information was the only way to move forward. This allowed more time to be spent into increasing the effectiveness of previous stages.

7 Future Work

Global least-squares offset solving. Knowing what was said before, with more time and resources, a global least-squares solution would in theory produce a much more effective result. Stacking adjacency-edge constraints into a sparse linear system distributes normal-estimation error evenly across the mesh rather than accumulating it along a tree path, and extends naturally to weighted least squares using boundary-cloud size.

Multi-view reconstruction. This was anticipated by the abstract and remains the most impactful extension. A second view reduces DOF for many variables and eliminates orthographic depth ambiguity, in turn removing the bas-relief residual, and enables direct triangulation.

Relaxing structural assumptions. Per-face albedo (estimated jointly via the block-structured lighting system) would handle cases like dark roof on light wall. Estimating ground normal from vanishing points would handle tilted cameras. Extending convexity to piecewise-convex objects would broaden the admissible scene class substantially.

Appendix: Code Structure and Usage

The codebase is implemented in MATLAB and organised into two main modules: `fisheye/` for fisheye calibration and dewarping, and `3d_reconstruction/` for polyhedron reconstruction. All scripts are run from the project root in MATLAB.

Top-level

`generate_synthetic_polyhedron_fisheye.m` Generates synthetic fisheye images of polyhedra for testing.

`test_fisheye_reconstruction.m` End-to-end pipeline: dewarp a fisheye image then reconstruct the polyhedron.

`fisheye/` - Calibration and Dewarping

`optimize_fisheye_parameters.m` Optimises intrinsic fisheye parameters on a single image.

`train_fisheye_dataset.m` Trains parameters across the dataset in `ground truth/`.

`cross_validate_fisheye.m` k-fold cross-validation of trained parameters.

`validate_fisheye_dewarping.m`, `validate_fisheye_batch.m` Evaluate and test dewarping quality on single / batch inputs.

`helpers/` Utilities: `dewarp_fisheye.m` (core dewarping), `interpolate_pixel.m`, `compute_rmse.m`, `compute_safe_theta_max.m`, `ternary_search.m`, `downsample.m`, `display_params.m`.

`3d_reconstruction/` - Polyhedron Reconstruction

`canny_edge_detection.m` Edge extraction from dewarped images.

`detect_planar_faces.m` Groups edges into planar polyhedral faces.

`generate_synthetic_polyhedron.m` Generates synthetic test polyhedra.

`test_*.m` Unit tests for each stage (planar detection, normal estimation, reconstruction).

Image Data

`ground truth/` contains the calibration dataset; `test images/` and `dev_test_images/` hold evaluation and development inputs.`ground truth/` and `test images/` contain 15GB of images and won't be included in the submission. They can be found at `https://parmisian.github.io/fiord360.github.io/`.

Running

Open MATLAB at the project root, add subfolders to the path (`addpath(genpath(pwd))`), then execute the desired top-level or `test_*` script.

8 Appendix: Planar Face Detection Stage Visualisations

This appendix collects the stage-by-stage visualisations of the planar face detection pipeline (Section 4.1) as applied to the example house image. Each figure illustrates the output of a specific stage described in the main text.



Figure 20: Stage 1 (Section 4.1.1): Canny edge detection on the example house image.

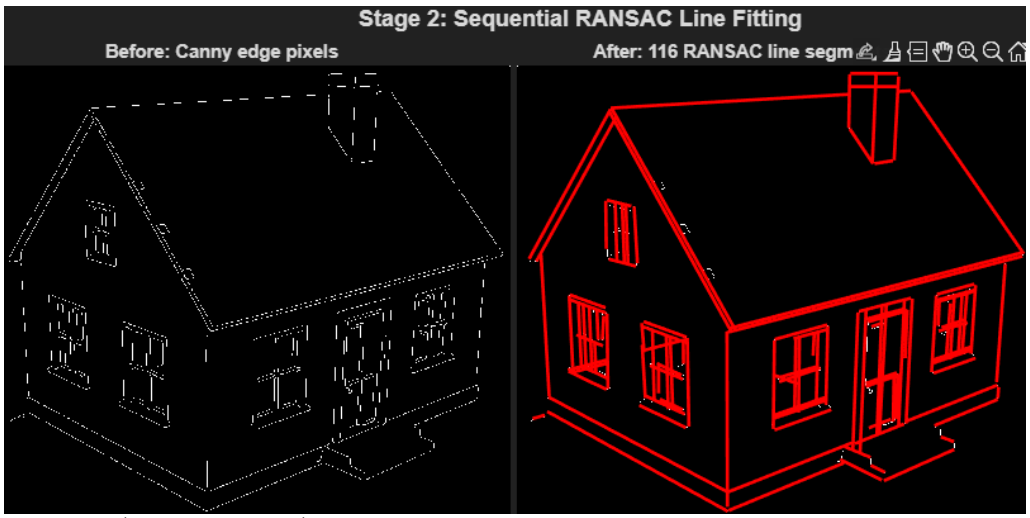


Figure 21: Stage 2 (Section 4.1.2): Sequential RANSAC line fitting on the example house image.

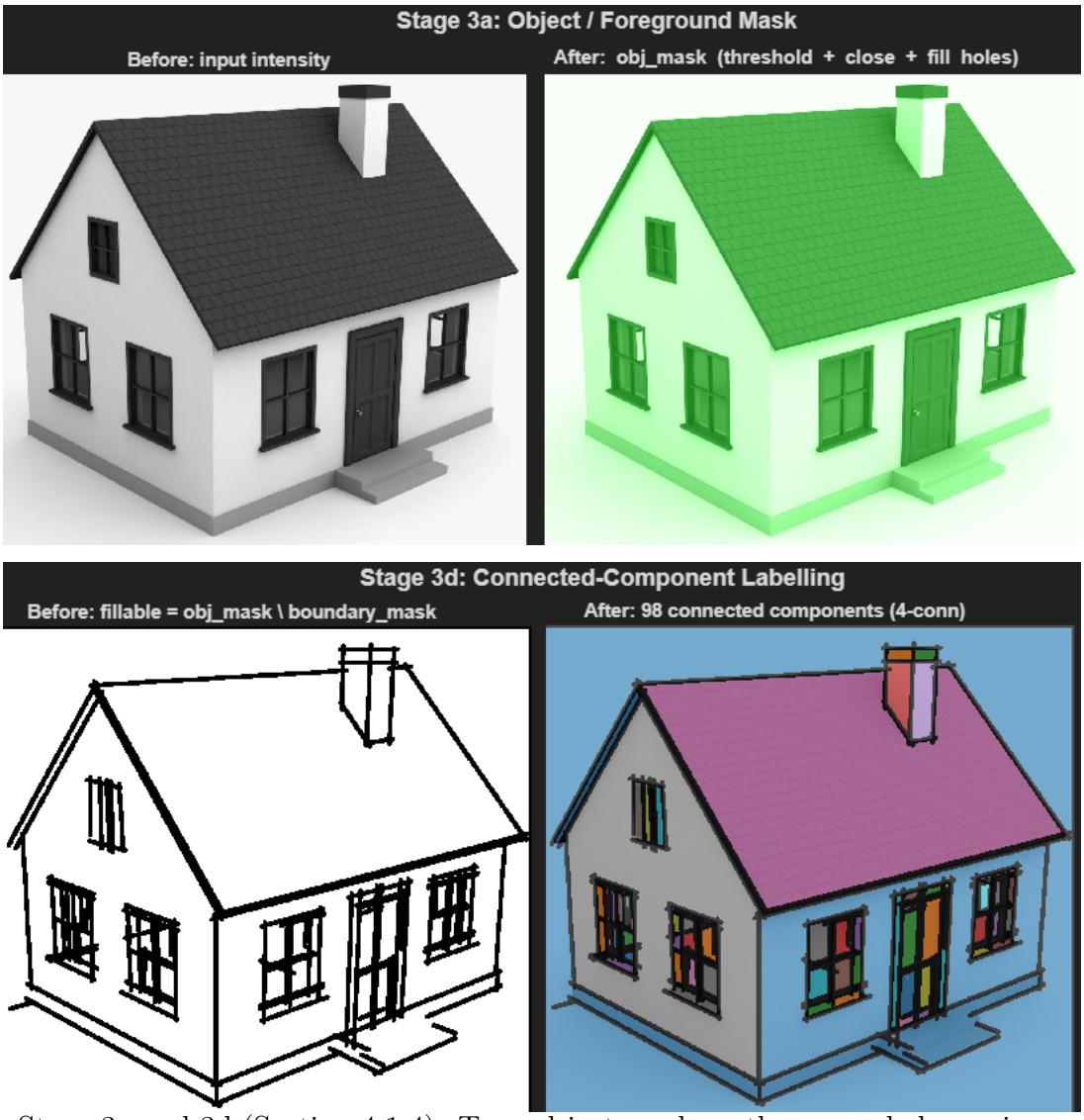


Figure 22: Stage 3a and 3d (Section 4.1.4). Top: object mask on the example house image. Bottom: connected-component labelling on the example house image.

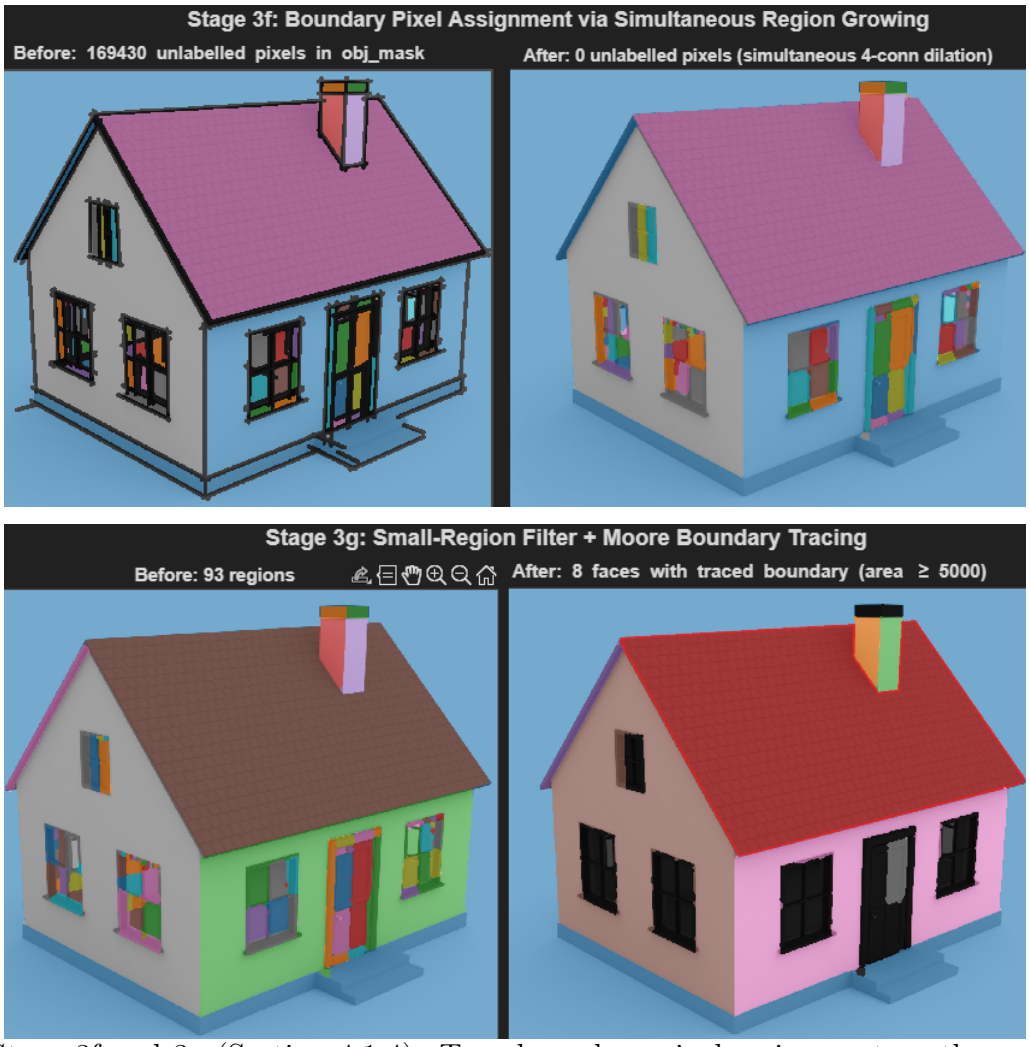


Figure 23: Stage 3f and 3g (Section 4.1.4). Top: boundary pixel assignment on the example house image. Bottom: face extraction on the example house image.



Figure 24: Stage 3h (Section 4.1.4): absorption of sub-threshold components on the example house image.

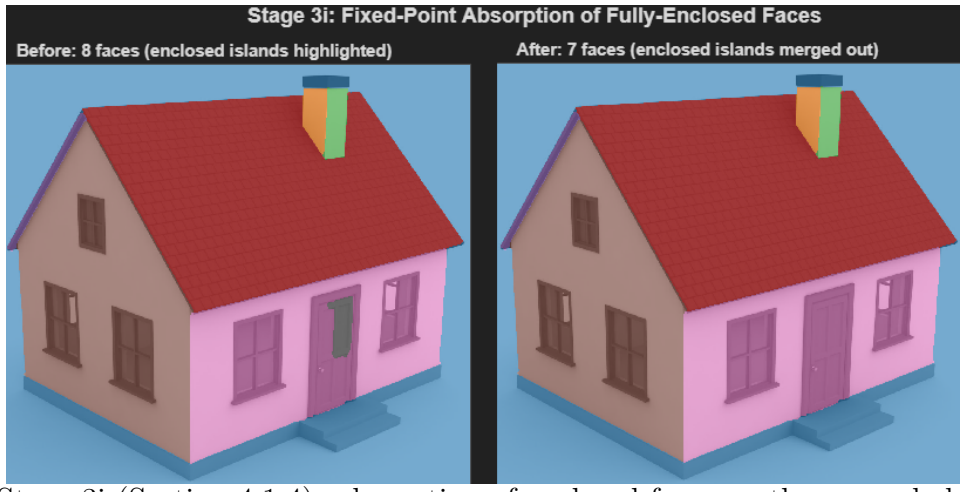


Figure 25: Stage 3i (Section 4.1.4): absorption of enclosed faces on the example house image.

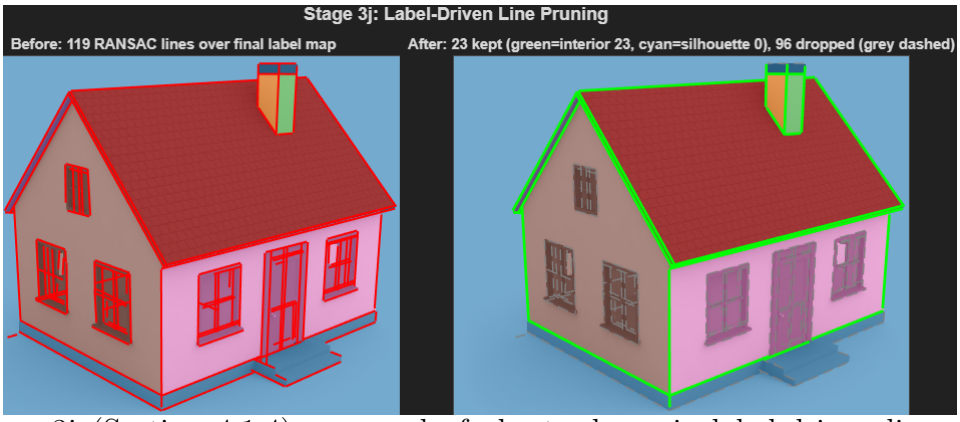


Figure 26: Stage 3j (Section 4.1.4): removal of ghost edges via label-driven line pruning on the example house image.

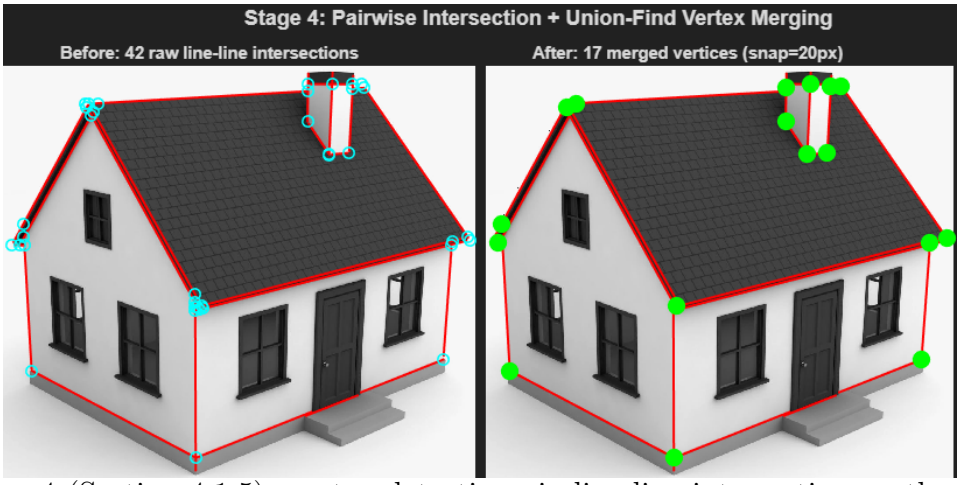


Figure 27: Stage 4 (Section 4.1.5): vertex detection via line-line intersection on the example house image.

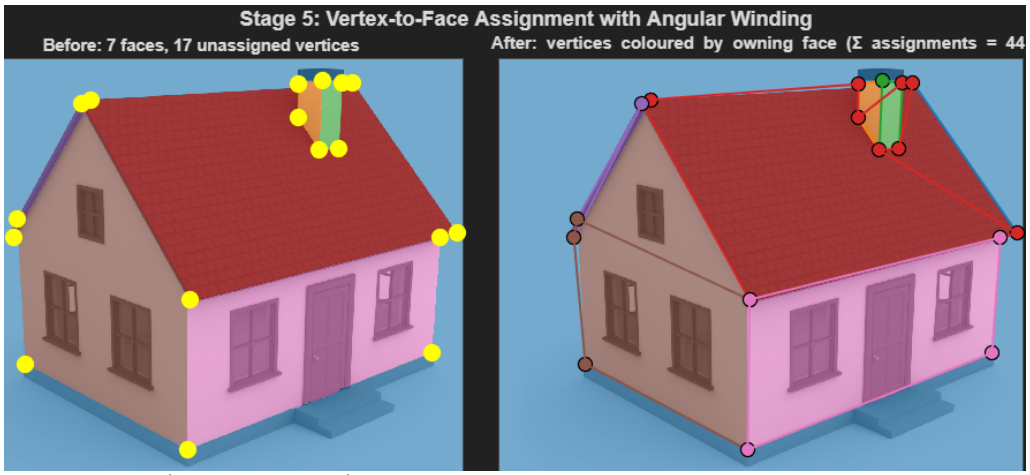


Figure 28: Stage 5 (Section 4.1.6): vertex-to-face assignment on the example house image.

9 Appendix: Project Management

Aims The project's main aim is to design, implement, and optimise a mathematical model capable of accurately dewarping fisheye lens images and reconstructing 3d geometries of real-world structures. The ultimate goal is for it to reconstruct fisheye lens photos of trees, as this would be a challenging task since the distance to the trees will be hard to differentiate between trees.

The central research question is how projective geometry and nonlinear optimisation techniques can be utilised to correct fisheye distortion and create precise 3D mappings from 2D images.

The project will be focused on understanding and researching the main maths behind all these machine learning techniques to develop and refine a custom model in MATLAB that can model the distortion behaviour of fisheye lenses and perform inverse transformations for accurate spatial recovery.

Performance of the model will be evaluated using RMSE and Structural Similarity Index (SSIM), as well as computational speed. The goal is to produce a model that balances accuracy and efficiency.

Objectives

- Research and create a mathematical model to describe fisheye lens distortion using projective geometry
  Measurable outcome: A set of equations and MATLAB functions that accurately describe how fisheye projection behaves
- Implement a dewarping algorithm based on the previous model's maths to transform fisheye lens photos into undistorted rectangle images
  Measurable outcome: A MATLAB implementation that can dewarp images with good accuracy, evaluated using RMSE and SSIM
- Design and optimise the 3D reconstruction algorithm that maps the dewarped images into a 3D coordinate space using nonlinear optimisation techniques
  Measurable outcome: A MATLAB implementation that can produce realistic 3D geometry of test structures, evaluated using defined accuracy benchmarks
- Evaluate model performance using multiple datasets of structures and natural environments.
  Measurable outcome: An analysis report quantifying the model's accuracy and processing speed against baseline methods

Activities and Methodology The project has 3 main steps: study, model development, and computational implementation. We will utilise fisheye geometry to process and dewarp fisheye images, which will subsequently be used in 3D reconstruction using projective geometry.

1. **Literature Review and Background Study**
   - **Fisheye Geometry**
     - Review fisheye projection models
     - Study lens calibration and rectification methods
     - Evaluate the suitability of different projection models to use for dewarping
   - **2D - 3D Geometry**
     - Study mathematical image processing to process the dewarped images
     - Investigate projective geometry for mapping 2D images to 3D structures
     - Study reconstruction algorithms to convert image data into geometric representations
     - Evaluate non-linear optimisation methods to improve accuracy
   - **Partial Differential Equations**
     - Study PDE-based models used in image processing
     - Explore their application to smoothing, edge detection, or regularisation in fisheye image dewarping
2. **Model Development**
   - Define fisheye distortion models and derive mapping function for dewarping 2D fisheye lens photos
   - Derive projective geometry function for accurate 2D-3D correspondence
   - Incorporate PDE-based regularisation where appropriate for image correction and stability
3. **Computational Implementation**
   - Distortion and dewarping algorithms will be implemented in MATLAB
   - Implement parameter estimation using nonlinear optimisation
   - Validate outputs using RMSE and SSIM against ground-truth or synthetic benchmarks

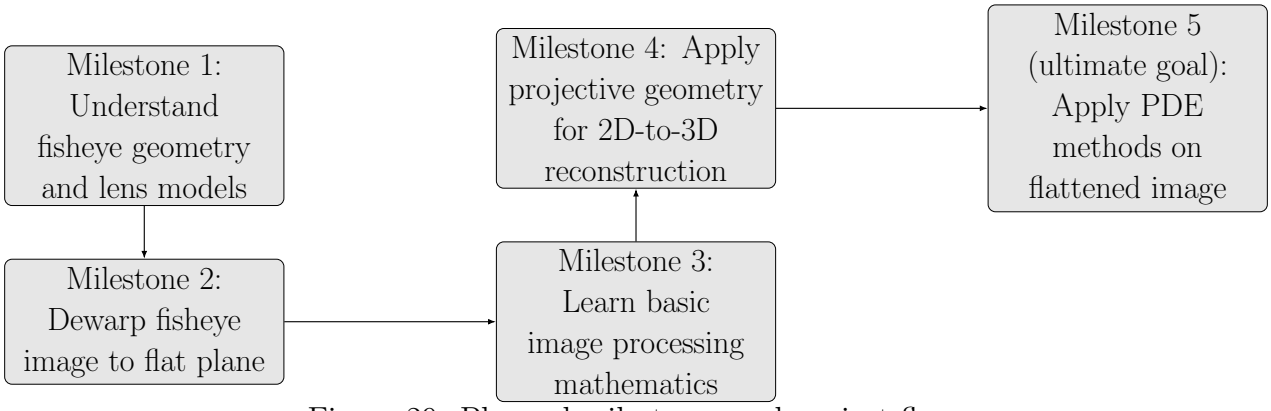


Figure 29: Planned milestones and project flow.

Plan and Milestones

Agreed Scope and Budget The project is focused on researching, developing and implementing a mathematical framework to dewarp fisheye-lens images and reconstruct the corresponding 3D geometry of photographed structures. The agreed scope includes:

- Acquiring fisheye images of public structures and natural environments - we will use public open datasets; no work involves personal data, and all images will exclusively feature non-personal, publicly visible objects.

- All implementing, modelling and evaluating will be done on MATLAB - the university provides this.

The budget required for this project is effectively zero. No external services, specialised hardware, or paid datasets are needed. The scope of this project is to operate entirely within existing resources, with no projected costs beyond standard university-provided software and equipment. Any incidental expenditures, such as optional printing or storage, would be minimal and not essential to completing the project.

Risk Type	**Description**	**Likelihood**	**Impact**	**Severity**	**Mitigating Plan**
Management	Infrequent or unclear communication with the supervisor may lead to misaligned objectives or misunderstandings about modelling decisions.	Medium	Medium	Medium	Hold regular meetings; seek clarification early on modelling or optimisation concerns.
Timeline	Nonlinear optimisation, projective-geometry derivations, or reconstruction debugging may require more time than planned.	High	High	High	Maintain a weekly schedule; add buffer time for debugging; keep supervisor updated with progress
Experiment / Technical	MATLAB optimisation may fail to converge; fisheye calibration might not generalise across images; unexpected artefacts may affect reconstruction quality.	Medium	Medium	Medium	Validate intermediate steps; test on simpler synthetic data; use MATLAB debugging or visualisation.
Data Quality	Captured fisheye images may contain blur, low light, or insufficient geometry detail, reducing output accuracy.	Medium	Medium	Medium	Use many different samples; pre-check images; supplement with open datasets if needed.
Privacy	Although images target structures or trees, outdoor capture may inadvertently include individuals.	Low	Low	Low	Review and discard any images containing people; ensure all datasets contain only non-personal content.
Legal	MATLAB licence is covered by the University; datasets are either open-source or self-captured, giving minimal legal risk.	Low	Low	Low	Use only university-licensed MATLAB; verify dataset licences; ensure self-captured images do not depict restricted locations.
Health & Safety	Long screen time for coding, debugging, and visual inspection of images may lead to fatigue or eye strain.	Low	Low	Low	Follow computer-use safety guidance; take regular breaks; adjust display brightness and sit ergonomically.
Environmental Impact	Fully software-based work with negligible energy use; MATLAB computations are light and local.	Low	Low	Low	Use efficient computation; avoid unnecessary re-runs and repeated high-resolution recalculations.

Risk Assessment Health and Safety Statement: There are no high-risk health and safety risks involved with this project; the main considerations relate to prolonged computer use. This can be mitigated by taking regular breaks and adjusting the screen brightness.

Environmental Impact Statement: This project does not actively interact with the environment. The project is conducted entirely on software. No external hardware (e.g. GPUs) or data storage is required for this project; thus, the environmental impact from the computation is near zero.

Data Management Plan We will store the code and any other related work (reconstructed 3D outputs, optimisation results, etc.) using a private GitHub repository to ensure version control and secure storage of the code. All data will be organised in a clear directory structure, with descriptive filenames.

The project will make use of publicly available fisheye-image datasets and self-captured photographs of non-personal, publicly visible structures. All external datasets used in the project originate from open and freely accessible sources, ensuring that licensing and reuse conditions are fully compliant with academic use.

The project doesn't involve any personal information; thus, no formal data destruction or anonymisation procedures are required. The stored data will not be shared publicly unless the dataset used permits redistribution under its respective open licenses.

Commercial Discussion The project's focus on reconstructing the 3D geometry of structures from fisheyelens photos has potential commercial applications, particularly in fields such as virtual reality, architecture and robotics. The use of publicly available datasets would significantly reduce the cost of 3D modelling when capturing high-quality 3D data isn't feasible. The commercial potential is limited by reliance on specific datasets, which could restrict its use in real-time applications. Furthermore, 3D reconstruction techniques can present intellectual property challenges. We would also need to consider any licensing issues related to the use of MATLAB.

Ethics Our project involves no human subjects, biological specimens, or copyrighted material. We will use publicly available datasets that consist of non-personal, publicly visible structures, such as trees and buildings. The project is entirely computational and thus doesn't involve any ethical dilemmas related to human or animal rights. Our project has low ethical concerns.

References

[1] K. A. Scott, "Fish eye lens dewarping and panorama stitching." https://www.kscottz.com/2013/08/, Aug. 2013.

[2] AI/HUB, "3d reconstruction with stereo images - part 1: Camera calibration." https://medium.com/@dc.aihub/3d-reconstruction-with-stereo-images-part-1-camera-calibration-d86f750a1ade, Oct. 2018.

[3] E. Plaut, "Tutorial on computer vision with fisheye cameras." https://plaut.github.io/fisheye_tutorial/, 2025.

[4] V. Rengarajan, "Ee5176 computational photography - basics of imaging." https://apvijay.github.io/ee5176/imaging/, 2025.

[5] K. Hata and S. Savarese, "Cs231a course notes 1: Camera models," course notes, Stanford University, 2025. https://web.stanford.edu/class/cs231a/course_notes/01-camera-models.pdf.

[6] Wikipedia, "Thin lens." https://en.wikipedia.org/wiki/Thin_lens, 2025.

[7] K. Sadekar, "Understanding lens distortion." https://learnopencv.com/understanding-lens-distortion/, Oct 5 2020.

[8] D. D. T. Jr., "Circular fisheye view of oude kerk amsterdam." Wikimedia Commons, 2014. CC BY-SA 4.0, https://creativecommons.org/licenses/by-sa/4.0/, https://commons.wikimedia.org/wiki/File:Circular_fisheye_view_of_Oude_Kerk_Amsterdam_Daniel_D._Teoli_Jr.jpg.

[9] C. Sahin, "Comparison and calibration of mobile phone fisheye lens and regular fisheye lens via equidistant model," *Journal of Sensors*, vol. 2016, pp. 1–11, 01 2016.

[10] J. Kannala and S. Brandt, "A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 8, pp. 1335–1340, 2006.

[11] U. Gunes, M. Turkulainen, X. Ren, A. Solin, J. Kannala, and E. Rahtu, "Fiord: A fisheye indoor-outdoor dataset with lidar ground truth for 3d scene reconstruction and benchmarking," 2025.

[12] Wikipedia, "Ternary search." `https://en.wikipedia.org/wiki/Ternary_search`, 2026.

[13] R. Szeliski, *Computer Vision: Algorithms and Applications*. Springer, 2 ed., 2022.

[14] OpenCV Development Team, "Canny edge detection - OpenCV 3.4 documentation," 2024. Accessed: 2 March 2026.

[15] M. Nixon and A. S. Aguado, *Feature Extraction & Image Processing for Computer Vision*. Academic Press, 3 ed., 2012.

[16] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, pp. 381–395, June 1981.

[17] M. Zuliani, C. Kenney, and B. Manjunath, "The multiransac algorithm and its application to detect planar homographies," in *IEEE International Conference on Image Processing 2005*, vol. 3, pp. III–153, 2005.

[18] I. Jolliffe, *Principal Component Analysis*. John Wiley Sons, Ltd, 2014.

[19] The scikit-learn developers, "Robust linear model estimation using RANSAC - scikit-learn documentation." `https://scikit-learn.org/stable/auto_examples/linear_model/plot_ransac.html`. Accessed: 2026-04-27.

[20] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars, *Computational Geometry: Algorithms and Applications*. Berlin, Heidelberg: Springer, 3 ed., 2008.

[21] A. Rosenfeld and J. L. Pfaltz, "Sequential operations in digital picture processing," *J. ACM*, vol. 13, pp. 471–494, Oct. 1966.

[22] A. Tremeau and P. Colantoni, "Regions adjacency graph applied to color image segmentation," *IEEE Transactions on Image Processing*, vol. 9, no. 4, pp. 735–744, 2000.

[23] R. C. Gonzalez and R. E. Woods, *Digital image processing*. Upper Saddle River, N.J.: Prentice Hall, 2008.

[24] B. Horn and M. Brooks, *Shape from Shading*, vol. 2. 01 1989.

[25] A. L. Yuille, "Lecture 10: The Lambertian reflectance model." Lecture notes, Stat238, Johns Hopkins University, February 2012. Accessed: 2026-03-10.

[26] CCS Inc., "Luminance properties." `https://www.ccs-grp.com/guide/theory-light-color/09_luminance-properties.html`, 2026. Accessed: 2026-04-28.

[27] P. N. Belhumeur, D. J. Kriegman, and A. L. Yuille, "The bas-relief ambiguity," *International Journal of Computer Vision*, vol. 35, pp. 33–44, Nov. 1999.

[28] D. P. Bertsekas, *Nonlinear Programming*. Belmont, MA: Athena Scientific, 2nd ed., 1999.

[29] J. Kiefer, "Sequential minimax search for a maximum," *Proceedings of the American Mathematical Society*, vol. 4, no. 3, pp. 502–506, 1953.

[30] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *Journal of the Society for Industrial and Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963.