Image Processing for Projected Grid Extraction

Nobuaki Tanaka

November 15, 2025

1 Overview

In this note we describe the image processing pipeline used to extract projected grid patterns from camera images. The goal is to obtain a one-pixel–wide skeleton of the grid lines and to detect their intersection points (junctions) in a robust and geometrically meaningful way.

Starting from a grayscale image $I(x, y)$, the pipeline consists of the following steps:

1. Gaussian filtering for noise reduction and smoothing.
2. Binarization and region filling to obtain a clean foreground mask.
3. Skeletonization (thinning) of the grid lines.
4. Extraction of junctions using connectivity numbers on the skeleton.

2 Gaussian filtering

Let $I(x, y)$ be the input grayscale image, where (x, y) denotes pixel coordinates on a regular grid. To suppress high-frequency noise while preserving the overall structure of the grid, we apply a Gaussian filter with standard deviation σ.

The continuous 2D Gaussian kernel is

$$G_\sigma(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right),$$

and the filtered image I_σ is defined by the convolution

$$I_\sigma(x, y) = (G_\sigma * I)(x, y) = \sum_{u,v} G_\sigma(u, v)\, I(x - u, y - v).$$

In practice, the kernel is discretized to a finite support (e.g. 5×5 or 7×7) and normalized so that $\sum_{u,v} G_\sigma(u, v) = 1$. The parameter σ is chosen according to the expected line width of the projected grid: too small values do not remove noise, whereas too large values blur the grid lines excessively.

3 Binarization and region filling

From the smoothed image $I_\sigma(x, y)$ we construct a binary mask $B(x, y) \in \{0, 1\}$ that separates grid lines from background. A simple global thresholding rule is

$$B(x, y) = \begin{cases} 1, & I_\sigma(x, y) \geq \tau, \\ 0, & I_\sigma(x, y) < \tau, \end{cases}$$

where τ is the intensity threshold. More advanced methods (e.g. Otsu's method or adaptive thresholding) can also be used but we keep the formulation simple here.

After binarization, small gaps may appear along the grid lines due to illumination variations, specular highlights, or sensor noise. To obtain a clean foreground region, we apply region filling and morphological operations:

- **Hole filling:** starting from the image boundary, a flood-fill is performed over background pixels ($B = 0$); any background region that is not connected to the boundary is considered a *hole* and is filled (set to 1). This removes small dark spots inside grid lines.
- **Closing:** a morphological closing (dilation followed by erosion) with a small structuring element connects nearby components and bridges small gaps between line segments.

The result is a binary image $F(x, y)$ in which the foreground ($F = 1$) forms thick grid lines with as few holes and gaps as possible.

4 Skeletonization (thinning)

The next step is to obtain a one-pixel–wide representation of the grid while preserving its topology (connectivity and junction structure). We denote the skeletonized image by $S(x, y) \in \{0, 1\}$.

Skeletonization is implemented as an iterative thinning process. At each iteration, boundary pixels of the foreground are examined and deleted if they satisfy a local deletion condition that guarantees:

1. The deletion does not disconnect any connected component.
2. The deletion does not create or destroy holes.
3. Endpoints of line segments are preserved.

Formally, let $N(p)$ be the 3×3 neighborhood of a pixel $p = (x, y)$ in F, and let p be a foreground pixel ($F(x, y) = 1$). A thinning algorithm (e.g. Zhang–Suen type) specifies a set of patterns on $N(p)$ for which p can be safely set to 0. The deletion is applied simultaneously (or in sub-iterations) to all such pixels, and the process is repeated until no more pixels can be deleted. The resulting binary image

$$S(x, y) \in \{0, 1\}$$

is the skeleton: each grid line is reduced to a one-pixel–wide curve, and the junctions where lines intersect are preserved as branching points.

5 Junction extraction using connectivity numbers

To detect grid intersections (junctions) on the skeleton S, we analyze the local connectivity around each foreground pixel. We work with the 8-neighborhood of a pixel $p = (x, y)$:

$$\begin{array}{ccc} p_1 & p_2 & p_3 \\ p_8 & p & p_4 \\ p_7 & p_6 & p_5 \end{array}$$

where p_i denotes the neighboring pixel position and we write

$$S_i = S(p_i) \in \{0, 1\}, \qquad i = 1, \ldots, 8.$$

We define the *connectivity number* $C(p)$ of a pixel p as

$$C(p) = \frac{1}{2} \sum_{i=1}^{8} \left| S_i - S_{i+1} \right|, \qquad S_9 = S_1.$$

Intuitively, $C(p)$ counts the number of $0 \to 1$ transitions when we traverse the 8-neighborhood in a circular order. For skeleton pixels, the connectivity number has the following typical values:

- $C(p) = 1$ and exactly two foreground neighbors $\Rightarrow$ p lies on a simple line segment.
- $C(p) = 1$ and exactly one foreground neighbor $\Rightarrow$ p is an endpoint of a line.
- $C(p) \geq 3 \Rightarrow$ p is a junction (branch point or crossing).

In our implementation, we identify junction candidates as skeleton pixels p that satisfy

$$S(x, y) = 1, \qquad C(p) \geq 3.$$

Optionally, small clusters of adjacent junction candidates can be merged (e.g. by connected-component labeling) and replaced by a single representative point (the centroid of the cluster). This yields a sparse set of grid intersection points that can be used as feature points for matching and subsequent triangulation.

6 Summary

The combination of Gaussian filtering, region filling, skeletonization, and connectivity-based junction extraction provides a robust way to convert a noisy image of a projected grid into a clean set of grid intersections. These intersections are then matched with the corresponding points on the smartphone display pattern and used as input for the structured-light triangulation described in the previous notes.