2025 SoC Lab 期末專題報告

基於 FPGA 之 Canny Edge Detection 實作

姓名：吳楚熙、楊啟弘

學號：7113056146、7113056083

班級：資訊工程系研究所

一、前言

本期末專題將選用 ZedBoard 作為開發平台，結合 Vivado 開發環境，實作出 Canny 邊緣偵測演算法，並且顯示在 VGA 介面顯示上。

Canny 邊緣偵測演算法由 John F. Canny 在 1986 年提出，是一種高品質、低誤報的多階段邊緣檢測方法，其核心即是同時滿足三大準則：

1. 低偵測錯誤率：儘量偵測到所有真實邊緣且減少誤報。
2. 高定位精度：標示的邊緣點要貼近真實邊緣中心。
3. 唯一回應原則：同一條邊緣只能被標示一次，避免多重回應。

演算法共包含五個關鍵步驟：

1. 高斯平滑：抑制雜訊。
2. 梯度計算：得出每個像素的梯度幅值與方向。
3. 非極大值抑制：保留局部最大響應以將邊緣細化為單像素寬。
4. 雙閾值分割：將強邊緣與弱邊緣區分開來。
5. 邊緣追蹤與連通：從所有強邊緣像素出發，將與之相連的弱邊緣一併納入最終邊緣圖，孤立的弱響應則予以捨棄。

透過此順序處理，Canny 能在抑制雜訊與維持邊緣完整性間取得理想平衡（圖 1）。

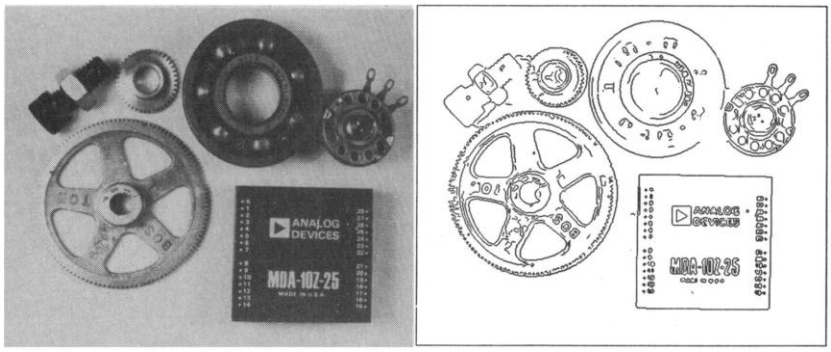


圖 1、Canny 邊緣偵測演算法示意圖（左圖為原圖；右圖為結果）

二、演算法探討

Canny 演算法包含五個依次執行的階段——從去噪到邊緣連通，環環相扣。

1. 高斯平滑（Gaussian Smoothing）：

   以標準差 σ 的二維高斯濾波器 G_σ 對輸入影像 $I_{(x,y)}$ 卷積，以去除高頻雜訊，輸出平滑影像：

$$I_{s(x,y)} = I * G_\sigma$$

2. 梯度計算(Gradient Computation)：

   對平滑後影像 Is 分別套用水平與垂直的一階導數運算子（如 Sobel 或高斯一階導數），得到梯度分量：

$$G_{x(x,y)} = (I_s * S_x)(x,y), \quad G_{y(x,y)} = \big(I_s * S_y\big)(x,y)$$

   接著計算梯度幅值與方向：

$$M(x,y) = \sqrt{G_x(x,y)^2 + G_y(x,y)^2}$$

$$\Theta(x,y) = atan2(G_y(x,y), G_x(x,y))$$

3. 非極大值抑制（Non-Maximum Suppression）：

   對每個像素，根據其梯度方向 $\Theta(x,y)$，分別比較它與梯度方向前後（正反向）相鄰兩點的幅值：

- 若當前幅值非三者中最大，則將其設為 0；
- 否則保留原值。

此舉可將寬幅的邊緣響應「細化」為單像素寬的銳利輪廓，顯著改善定位精度。

4. 雙閾值分割（Double Thresholding）：

以兩道閾值 T_H（高閾）與 T_L（低閾）將非極大值抑制後的幅值分類：

- $M \geq T_H$：標為「強邊緣」（Strong Edge）；
- $T_L \leq M < T_H$：標為「弱邊緣」（Weak Edge）；
- $M < T_L$：「非邊緣」，直接捨棄。

此機制一方面保留強邊緣，同時也暫存可能因雜訊導致幅度較低但連續的邊緣段，以利後續連通處理。

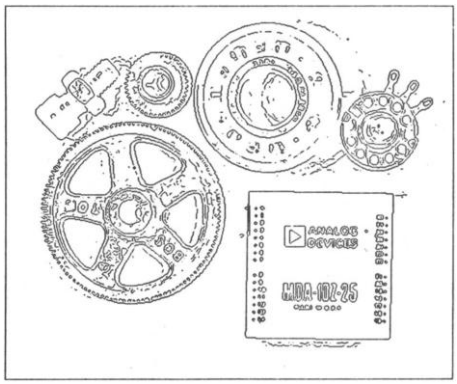


圖 2、低閾值 *TL* 的結果：以 *TL* 為分割閾值直接保留所有梯度強度 $\geq TL$ 的像素，會偵測到大部分邊緣，但同時也會產生許多雜訊假邊緣和邊緣斷裂。

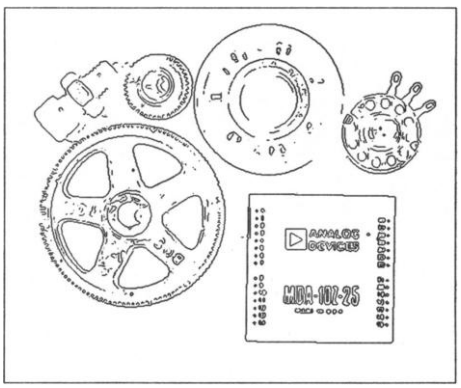


圖 3、高閾值 TH=2TL 的結果：以更高的閾值 TH 為分割標準，只保留最強的邊緣響應，雖然雜訊明顯減少，卻又漏檢了許多真實但偏弱的邊緣段。

5. 邊緣連通與磁滯追蹤（Edge Tracking by Hysteresis）

從所有強邊緣像素出發，沿著 8-連通方向追蹤：

- 將所有與強邊緣相連的弱邊緣納入最終邊緣；
- 為連通的孤立弱邊緣則棄置。

由此既可剔除獨立雜訊，又能補全因單一閥值而產生的邊緣中斷。

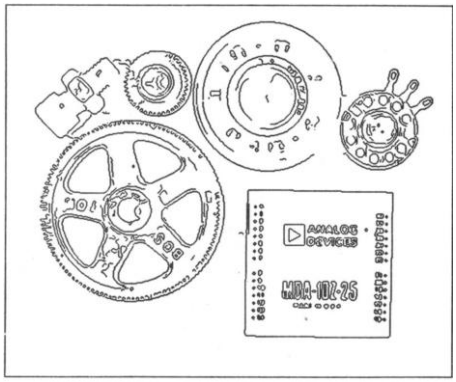


圖 4、雙閾值 + 磁滯追蹤：先以兩道閾值 TH 和 TL 將像素分為強邊緣、弱邊緣與非邊緣；再從所有強邊緣出發，將所有 8 連通的弱邊緣像素納入最終結果，其餘孤立弱邊緣則捨棄。如此可兼顧偵測靈敏度與邊緣連續性——既降低了假邊緣，又補全了因單一閾值而中斷的真實邊緣。

三、實作

本專題採用 ZedBoard 硬體平台，並以 Vivado 開發環境進行設計與實作。目標為整合 ZedBoard 上的 Processing System（PS）與 Programmable Logic（PL），實現 Canny 邊緣偵測演算法。系統中，PS 主要負責影像資料的載入、資料搬移與流程控制，以及驅動PL端的ip；PL 則專注於 Canny 演算法的硬體加速，充分發揮 FPGA 的高度並行特性，並以 Pipeline 架構提升處理效能。

圖 5 為透過 Vivado 所開發的 Block Design，所實現出的 Canny 演算法硬體加速與 VGA 顯示輸出功能。

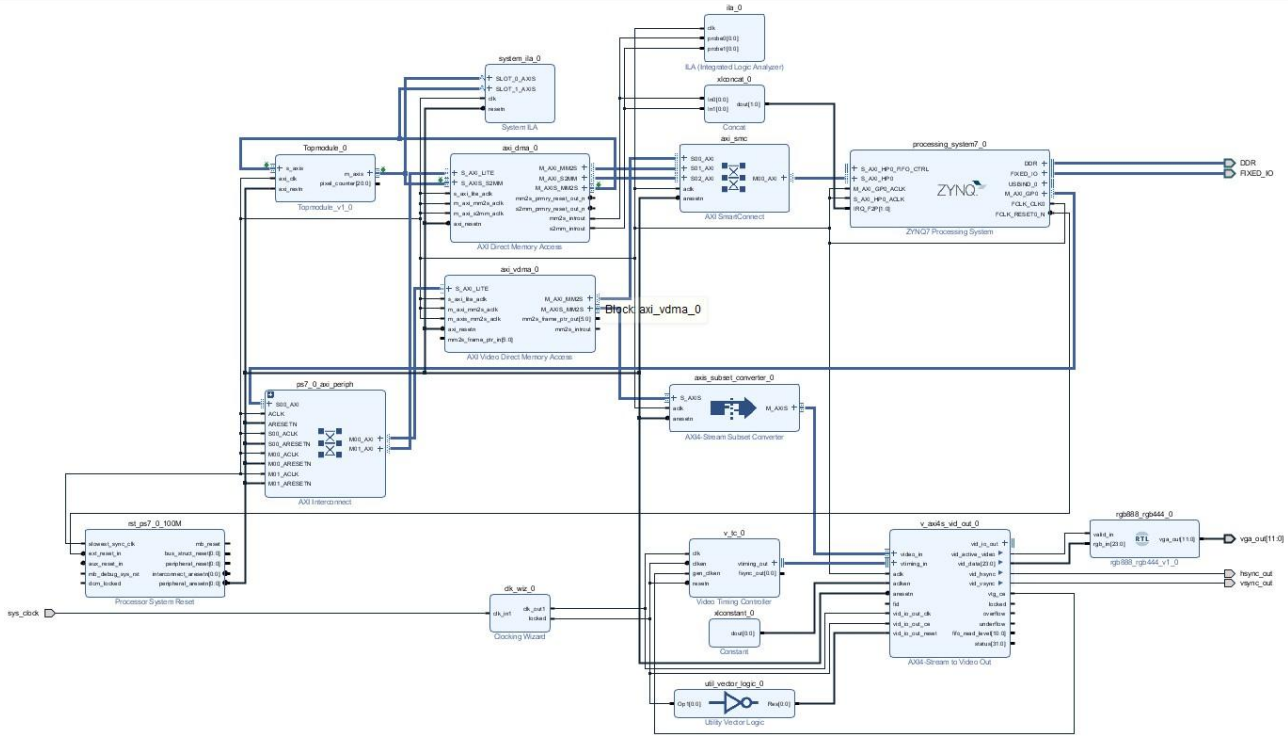


圖5、Block Design

以下為主要模組與資料流分工：

1. PS（Processing System）
   - 影像資料載入：PS 端於啟動階段將 .h 檔案中的圖像資料（RGB 格式）載入，並寫入 DDR 記憶體中指定位置。
   - DMA 控制與資料搬移：透過 AXI DMA 將影像資料從 DDR 傳送至 PL，自訂硬體canny IP［圖6］處理完後再寫回 DDR。DMA 為一次性批次搬運（Memory-Mapped to Stream, Stream to Memory-Mapped）。
   - VDMA 控制：PS 負責設定 VDMA 參數，包括影像尺寸、位址、Stride 等，VDMA 負責將影像資料從 DDR 傳送至video out ip。
   - 按鈕切換控制：PS 偵測按鈕輸入，依據使用者操作切換 VGA 顯示來源（原始影像或處理後影像），透過控制 VDMA 設定的 base address 實現。

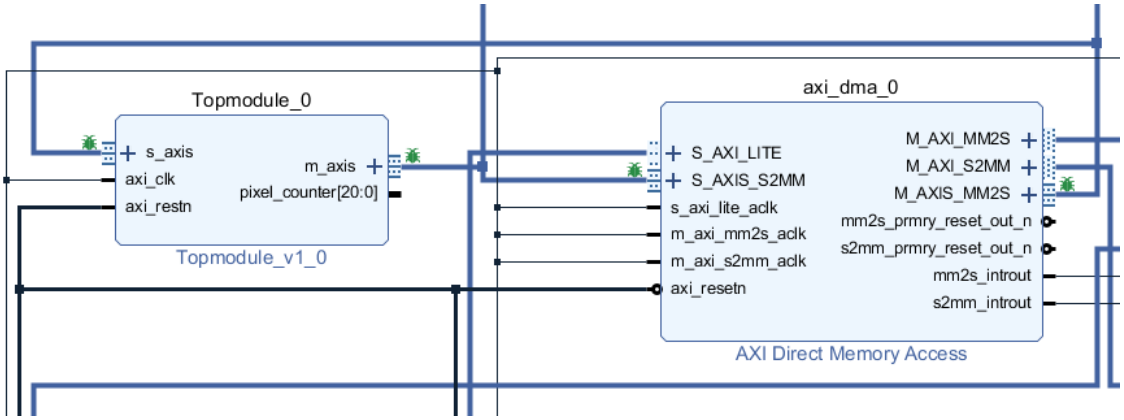


圖6 、canny IP以及DMA

2. PL (Programmable Logic)
   - Canny 邊緣偵測硬體加速器：使用自訂 IP Topmodule_0 以多階段 Pipeline 實作 Canny 邊緣偵測，包括灰階轉換、Gaussian 模糊、Sobel 偵測、非最大抑制與雙閾值處理等。
   - AXI4-Stream 資料流：資料在 IP 間透過 AXI4-Stream 傳遞，串聯順序如下：

     DMA → RGB2Gray → Gaussian → Sobel → NMS → Thresholding → DMA。
   - 多時脈設計與顯示時脈匹配：本系統為異步多時脈架構。影像處理模組（包含 Canny IP）皆以 PS 提供之 100 MHz 系統時脈運作，確保與 DMA 傳輸同步。而為符合 1920×1080 @ 60 Hz VGA 顯示需求，AXI4-Stream to Video Out 與 Video Timing Controller (VTC) 則採用 108 MHz 的 pixel clock，以正確生成顯示同步訊號與畫面輸出時序。
   - 格式與同步訊號控制：
     - 透過 Video Timing Controller (VTC) 產生 hsync、vsync、active_video 等同步訊號。
     - 使用 AXI4-Stream to Video Out 模組將資料轉換為 VGA 格式（如 RGB444），搭配 VTC 驅動 VGA 顯示輸出。

最終成功在 ZedBoard 上成功實作出Canny演算法，透過開發板上的按鈕，將顯示在VGA顯示器上[圖7]的原影像，切換成處理完成的邊緣影像。

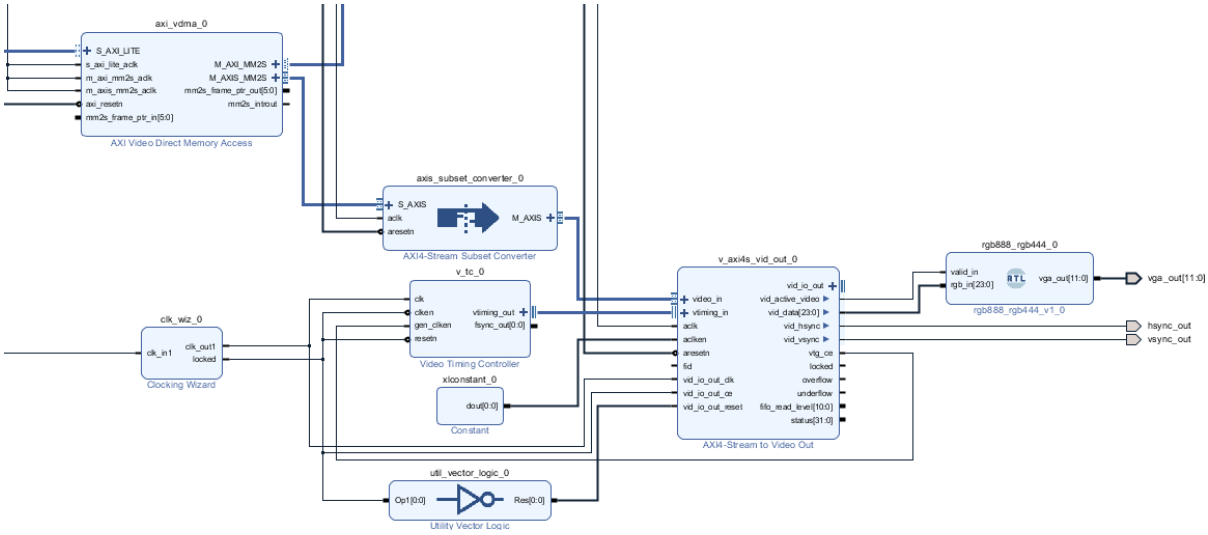


圖7、 vga顯示所需的block

四、合成結果

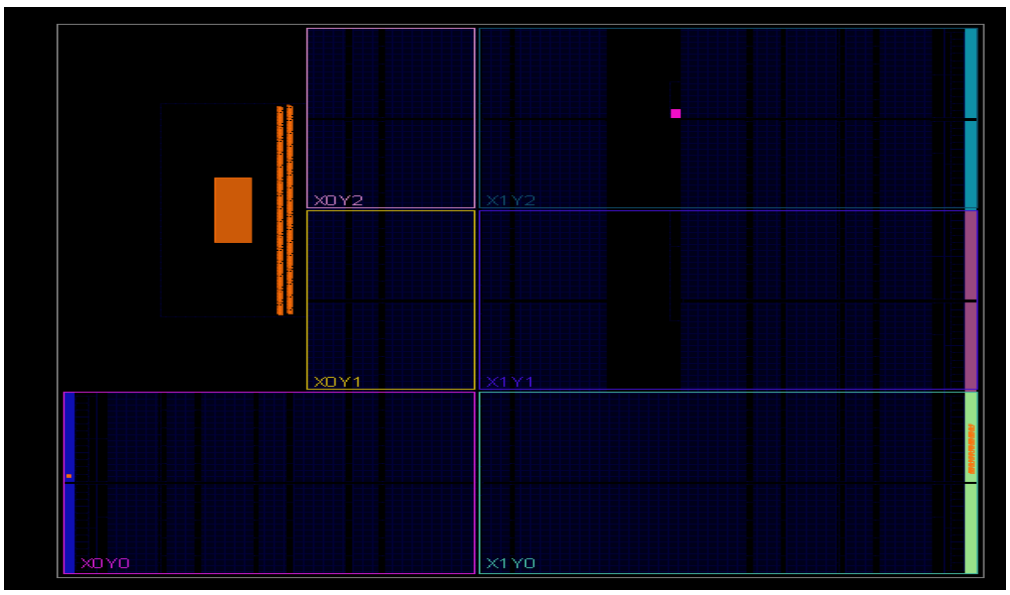
Timing:

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	2.241 ns	Worst Hold Slack (WHS):	0.045 ns	Worst Pulse Width Slack (WPWS):	3.000 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	40025	Total Number of Endpoints:	39983	Total Number of Endpoints:	14012

All user specified timing constraints are met.

Implementation Design:



Utilization:

Hierarchy

Name	Slice LUTs (53200)	Slice Registers (106400)	F7 Muxes (26600)	Block RAM Tile (140)	DSPs (220)	Bonded IOB (200)	Bonded IOPADs (130)	BUFGCTRL (32)	MMCME2_ADV (4)
design_1_wrapper	9885	13706	6	14	1	15	130	3	1
dbg_hub (dbg_hub_CV)	0	0	0	0	0	0	0	0	0
design_1_i (design_1)	9885	13706	6	14	1	0	0	3	1

Hierarchy

Name	Slice LUTs (53200)	Slice Registers (106400)	F7 Muxes (26600)	Block RAM Tile (140)	DSPs (220)	Bonded IOB (200)	Bonded IOPADs (130)	BUFGCTRL (32)	MMCME2_ADV (4)
design_1_wrapper	18.58%	12.88%	0.02%	10.00%	0.45%	7.50%	100.00%	9.38%	25.00%
dbg_hub (dbg_hub_CV)	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
design_1_i (design_1)	18.58%	12.88%	0.02%	10.00%	0.45%	0.00%	0.00%	9.38%	25.00%

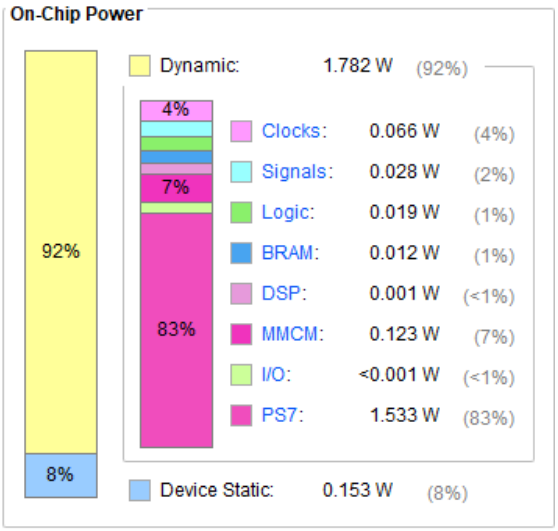
Power:

Summary

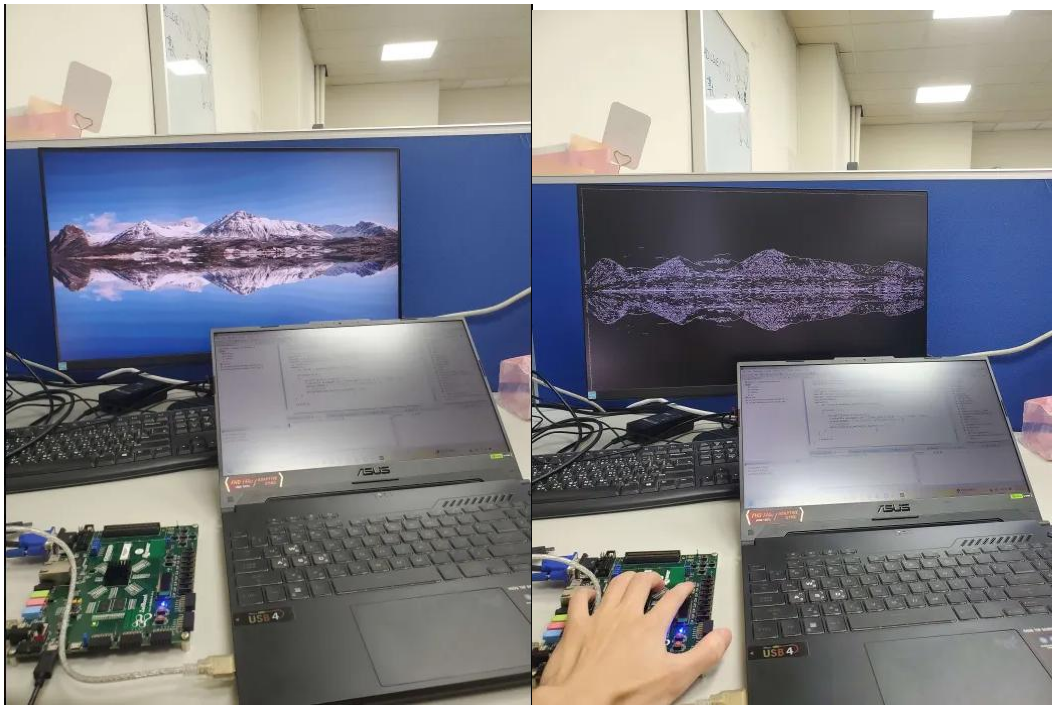
Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

Total On-Chip Power:	**1.935 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**47.3°C**
Thermal Margin:	37.7°C (3.1 W)
Effective ϑJA:	11.5°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Medium

Launch Power Constraint Advisor to find and fix invalid switching activity



五、實作結果



左圖為處理前，右圖為處理後。

六、參考文獻

1. https://github.com/DOUDIU/Hardware-Implementation-of-the-Canny-Edge-Detection-Algorithm
2. https://www.bilibili.com/video/BV1pq4y127Wt/?spm_id_from=333.1387.upload.video_card.click&vd_source=3382bd3e6cdd370a44a1bd420eede3c6
3. https://ieeexplore.ieee.org/document/4767851