各章導讀

- **第 1 章 單板電腦與邊緣運算**：介紹邊緣運算裝置的發展背景與其應用，強調其低功耗、高效率及在本地端進行運算的優勢。同時簡介單板電腦的特性與應用場景，聚焦 NVIDIA Jetson 系列的發展歷程與技術優勢，特別是 Jetson Orin Nano 的規格與實用性。之後介紹了 NVIDIA 提供的學習資源與認證機制，開發者可充分運用 DLI 深度學習機構所規劃的完整學習路徑來提升專業能力。透過本章，讀者將可掌握邊緣運算與單板電腦的基礎概念。

- **第 2 章 Jetson Orin Nano 初體驗**：聚焦於引導讀者完成 Jetson Orin Nano 的初始設定與基本操作，包括硬體準備、作業系統安裝、網路設定及遠端操作等內容。重點涵蓋開機準備（如 micro SD 卡或 SSD 固態硬碟）、硬體連接與網路設定（Wi-Fi 或乙太網路），以及利用 SSH 或 USB 連線進行遠端登入操作。最後還介紹了 jtop 系統監控工具，以及 USB 攝影機與 CSI 攝影機模組的連接與測試。透過本章，讀者將能順利啟動並熟悉 Jetson Orin Nano 的基本操作方式。

- **第 3 章 深度學習結合視覺辨識應用**：將帶領讀者在 Jetson Orin Nano 平台上實現電腦視覺應用，結合深度學習技術進行圖像處理與分析。內容涵蓋 OpenCV 的基礎使用（如拍照、灰階處理、顏色提取），以及 Jetson Inference 函式庫在圖像辨識、物件偵測、圖像分割等基本圖像辨識上的應用上，並說明 TensorRT 對於推論效能的提升。此外，還進一步提供姿勢估計、動作辨識、背景更換與距離估計等範例，幫助讀者在實用中掌握相關技術。本章奠定了 AI 視覺應用的基礎，並為下一章延伸到立體視覺與場景重建技術做好準備。

- **第 4 章 整合深度視覺**：介紹 NVIDIA Jetson Orin Nano 搭配兩款主流深度攝影機（Intel RealSense D435 和 StereoLab ZED2i）的應用，聚焦於景深技術如何實現三維立體視覺，提升裝置在自動化智慧系統中的表現。內容涵蓋深度視覺應用（如三維建模、物體追蹤、自主導航）、D435 與 ZED2i 的功能與安裝步驟，以及透過 Python 實作點雲生成、深度影像檢視與距離估算等操作範例。本章提供了關於景深攝影機的技術背景與實作指南，以便後續結合 ROS2 機器人作業系統的各種進階功能。

- **第 5 章 ROS2 機器人作業系統**：說明如何在 NVIDIA Jetson Orin Nano 上使用 ROS2 機器人作業系統搭配 NVIDIA Isaac ROS 套件實現多樣化的機器人應用，從基礎功能到進階技術全面解析。內容涵蓋 ROS2 的特性與改進（如即時性與多平台支援），SLAM 定位與地圖建置以及 ROS2 節點、導航、建圖與影像串流等功能實作。進階應用包括物體辨識、路徑規劃、影像分割與景深攝影機的 ArUco 標記辨識等。本章介紹了 ROS2 與 AI 技術結合後的全新面貌，讓您的機器人功能更加全面。

- **第 6 章 生成式 AI 結合邊緣運算裝置**：作為全書壓軸，聚焦生成式 AI 在 NVIDIA Jetson 邊緣運算平台上的應用，涵蓋文字生成、圖像生成、多模態技術與聲音處理等創新案例，展現其在智慧監控、機器人技術與個性化內容創作中的潛力。內容包括生成式 AI 的基礎概念與多領域應用，再帶入 Jetson AI Lab 提供的範例（如文字生成、圖像生成、多模態整合與聲音處理）與進階應用（如 RAG 技術在工業、醫療與交通領域的實踐）。本章詳細說明了在邊緣裝置端執行生成式 AI 的創新可能，為智慧應用開啟更多想像，也是全書技術與創意交融的完美句點。

材料表

- NVIDIA Jetson Orin Nano 開發者套件
- 19V 2.37A 變壓器
- 128GB Samsung micro SD 卡 / 500GB SSD 固態硬碟（本章使用後者）
- 外接螢幕（HDMI 接頭）
- DP（Display Port）轉 HDMI 訊號轉換器，DC202
- USB 鍵盤
- USB 滑鼠
- 乙太網路線（如果沒有無線網路的情況）
- 遠端連入 Jetson 的電腦（本篇文章使用 Windows 系統）
- 羅技 C270 webcam/ Pi camera V2

Section

2.1 Jetson Orin Nano 開機！

本節將說明如何準備開機用的 micro SD 記憶卡、Jetson Orin Nano 硬體介面說明以及如何設定 Jetson Orin Nano 網路來遠端登入。更多資料請參考 Jetson Orin Nano 主頁面[1]。

注意！

NVIDIA 原廠建議使用 SSD 固態硬碟會有最佳效能，本書兩種方式都會說明，請根據您的預算需求來選擇要用哪一種開機方式吧！

1 註解內容請見本書 github（https://github.com/cavedunissin/edgeai_jetson_orin）。以下註解皆是。

2.1.4 硬體架設與開機設定

接著來認識一下 Jetson Orin Nano 各個接頭用途，請看下圖。

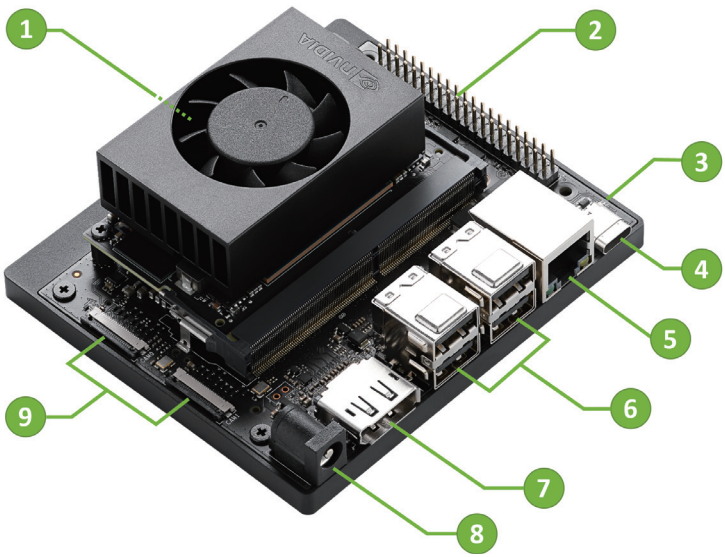

圖 2-20 Jetson Orin Nano 各接頭介紹，來源：NVIDIA 官方頁面 [1]

1. Micro SD 卡插槽
2. 40pin GPIO 排座
3. 電源指示 LED
4. USB-C 傳輸埠，只用於資料傳輸，不用於供電
5. Gigabit 乙太網路接頭
6. USB 3.1 type-A 接頭（4 個）
7. DisplayPort 接頭
8. 19V 直流電源輸入
9. MIPI CSI 攝影機接頭

請根據以下步驟讓 Jetson Orin Nano 開機吧！

Step 25 **[如使用 SSD 請跳過本步驟]** 請把 SD 卡插入 Jetson Orin Nano 底下的 SD 卡插槽，底部有一個彈簧卡榫，推到底會卡住，再按一次就會反向推出 SD 卡方便拿出來。

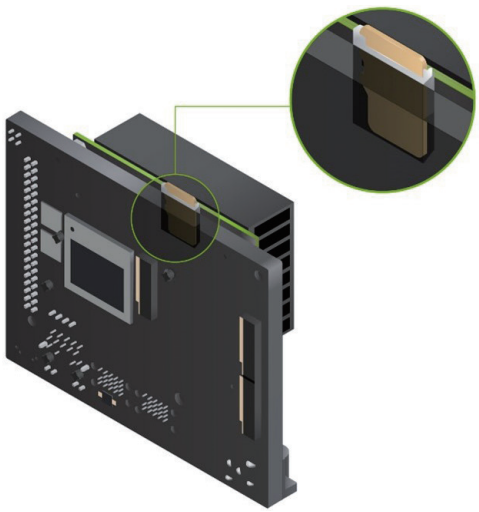

圖 2-21 插入已燒錄好的 micro SD 卡，來源：NVIDIA 官方頁面 [1]

Step 26 接下來需要接上電源，Jetson Orin Nano 需使用合乎原廠規格的直流電源，否則可能無法正常運作或損壞。詳細配件請參考 NVIDIA 原廠說明 [1] 以及機器人王國整理的 Jetson Orin Nano 套件包 [7]。如果您是要將 Jetson 當作一般 PC 來使用，請在接上電源之前，先將其他外接硬體（螢幕、鍵盤、滑鼠）接好之後再連接電源，這樣是最安全的建議作法。鍵盤滑鼠有很多選擇，可以買共用同一個 USB 發射器的鍵鼠組，可以節省一個寶貴的 USB 接頭喔！

圖 2-22 原廠 DC 電源供應器

使用原廠的 19V 2.37A 變壓器電源線接上電源之後，如果開發板上亮起綠燈就代表 Jetson Orin Nano 已經啟動囉！

Step 27 順利開機之後就會看到 Ubuntu 的登入畫面了。

圖 2-23 將 Jetson Orin Nano 作為桌上型電腦來開機

Step 28 請依序設定好以下內容：

同意條款→選擇語言→選擇鍵盤排列方式→選擇時區→設定帳號及密碼→ App Partition Size。這些設定後續都可再次進入 Ubuntu 的系統設定中來修改。

Step 29 設定完成，再稍等一下就可以看到 Ubuntu 桌面環境，如要做其他設定可到 Ubuntu 官方網站[9] 或搜尋相關資源。常見操作如上網或文書處理應該是與 Windows 相當類似，但本書多數操作都是在終端機或 Jupyter Lab 中完成。

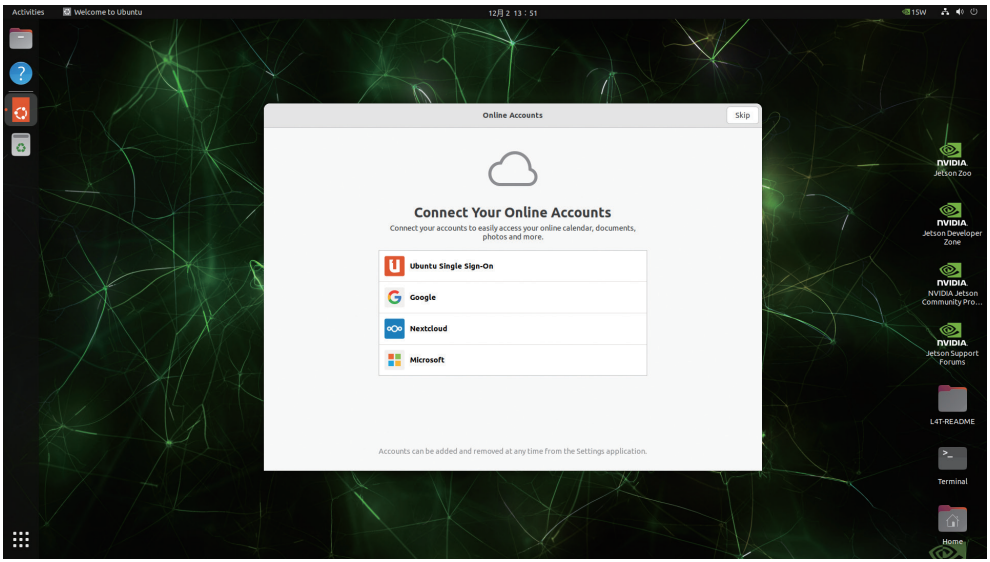

圖 2-24 Jetson Orin Nano 開機後的桌面環境

桌面上也有兩個 NVIDIA 的捷徑：Nvidia Jetson Developer Zone[10]、NVIDIA 開發者論壇[11]，點選後會連結到官方頁面和論壇，上面有許多關於 Jetson Orin Nano 的資料。

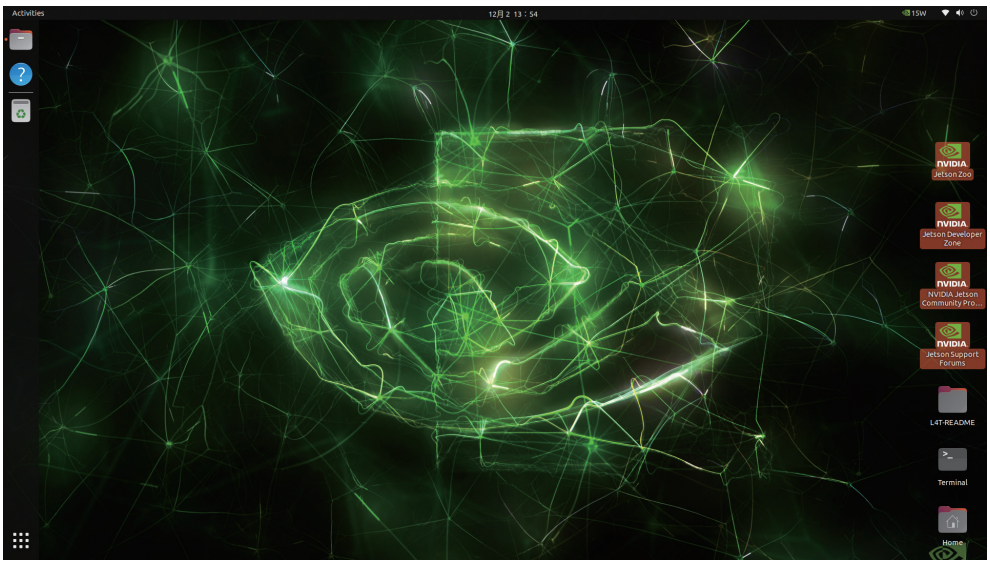

圖 2-25 桌面上的資源捷徑

· CHAPTER ·

04

整合深度視覺

隨著邊緣運算的快速發展，NVIDIA Jetson 平台以其卓越的運算能力與能效比成為 AI 與深度學習應用的核心選擇。而談到了所謂的智能系統，結合景深視覺攝影機更是賦予了裝置「理解世界」的能力。透過這些高精度的深度感測器，Jetson 平台不僅能實現三維空間建模與物體追蹤，還能在自主導航、機器人控制、手勢辨識等應用中大放異彩。

本章將深入探討如何讓 Jetson Orin Nano 整合 Intel RealSense 和 ZED 兩款主流的深度視覺攝影機，從硬體接入到軟體開發，並展示其在實際應用中的強大潛力。透過這些技術，我們將看到 AI 系統如何從平面視角跨越到三維世界，為智慧邊緣裝置帶來更多創新可能性。

所需硬體：

1. NVIDIA Jetson Orin Nano 開發者套件
2. Intel RealSense D435 景深攝影機
3. ZED2 景深攝影機

Section

4.1 Intel RealSense 景深攝影機

Intel RealSense 景深攝影機[1] 採用立體影像感測技術，使裝置能藉由立體視覺來理解周遭的環境，進而與環境互動。Intel RealSense 景深攝影機可在各種光照條件下於室內與室外運作，也可在多種攝影機配置中使用而無需額外校正。分為多條產品線：景深、光達、臉部辨識與追蹤等，本章將介紹 D435 景深攝影機，另外同系列相同規格的還有 D435i，差別在於後者多了 IMU。其餘規格請參考原廠介紹。

Intel D435[2] 擁有左右雙鏡頭，搭配紅外線感測器 (IR Sensor)，可以使用點雲格式描繪攝影機前物體的 3D 座標資料，藉此進行 3D 掃描等應用。本節將介紹 Intel RealSense D435 的安裝方式以及與 Jetson Orin Nano 結合之後的應用。

圖 4-1 Intel RealSense D435 景深攝影機

1 註解內容請見本書 github（https://github.com/cavedunissin/edgeai_jetson_orin）。以下註解皆是。

4.1.1 在 Jetson Orin Nano 上安裝 RealSense 套件

在此先說明如何在 Jetson Orin Nano 上安裝 RealSense 套件，安裝流程參考 JetsonHacks 的教學[3]，安裝時間約 60 分鐘。

Step 01 請用以下指令取得 RealSense SDK。這個 SDK 是以 Intel RealSense 原廠的 SDK[4] 做修改，支援 Intel RealSense 的 D400 系列、T265、SR300 等型號。

```
git clone https://github.com/jetsonhacksnano/installLibrealsense
```

Step 02 請用以下指令來安裝所需套件。安裝時可能要輸入使用者密碼。

```
cd installLibrealsense
./installLibrealsense.sh
```

Step 03 執行以下指令來建置所有套件，安裝時間需要大約一小時，泡杯咖啡耐心等候吧！注意：建置過程使用 `libuvc`，因此不必重新建置 kernel。

```
./buildLibrealsense.sh
```

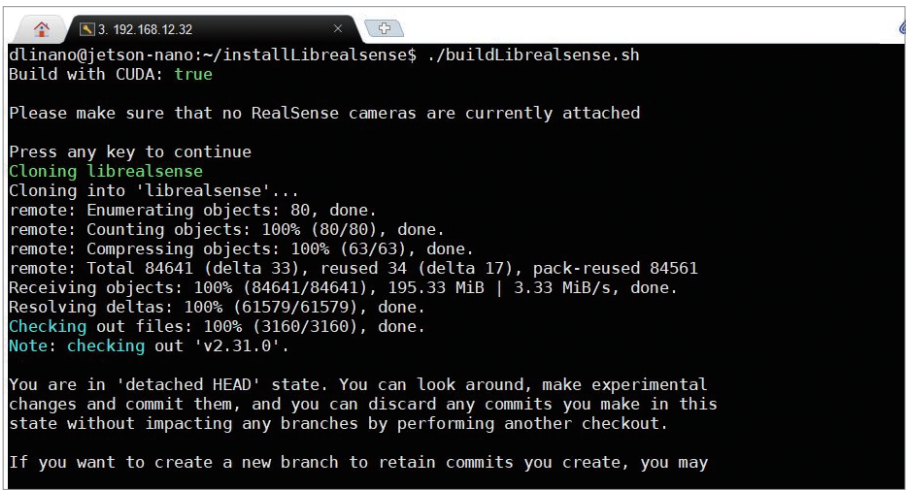

圖 4-2 Librealsense 建置過程畫面

Step 04 安裝必要的相依套件

```
sudo apt-get install libcanberra-gtk-module libcanberra-gtk3-module
```

Step 05 安裝成功後重新開機讓相關設定生效

```
sudo reboot
```

4.1.2 在 RealSense Viewer 中檢視深度影像

RealSense Viewer[5] 是 RealSense SDK 中一個很實用的小程式，它可以幫助使用者在開發程式之前快速確認以下資訊：

- RealSense 裝置型號
- 裝置與 RealSense 的 USB 版本
- 確認 RGB、深度、IR 影像
- 確認 RGB 與深度的影像整合畫面
- 設定輸出像素、FPS、ROI 與簡易的濾波

請在終端機中輸入以下指令來啟動 RealSense 操作介面：

```
realsense-viewer
```

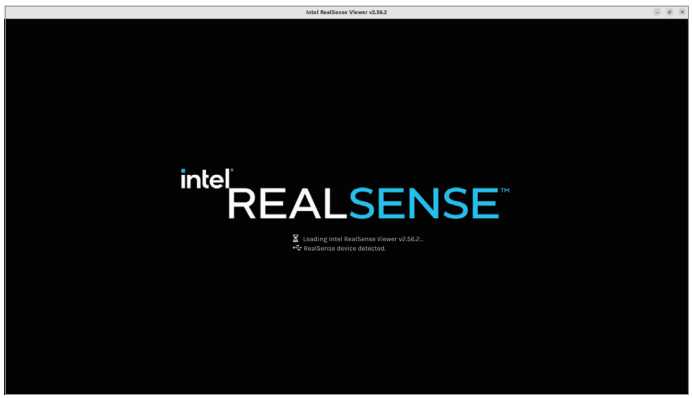

圖 4-3 RealSenser Viewer 初始畫面

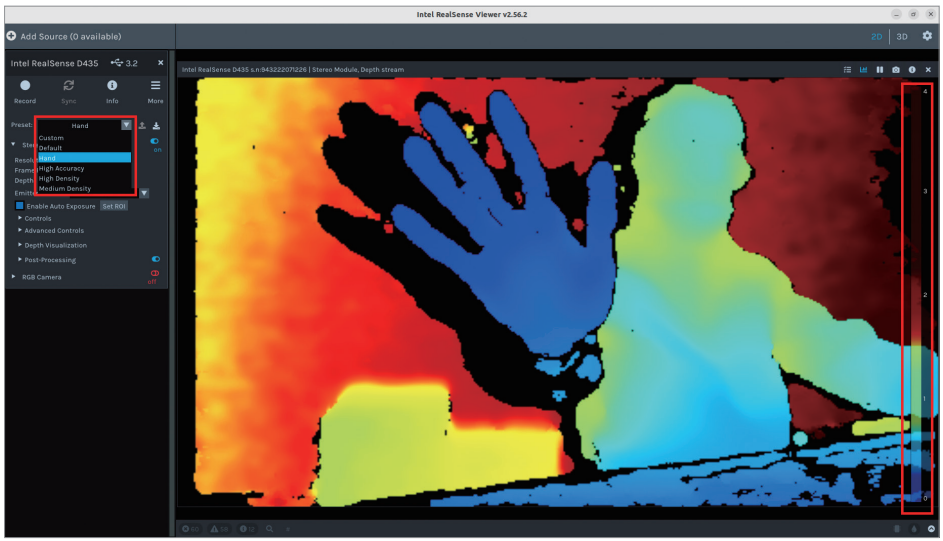

圖 4-5 檢視深度畫面

Stereo Module 下拉式選單有更多細節設定，包含輸出畫面的像素、FPS 以及開啟紅外線畫面。如下圖，使用者可以更細緻去確認深度、IR、RGB 影像以及 RealSense 同時輸出的狀態。每個影像視窗上的選單可以個別暫停畫面輸入、拍照與查看設定參數等等。

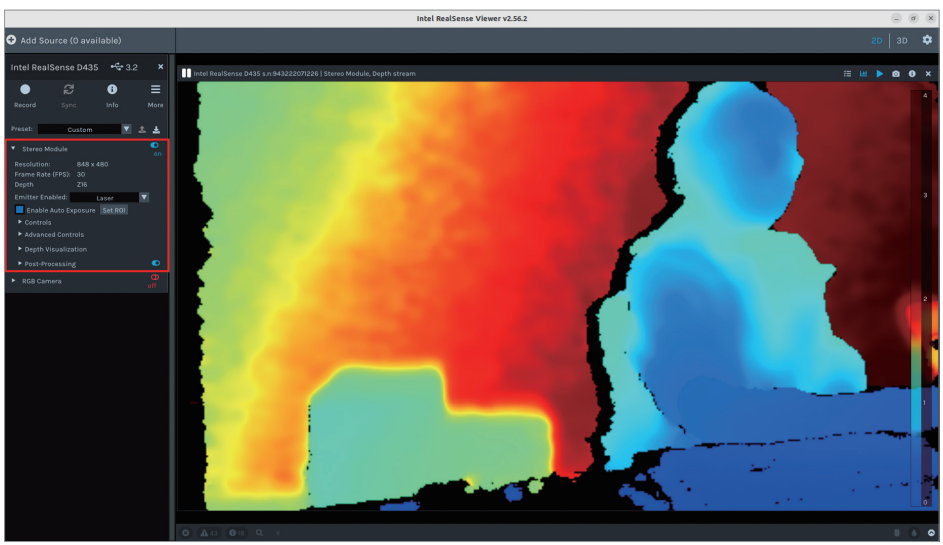

圖 4-6 **Stereo Module** 選單下的設定選項

4.1.3 RealSense 的 Python 範例

Intel 原廠針對 RealSense 系列裝置提供了 C 與 Python 等程式語言範例[6]，本書皆以 Python 來說明，C 語言範例位於 `librealsense/example` 路徑下。以下依序說明幾個重要的 Python 範例：

在終端機顯示深度資訊

本範例屬於輕量型的測試程式，請用以下指令來執行本範例：

```
cd ~/librealsense/wrappers/python/example
python3 python-tutorial-1-depth.py
```

執行後即可在終端機中看到以不同符號所呈現的深度影像，如以下畫面是一個人伸出一隻手（自行想像囉）：

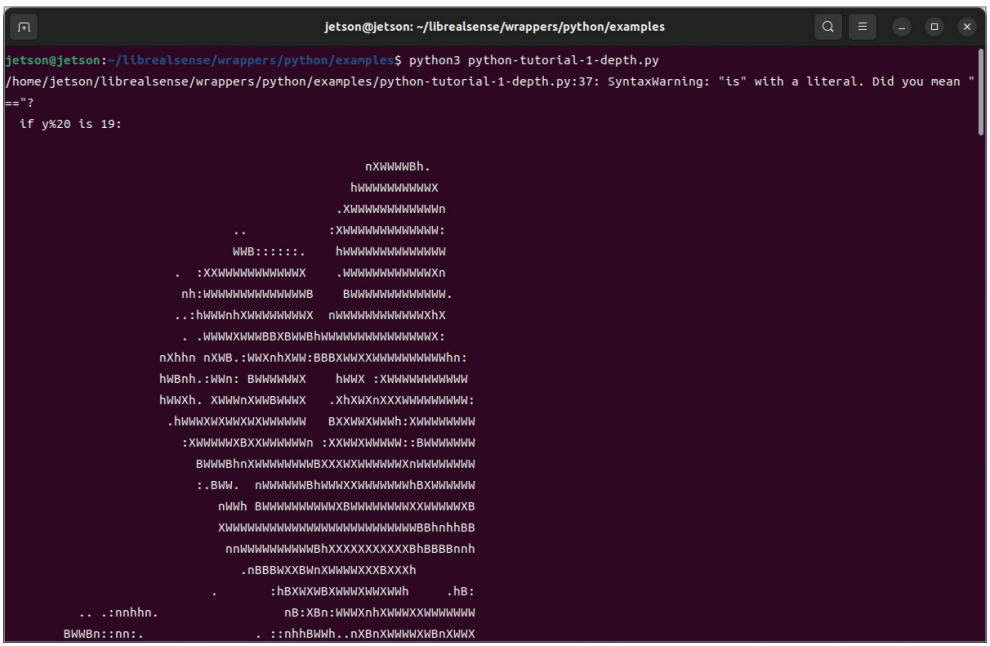

圖 4-10 python-tutorial-1-depth.py 執行畫面

python-tutorial-1-depth.py 程式碼完整內容如下：

```
## License: Apache 2.0. See LICENSE file in root directory.
## Copyright(c) 2015-2017 Intel Corporation. All Rights Reserved.

###############################################################
## librealsense tutorial #1 - Accessing depth data ##
###############################################################

# First import the library
import pyrealsense2 as rs

try:
    # Create a context object. This object owns the handles to all
connected realsense devices
    pipeline = rs.pipeline()
    pipeline.start()

    while True:
        # This call waits until a new coherent set of frames is available
on a device
        # Calls to get_frame_data(...) and get_frame_timestamp(...) on a
device will return stable values until wait_for_frames(...) is called
        frames = pipeline.wait_for_frames()
        depth = frames.get_depth_frame()
        if not depth: continue

        # Print a simple text-based representation of the image, by
breaking it into 10x20 pixel regions and approximating the coverage of
pixels within one meter
        coverage = [0]*64
        for y in range(480):
            for x in range(640):
                dist = depth.get_distance(x, y)
                if 0 < dist and dist < 1:
                    coverage[x//10] += 1

            if y%20 is 19:
```

```
                line = ""
                for c in coverage:
                    line += " .:nhBXWW"[c//25]
                coverage = [0]*64
                print(line)
    exit(0)
#except rs.error as e:
#    # Method calls agaisnt librealsense objects may throw exceptions of
type pylibrs.error
#    print("pylibrs.error was thrown when calling %s(%s):\n", % (e.get_
failed_function(), e.get_failed_args()))
#    print("    %s\n", e.what())
#    exit(1)
except Exception as e:
    print(e)
    pass
```

即時深度資訊與 RGB 影像串流對齊

本範例首先把深度影像與 RGB 影像對齊，之後再刪除距離較遠的部分，只留下較近的 RGB 影像，藉此做到前景後景分離的效果。

請用以下指令來執行本範例：

```
cd ~/librealsense/wrappers/python/example
python3 align-depth2color.py
```

執行畫面如下，可以看到一定距離之外的東西都被刪除了，這個距離閾值可在程式碼中修改，預設為 1 公尺：

```
clipping_distance_in_meters = 1 #1 meter
```

以下三張圖分別為 0.5、1 與 1.5 公尺的偵測結果：

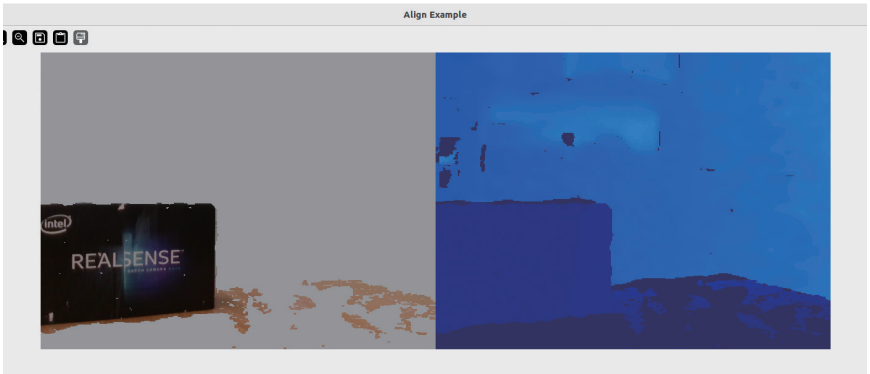

圖 4-11a 0.5 公尺偵測結果

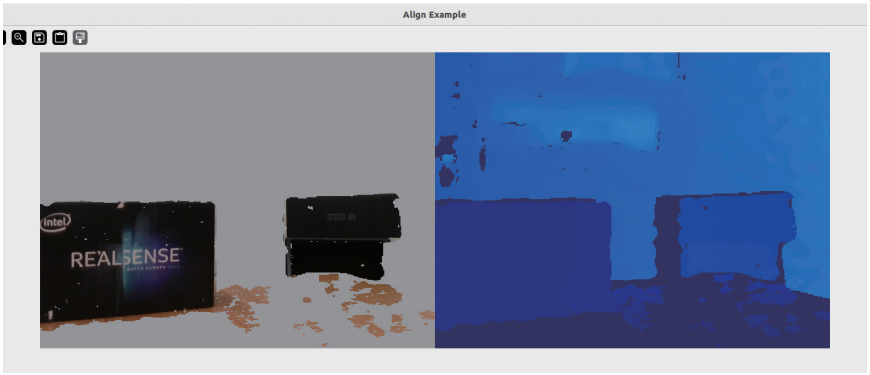

圖 4-11b 1 公尺偵測結果

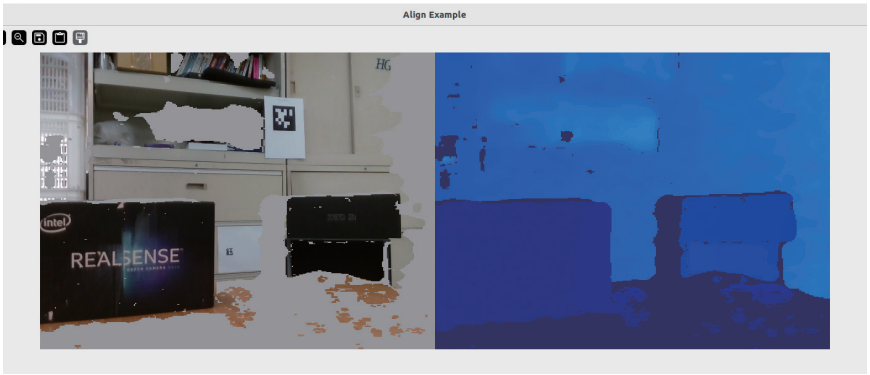

圖 4-11c 1.5 公尺偵測結果

到目前為止，我們已可在畫面上即時呈現深度距離訊息，也在畫面正中間加了黃色標記點幫助您確認量測點。請回顧第 2 章的 cv2.putRectangle 或 cv2.line 語法在畫面上做各種標註或格線。本範例可以延伸出很多與距離偵測相關的應用，例如室內空間丈量、偵測物品是否擺放整齊或是搭配大型螢幕做成互動遊戲裝置等等，甚至連設計與服裝領域都是這類小型距離量測裝置的絕佳舞台呢。

人臉辨識並取得臉部距離

想要做人臉辨識的話，只要稍微修改前一個範例就能使其改為偵測人臉，並在人臉的方框左上方顯示人臉距離，是不是愈來愈厲害了呢？本範例是修改原廠範例 opencv_singlepoint_viewer_example.py 而來，請試著根據以下步驟親自做一遍吧！完整程式碼 opencv_facedistance_viewer_example.py 請由本書 GitHub 取得，或自行下載完整的 Haar 分類器檔案來測試更多效果，後續步驟會介紹。

加入的第一行為 Haar 分類器的檔案路徑，在此使用 haarcascade_frontalface_default.xml。請根據您的擺放位置來修改以下的路徑。有興趣的人可以玩看看 /haarcascades 資料夾下的其他偵測器 [11]。

> 注意！
>
> Haar 分類器可以偵測畫面中的人臉，但無法分辨兩張臉的區別。如果要分辨臉孔的話，當然就需要訓練神經網路來進行推論喔！

```
face_cascade = cv2.CascadeClassifier('/home/your_user_name/opencv/data/
haarcascades/haarcascade_frontalface_default.xml')
```

下一步將圖像轉為灰階，方便後續偵測：

```
gray = cv2.cvtColor(color_image, cv2.COLOR_BGR2GRAY)
```

設定人臉偵測的參數，在此設定人臉偵測的最小尺寸為 50x50 像素，低於此大小則忽略不會將其視為人臉，詳細參數請參考 OpenCV 相關文件[12]：

```
faces = face_cascade.detectMultiScale(gray, scaleFactor=1.2,
    minNeighbors=5, minSize=(50,50))
```

每一張人臉都會用方框框起來並標記深度距離。本範例以人臉方框的正中央來代表人臉與鏡頭的距離。顯示深度的字串位置如果直接設為 `(x, y)` 會跟方框重疊在一起，所以將 `y` 修改為 `y-5` 讓文字略高於方框。任何顏色、粗細、字型、字體等喜好都可以自行做調整。

```
for (x, y, w, h) in faces:
    cv2.rectangle(color_image, (x, y), (x+w, y+h), (255, 0, 0), 2)
    text_depth = "depth is "+str(np.round(depth_frame.get_
distance(int(x+(1/2)*w), int(y+(1/2)*h)),3))+"m"
    color_image = cv2.putText (color_image, text_depth,(x, y-5), cv2.FONT_
HERSHEY_PLAIN,1,(0,0,255),1,cv2.LINE_AA)
```

輸入以下指令來開啟檔案並貼入上述程式碼。

```
nano opencv_facedistance_viewer_example.py
```

執行本範例之前，首先要下載 OpenCV 的 Haar 人臉分類器檔案[11]，請回到 `/home` 目錄，並輸入以下指令來下載 OpenCV 資料集。

```
cd ~
git clone https://github.com/opencv/opencv.git
```

下載完之後移動回範例資料夾，並執行以下指令來編輯檔案，並貼上剛剛的程式碼。

```
cd ~/librealsense/wrappers/python/examples
nano opencv_viewer_example.py
```

完成之後將其另存為 opencv_facedistance_viewer_example.py，完整程式碼請由本書 GitHub 取得。最後請用以下指令來執行本範例：

```
python3 opencv_facedistance_viewer_example.py
```

執行成果如下圖，可以順利偵測到多張人臉了，並可看到臉部與鏡頭的距離。由於臉部辨識效果完全仰賴 Haar 分類器，因此只有某些特定角度比較容易偵測得到，您可以在鏡頭前轉動頭部來看看偵測的極限。

單人版本：

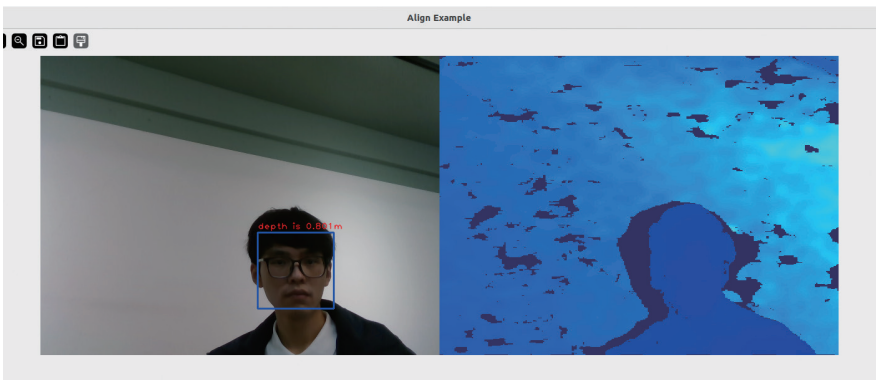

圖 4-14 單人偵測畫面

多人版本：

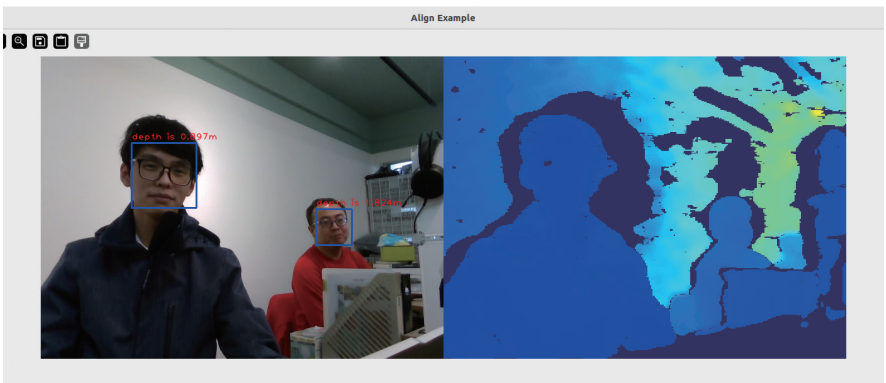

圖 4-15 多人偵測畫面