Thermal Camera Installation

Koushik K
Master's Thesis

December 2025

1 Hardware Required

Laptop with Windows, PureThermal 2 Camera as in Fig 1 and USB-B cable.

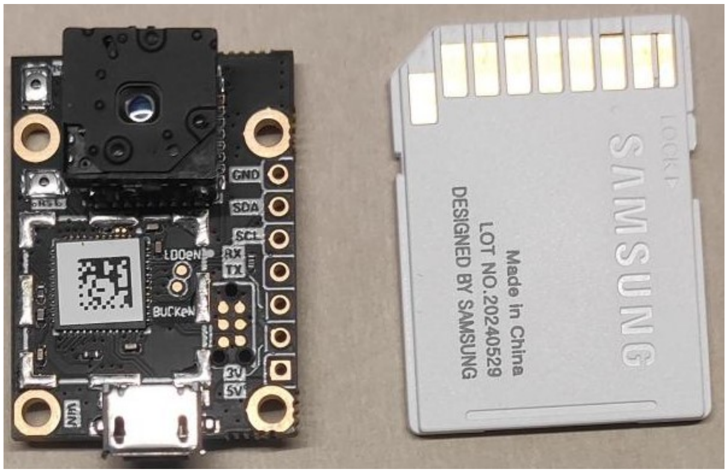

Figure 1: PureThermal 2 camera

2 Steps to Install

1. Install Python from: `https://www.python.org/downloads/`
2. Open Command Prompt and run:

   ```
   python --version
   ```

   If it prints the Python version, proceed. Otherwise continue to the next step.
3. Press `Win + R`, type `sysdm.cpl`, and press Enter.
4. Go to **Advanced** → **Environment Variables**.
5. Under **System variables**, locate **Path** → click **Edit**. Click **New** and add:

   ```
   C:\Users\acer\AppData\Local\Programs\Python\Python313\
   C:\Users\acer\AppData\Local\Programs\Python\Python313\Scripts\
   ```

6. Click OK everywhere to save and close all dialogs.
7. Restart the system, open Command Prompt, and run again:

   ```
   python --version
   ```

8. In Command Prompt, install OpenCV:

   ```
   pip install opencv-contrib-python
   ```

9. Create a workspace folder `D:\thermal`. In Command Prompt type:

   ```
   notepad code.py
   ```

   and paste script from AppendixA.

Code is modular to add any number of thermal cameras, just add indices for the required number.

3 Deploying the Setup

Connect the laptop and camera using USB cable and run the script in the same workspace.

```
python code.py
```

Upon running the script, a folder with a timestamp as its name will be created, and images will be stored in it. Captured images are saved with a timestamp as shown in Fig 2. In Fig 2 two thermal cameras were concatenated.

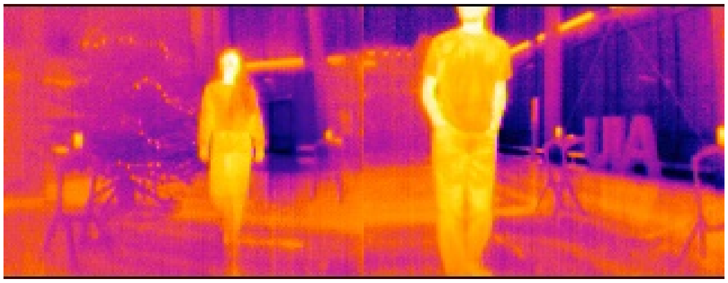

Figure 2: Output of the camera with name as ”20251127_222743_501947.jpg”

4 Appendix

A Python Script for Thermal Frame Capture

```
import cv2
import numpy as np
import threading
import time
import os
from datetime import datetime

# === CONFIGURE CAMERA INDICES AND RESOLUTION HERE ===
# List all camera indices you want to use. Index 0 is excluded as requested.
CAM_INDICES = [1,2] # Example: use cameras at index 1 and 2. Add more like [1, 2, 3]
FRAME_WIDTH = 160
FRAME_HEIGHT = 122
TARGET_FPS = 11 # The desired frames per second

# === Global variables for thread communication ===
stop_threads = False
# A dictionary to hold the latest frame from each camera, protected by a lock
frames = {}
frames_lock = threading.Lock()
```

```
def camera_capture_thread(cam_index, width, height):
    """
    Thread function to capture frames from a single camera.
    It reads frames continuously and updates a global dictionary.
    """
    global frames, stop_threads

    cap = cv2.VideoCapture(cam_index)
    if not cap.isOpened():
        print(f"[ERROR] Camera {cam_index} could not be opened.")
        return

    cap.set(cv2.CAP_PROP_FRAME_WIDTH, width)
    cap.set(cv2.CAP_PROP_FRAME_HEIGHT, height)

    actual_width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
    actual_height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
    print(f"[SUCCESS] Camera {cam_index} started. (Actual Res: {actual_width}x{actual_height})" )

    while not stop_threads:
        ret, frame = cap.read()
        if ret:
            # Rotate the frame 180 degrees
            frame = cv2.rotate(frame, cv2.ROTATE_180)

            # Safely update the global frames dictionary with the latest frame
            with frames_lock:
                frames[cam_index] = frame
        else:
            print(f"[WARN] Camera {cam_index}: Skipped a failed frame grab.")
            # Give the camera a moment to recover
            time.sleep(0.1)

    print(f"[INFO] Releasing camera {cam_index}.")
    cap.release()


def main():
    """
    Main function to start camera threads and display the combined video feed.
    """
    global stop_threads

    # Start a capture thread for each camera
    threads = []
    for index in CAM_INDICES:
        thread = threading.Thread(target=camera_capture_thread, args=(index, FRAME_WIDTH, FRAME_HEIGHT))
        threads.append(thread)
        thread.start()

    print(f"\n[INFO] All camera threads started. Press ’q’ in the preview window to stop all.")

    # Create a unique folder for this run
    folder_name = "./captures/" + datetime.now().strftime("%Y%m%d_%H%M%S_%f")
    os.makedirs(folder_name, exist_ok=True)
    print(f"[INFO] Saving combined frames to ’{folder_name}/’ folder.")

    # Create a placeholder for cameras that haven’t sent a frame yet
    placeholder_frame = np.zeros((FRAME_HEIGHT, FRAME_WIDTH, 3), dtype=np.uint8)
    cv2.putText(placeholder_frame, ’NO SIGNAL’, (20, FRAME_HEIGHT // 2), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 1)

    # === FPS CALCULATION VARIABLES ===
    frame_counter = 0
    start_time = time.time()
    fps = 0
    target_frame_time = 1.0 / TARGET_FPS
    # =================================

    # Main loop to display and save the concatenated frames
    while True:
        loop_start_time = time.time() # Time at the start of the loop

        frames_to_show = []
        with frames_lock:
            # Collect frames in the correct order, using a placeholder if a frame is missing
            for index in CAM_INDICES:
                frames_to_show.append(frames.get(index, placeholder_frame))

        # Only proceed if we have frames to show
        if frames_to_show:
            # Concatenate frames horizontally
            combined_frame = cv2.hconcat(frames_to_show)

            # --- FPS Calculation ---
            frame_counter += 1
            # Check if one second has passed
            elapsed_time = time.time() - start_time
            if elapsed_time > 1.0:
                # Calculate FPS
                fps = frame_counter / elapsed_time
                print(f"FPS: {fps:.2f}") # Print FPS to console
                # Reset for the next second
                frame_counter = 0
                start_time = time.time()

            cv2.imshow("Multi-Camera Live Feed", combined_frame)

            # Save the combined frame with a unique timestamp
            timestamp = datetime.now().strftime("%Y%m%d_%H%M%S_%f")
            filename = os.path.join(folder_name, f"{timestamp}.jpg")
            cv2.imwrite(filename, combined_frame)

        # === FPS LIMITER ===
        # Calculate how long the loop took and sleep for the remaining time
        time_to_wait = target_frame_time - (time.time() - loop_start_time)
        if time_to_wait > 0:
            time.sleep(time_to_wait)
        # ===================

        # Check for ’q’ key press to exit
        if cv2.waitKey(1) & 0xFF == ord(’q’):
            stop_threads = True
            break

    # Wait for all camera threads to finish cleanly
    for thread in threads:
        thread.join()

    print("[INFO] All cameras stopped. Exiting.")
    cv2.destroyAllWindows()


if __name__ == "__main__":
    main()
```