



Informe I - Proyecto DetPax

27/03/2020

- Alumno:
- Paulo Morales Aguayo

- Profesores:
- José Delpiano
- Sebastián Seriani



Resumen Ejecutivo

El denominado proyecto DetPax es un dispositivo que se compone principalmente por una Raspberry Pi 3B, un acelerador ML (Google Edge TPU coprocessor), una cámara, un teclado inalámbrico y una batería de 5V.

La función del proyecto se basa en el análisis y detección en tiempo real de flujos de personas en distintos medios de transporte públicos, tales como Metro de Santiago o TrenCentral Grupo EFE.

En lo que sigue, se plantearán los desarrollos que se ejecutaron durante la mejora del dispositivo; sus alcances y limitaciones; y, finalmente los logros obtenidos al finalizar la segunda entrega del proyecto.

En conjunto al informe, se adjunta la carpeta “Avances_DetPax” con experimentos detallados.



1. Objetivos

Objetivo general: Lograr la detección de personas y sus flujos en respectivos medios de transporte, bajo distintas condiciones ambientales, ya sean por cambios de luz o sobrepoblación de las plataformas.

Objetivos Específicos:

- Probar y comparar DetPax bajo distintos parámetros y gadgets, tales como distintas cámaras.
- Comparar el uso de CPU y elementos alternos/externos al dispositivo original.
- Realizar pruebas con DetPax en una estación de metro existente, o en un video con flujos de personas.

2. Metodología Experimental

El trabajo se basó en la previa utilización, y posterior reciclaje, del código implementado en la etapa primeriza del proyecto¹.

Habiendo aclarado el punto de partida, se procederá a detallar los elementos que tuvieron un papel preponderante tanto en la primera, como en la segunda parte; y el flujo, en cuanto a mejorar el dispositivo, para los casos respectivos. Se detallan a continuación.

2.1. Estructura

La comparación de estructuras de códigos, de la primera etapa a la segunda (actual), se muestra a continuación.

```
.
|--- pyimagesearch
|   |--- __init__.py
|   |--- centroidtracker.py
|   |--- trackeableobject.py
|--- mobilenet_ssd
|   |--- MobileNetSSD_deploy.caffemodel
|   |--- MobileNetSSD_deploy.prototxt
|--- output
|   |--- output_01.avi
|--- people_counter.py

↓

.
|--- pyimagesearch
|   |--- __init__.py
|   |--- centroidtracker.py
|   |--- trackeableobject.py
|-----| CLAHE
|--- Coral Edge TPU
|   |--- mobilenet_ssd_v2_coco_quant_postprocess_edgetpu.tflite
|   |--- coco_labels.txt
|--- output
|   |--- output_01.avi
|--- detect_video.py
```

Fig. 1: Primera etapa y segunda etapa de códigos, respectivamente.

¹ <https://www.pyimagesearch.com/2018/08/13/opencv-people-counter/>



El código, originalmente, se encarga en la detección de personas, y realizando un conteo de ellas dependiendo de la dirección en que se movían. Es decir, se contaban a los miembros que cruzaran hacia arriba o hacia abajo, usando como referencia una línea amarilla horizontal en el centro del frame.

El conteo y seguimiento son facilitados gracias a los módulos de pyimagesearch, centroidtracker.py y trackeableobject.py, los cuáles realizan cálculos y proyecciones de ubicación.

Si bien la arquitectura cambiada de la etapa 1 a la 2 no difieren del todo, se debe a que en dicha estructura la mayor desventaja era representada por el procesamiento y manejo de información, y por la falta de detallismo y afinamiento del código.

2.2 MobileNet_SSD => Coral Edge TPU

2.2.1 MobileNet_SSD

MobileNet es una simple arquitectura, de bajo consumo de recursos, que consiste de una convolución en profundidad (depthwise) de 3x3, seguido de una convolución puntiaguda (pointwise) de 1x1.

SSD, Single-shot-detector, se caracteriza por tomar una sola fotografía para detectar múltiples objetos dentro de una imagen, que para nuestro modelo se restringe sólo a identificar personas.

Existen 2 modelos de SSD:

1. SSD300: En este modelo, la imagen de entrada posee un tamaño de 300x300. Es usado en imágenes con baja resolución, de rápido procesamiento y es menos preciso que el SSD512.
2. SSD512: Este modelo tiene una imagen de entrada de 500x500. Es usado en imágenes de resoluciones más altas y es más preciso que otros modelos.

El código original maneja estas estructuras y datos con Caffe².

Caffe es un Open framework, poseyendo así modelos y ejemplos para Deep learning; también, su procesamiento puede variar entre CPU y GPU.

Habiendo aclarado los conceptos relacionados a inferencias de la primera parte del proyecto, queda por asegurar que el procesamiento de este bloque cae en completa responsabilidad del hardware de Raspberry Pi para el manejo y procesamiento de datos.

² <https://caffe.berkeleyvision.org/>

2.2.2 Coral Edge TPU

El acelerador USB 3.0 de Google es un dispositivo que posee una Edge Tensor Processing Unit (TPU) como un co-procesador para el computador. Se encarga de acelerar las inferencias para los respectivos modelos de machine learning o Deep learning, con un muy bajo consumo de energía.

El uso de este complemento debe ser a través de la librería `edgetpu_api`³, la cual, al momento de la instalación, ofrece 2 modalidades: (1) Frecuencia normal, o (2) Frecuencia máxima, habiendo elegido la primera.

La librería entrega las herramientas para llamar al modelo `tflite` y sus labels respectivos al procesamiento.

Con esto, la implementación requería cambios en el código base.

Cabe notar que el uso de Coral permite que la Raspberry Pi se libere del manejo y procesamiento de las inferencias.



Fig. 2: USB 3.0 Coral

³ <https://gilberttanner.com/blog/google-coral-usb-accelerator-introduction>

2.3 Webcam LifeCam HD-3000 USB => Pi Camera V2

2.3.1 Webcam LifeCam HD-3000 USB

Características:

- Cuadros por segundo: 30 FPS
- Interfase: USB 2.0
- Resolución:
 - Grabación de video: 1280x720 pixeles
 - Captura de imagen: 1280x800 pixeles

Requerimientos del sistema:

- Transmisión con calidad VGA (640x480):
CPU Dual Core @ 1,6 GHz
1 GB de RAM
- Grabación con calidad HD (720p):
CPU Dual Core @ 3,0 GHz o más alto
2 GB de RAM
1,5 GB de espacio en el disco duro (dependiendo del tamaño del video, puede ser más)



Fig. 3: Webcam HD.

2.3.2 Pi Camera V2

El módulo de cámara Raspberry Pi v2 es un complemento adicional de alta calidad de 8 megapíxeles con sensor de imagen Sony IMX219 diseñado especialmente para

Raspberry Pi, con lente de enfoque fijo. Es capaz de imágenes estáticas de 3280 x 2464 píxeles, y también es compatible con 1080p30, 720p60 y

Video 640x480p60 / 90. Se conecta a Raspberry Pi por medio de uno de los pequeños sockets en la superficie superior de la placa y utiliza la interfaz dedicada CSI, diseñada especialmente para la interfaz con cámaras⁴.



Fig. 4: Pi Camera V2.

⁴ http://www.farnell.com/datasheets/2056179.pdf?_ga=1.152577328.880870297.1479740269

2.4 CLAHE

La ecualización de histograma adaptativo (AHE - Adaptive Histogram Equalization) es una técnica de procesamiento de imágenes utilizada para mejorar el contraste de las imágenes. Se diferencia de la ecualización de histograma ordinario en el sentido de que el método adaptativo calcula varios histogramas, cada uno correspondiente a una sección distinta de la imagen, y los usa para redistribuir los valores de claridad de la imagen. Por lo tanto, es adecuado para mejorar el contraste local y mejorar las definiciones de bordes en cada región de una imagen.



Fig. 5: Ejemplo de aplicación de CLAHE.

Sin embargo, el AHE ordinario tiende a sobre amplificar el contraste en regiones casi constantes de la imagen, ya que el histograma en tales regiones está altamente concentrado. Como resultado, AHE puede hacer que el ruido se amplifique en regiones casi constantes. Contrast Limited AHE (CLAHE) es una variante de ecualización de histograma adaptativo en la que la amplificación de contraste es limitada, para reducir este problema de amplificación de ruido.

En CLAHE, la amplificación de contraste en la vecindad de un valor de píxel dado viene dada por la pendiente de la función de transformación. Esto es proporcional a la pendiente de la función de distribución acumulativa de vecindad (CDF) y, por lo tanto, al valor del histograma en ese valor de píxel. CLAHE limita la amplificación recortando el histograma a un valor predefinido antes de calcular el CDF. Esto limita la pendiente del CDF y, por lo tanto, de la función de transformación. El valor al que se recorta el histograma, el llamado límite de clip, depende de la normalización del histograma y, por lo tanto, del tamaño de la región vecina.

Es ventajoso no descartar la parte del histograma que excede el límite del clip, sino redistribuirlo por igual entre todos los contenedores de histograma.

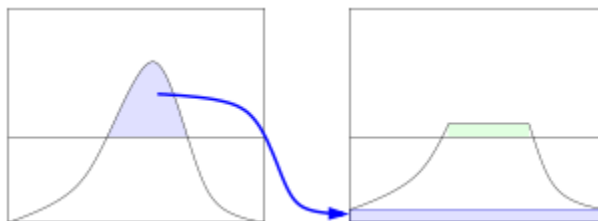


Fig. 6: Ejemplo de manejo de distribución de probabilidad.

2.5 Raspberry Pi 3 modelo b

La Raspberry Pi 3 modelo b es el primero de la 3ra generación de Raspberry Pi.

Sus especificaciones son las siguientes:

- Quad Core 1.2GHz Broadcom BCM2837 64bit CPU (ARM Cortex-A53)
- 1GB RAM
- BCM43438 wireless LAN y Bluetooth Low Energy (BLE)
- 100 Base Ethernet
- 40-pin GPIO
- 4 puertos USB 2.0
- 4 salidas estéreo de 4 polos y puerto de video compuesto
- HDMI de tamaño completo
- Puerto de CSI camera para conectar a Raspberry Pi camera
- Puerto de display DSI para conectar un Raspberry Pi touchscreen display
- Puerto Micro SD para cargar tus OS y guardar información
- Fuente de alimentación micro USB conmutada actualizada de hasta 2.5A



Fig. 7: Raspberry Pi 3b.

3. Resultados y Discusiones

A continuación, se expondrán los motivos relacionados a los cambios efectuados en el dispositivo, ya sea por gadgets o por códigos.

3.1 MobileNet_SSD => Coral Edge TPU

El cambio de MobileNet_SSD a Coral Edge TPU se debe principalmente al ahorro de recursos como CPU, y a la aceleración brindada al dispositivo.

Si bien ambos elementos requieren de modelos (.caffe en el caso de Caffe, como de .tflite en el de Coral), es decir, poseen una estructura bastante parecida; pero no cumplen este desempeño al momento de ejecutar el programa.

Esto se debe a que MobileNet_SSD trabaja directamente con el hardware de Raspberry Pi, mientras que Coral trabaja de manera externa, apoyándose en la CPU solo en el previo y posterior manejo de las imágenes.

Es posible ver un cuadro comparativo de ambos modelos:

| Modelo | Tiempo de Inferencias (segundos) | FPS | Sobrecalentamiento | Detección (%) |
|----------------|----------------------------------|------|--------------------|---------------|
| MobileNet_SSD | 0.53 | 2.96 | SI | 7.57 |
| Coral Edge TPU | 0.46 | 1.5 | NO | 200 |

Tabla 1: Comparación de MobileNet_SSD y Coral.

El porcentaje de detección se midió usando el mismo video (metro.mp4), hasta el minuto 1:03 (esto se debió a que el modelo de MobileNet_SSD no podía llegar más lejos a causa de un sobrecalentamiento, y, por ende, la finalización de su ejecución). En este video se contabilizaron “a mano” las personas aparecidas, que fueron 66 en total. Luego, se ejecutaron los distintos modelos; en el caso de Coral se efectuó con el algoritmo CLAHE. En ambos casos se usó un valor de confianza del 40%.

El porcentaje de MobileNet_SSD no requiere un mayor análisis, pero en el caso contrario, Coral sí lo necesita.

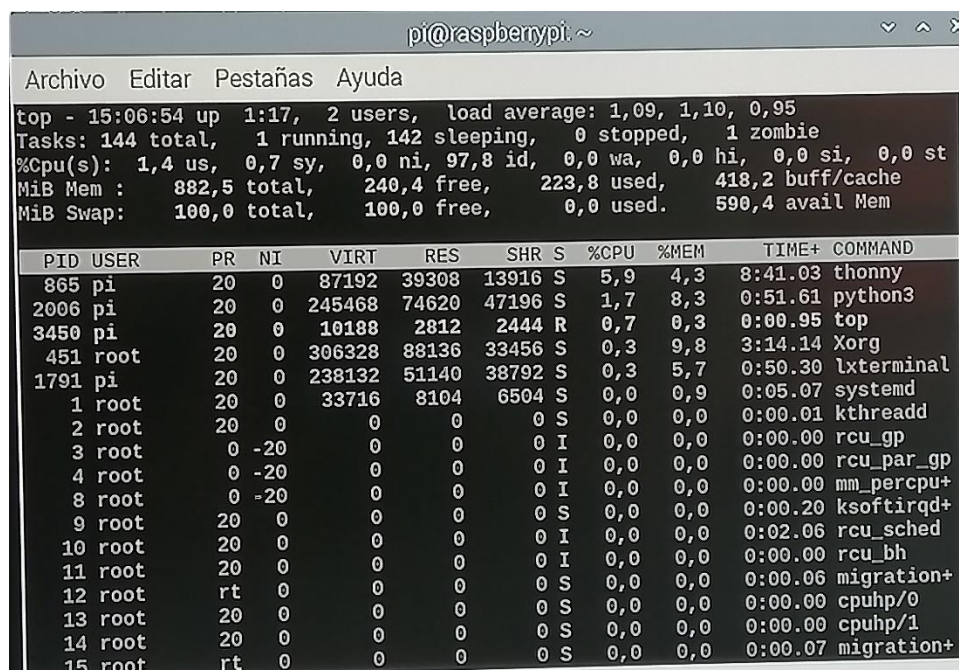
Es necesario notar que el video no presenta una cámara fija, o un ángulo privilegiado (que podría ser de 45°), ni tampoco una altura ideal (a veces estaba muy lejos), y menos una resolución ideal; por lo cual el índice de ruido o de memoria para recordar ciertas ubicaciones de detecciones previas entorpecen el análisis. He ahí la duplicación que presenta Coral.

Además, es preponderante notar el poco avance de análisis en el mismo video. En el caso de Coral, basado en los videos anexados en la carpeta Avances_DetPax asociada al informe, llega a procesar hasta el minuto 2:03, pudiendo llegar más adelante, sin presencia de sobrecalentamiento del sistema.

En base a este detalle, realmente no existe una comparación “lineal” de ambos modelos frente al material propuesto.

Es más, en base al sobrecalentamiento, se decidió ejecutar el comando << top >> en el prompt de Raspberry Pi mientras se ejecutaban los programas, obteniendo la siguiente comparación en cuanto al uso de memoria y CPU:

- CPU Normal (sin uso de programas): 5.9 %



```

top - 15:06:54 up 1:17, 2 users, load average: 1.09, 1.10, 0.95
Tasks: 144 total, 1 running, 142 sleeping, 0 stopped, 1 zombie
%Cpu(s): 1.4 us, 0.7 sy, 0.0 ni, 97.8 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 882.5 total, 240.4 free, 223.8 used, 418.2 buff/cache
MiB Swap: 100.0 total, 100.0 free, 0.0 used. 590.4 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM    TIME+  COMMAND
 865  pi         20   0   87192   39308  13916 S   5.9   4.3   8:41.03 thonny
2006  pi         20   0  245468  74620  47196 S   1.7   8.3   0:51.61 python3
3450  pi         20   0   10188   2812   2444 R   0.7   0.3   0:00.95 top
 451  root       20   0  306328  88136  33456 S   0.3   9.8   3:14.14 Xorg
1791  pi         20   0  238132  51140  38792 S   0.3   5.7   0:50.30 lxterminal
    1  root       20   0   33716   8104   6504 S   0.0   0.9   0:05.07 systemd
    2  root       20   0         0         0         0 S   0.0   0.0   0:00.01 kthreadd
    3  root        0 -20         0         0         0 I   0.0   0.0   0:00.00 rcu_gp
    4  root        0 -20         0         0         0 I   0.0   0.0   0:00.00 rcu_par_gp
    8  root        0 -20         0         0         0 I   0.0   0.0   0:00.00 mm_percpu+
    9  root       20   0         0         0         0 S   0.0   0.0   0:00.20 ksoftirqd+
   10  root       20   0         0         0         0 I   0.0   0.0   0:02.06 rcu_sched
   11  root       20   0         0         0         0 I   0.0   0.0   0:00.00 rcu_bh
   12  root       rt    0         0         0         0 S   0.0   0.0   0:00.06 migration+
   13  root       20   0         0         0         0 S   0.0   0.0   0:00.00 cpuhp/0
   14  root       20   0         0         0         0 S   0.0   0.0   0:00.00 cpuhp/1
   15  root       rt    0         0         0         0 S   0.0   0.0   0:00.07 migration+
  
```

Fig. 8: CPU normal.

- CPU – Coral: 108.6%

pi@raspberrypi: ~

Archivo Editar Pestañas Ayuda

```
top - 15:05:08 up 1:16, 2 users, load average: 1,29, 1,08, 0,92
Tasks: 144 total, 3 running, 140 sleeping, 0 stopped, 1 zombie
%Cpu(s): 29,0 us, 1,5 sy, 0,0 ni, 69,6 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
MiB Mem : 882,5 total, 109,9 free, 354,5 used, 418,1 buff/cache
MiB Swap: 100,0 total, 100,0 free, 0,0 used. 459,1 avail Mem
```

| PID | USER | PR | NI | VIRT | RES | SHR | S | %CPU | %MEM | TIME+ | COMMAND |
|------|------|----|-----|--------|--------|-------|---|-------|------|---------|------------|
| 3289 | pi | 20 | 0 | 464732 | 217944 | 79576 | R | 108,6 | 24,1 | 6:30.05 | python3 |
| 865 | pi | 20 | 0 | 87192 | 39308 | 13916 | S | 6,6 | 4,3 | 8:34.35 | thonny |
| 2006 | pi | 20 | 0 | 245468 | 74620 | 47196 | S | 1,7 | 8,3 | 0:49.80 | python3 |
| 451 | root | 20 | 0 | 307040 | 88744 | 34064 | R | 1,3 | 9,8 | 3:12.84 | Xorg |
| 1791 | pi | 20 | 0 | 238132 | 51132 | 38792 | S | 1,3 | 5,7 | 0:49.32 | lxterminal |
| 3450 | pi | 20 | 0 | 10188 | 2812 | 2444 | R | 0,7 | 0,3 | 0:00.25 | top |
| 328 | root | 20 | 0 | 27656 | 80 | 0 | S | 0,3 | 0,0 | 0:02.48 | rngd |
| 1 | root | 20 | 0 | 33716 | 8104 | 6504 | S | 0,0 | 0,9 | 0:05.06 | systemd |
| 2 | root | 20 | 0 | 0 | 0 | 0 | S | 0,0 | 0,0 | 0:00.01 | kthreadd |
| 3 | root | 0 | -20 | 0 | 0 | 0 | I | 0,0 | 0,0 | 0:00.00 | rcu_gp |
| 4 | root | 0 | -20 | 0 | 0 | 0 | I | 0,0 | 0,0 | 0:00.00 | rcu_par_gp |
| 8 | root | 0 | -20 | 0 | 0 | 0 | I | 0,0 | 0,0 | 0:00.00 | mm_percpu+ |
| 9 | root | 20 | 0 | 0 | 0 | 0 | S | 0,0 | 0,0 | 0:00.20 | ksoftirqd+ |
| 10 | root | 20 | 0 | 0 | 0 | 0 | I | 0,0 | 0,0 | 0:02.02 | rcu_sched |
| 11 | root | 20 | 0 | 0 | 0 | 0 | I | 0,0 | 0,0 | 0:00.00 | rcu_bh |
| 12 | root | rt | 0 | 0 | 0 | 0 | S | 0,0 | 0,0 | 0:00.06 | migration+ |
| 13 | root | 20 | 0 | 0 | 0 | 0 | S | 0,0 | 0,0 | 0:00.00 | cpuhp/0 |

Fig. 9: CPU Coral.

- CPU – MobileNet_SSD: 242.7%

pi@raspberrypi: ~

Archivo Editar Pestañas Ayuda

```
top - 15:07:55 up 1:18, 2 users, load average: 1,99, 1,30, 1,03
Tasks: 146 total, 2 running, 143 sleeping, 0 stopped, 1 zombie
%Cpu(s): 60,8 us, 3,6 sy, 0,0 ni, 35,5 id, 0,1 wa, 0,0 hi, 0,0 si, 0,0 st
MiB Mem : 882,5 total, 108,3 free, 354,7 used, 419,5 buff/cache
MiB Swap: 100,0 total, 100,0 free, 0,0 used. 459,1 avail Mem
```

| PID | USER | PR | NI | VIRT | RES | SHR | S | %CPU | %MEM | TIME+ | COMMAND |
|------|------|----|-----|--------|--------|-------|---|-------|------|---------|------------|
| 3525 | root | 20 | 0 | 415016 | 204448 | 74068 | R | 242,7 | 22,6 | 1:30.43 | python |
| 865 | pi | 20 | 0 | 87192 | 39308 | 13916 | S | 6,3 | 4,3 | 8:45.03 | thonny |
| 451 | root | 20 | 0 | 308176 | 89760 | 33924 | S | 3,6 | 9,9 | 3:16.35 | Xorg |
| 1791 | pi | 20 | 0 | 238132 | 51148 | 38792 | S | 2,6 | 5,7 | 0:51.67 | lxterminal |
| 2006 | pi | 20 | 0 | 245468 | 74620 | 47196 | S | 1,7 | 8,3 | 0:52.66 | python3 |
| 3450 | pi | 20 | 0 | 10188 | 2812 | 2444 | R | 0,7 | 0,3 | 0:01.31 | top |
| 10 | root | 20 | 0 | 0 | 0 | 0 | I | 0,3 | 0,0 | 0:02.11 | rcu_sched |
| 328 | root | 20 | 0 | 27656 | 80 | 0 | S | 0,3 | 0,0 | 0:02.77 | rngd |
| 1 | root | 20 | 0 | 33716 | 8104 | 6504 | S | 0,0 | 0,9 | 0:05.07 | systemd |
| 2 | root | 20 | 0 | 0 | 0 | 0 | S | 0,0 | 0,0 | 0:00.01 | kthreadd |
| 3 | root | 0 | -20 | 0 | 0 | 0 | I | 0,0 | 0,0 | 0:00.00 | rcu_gp |
| 4 | root | 0 | -20 | 0 | 0 | 0 | I | 0,0 | 0,0 | 0:00.00 | rcu_par_gp |
| 8 | root | 0 | -20 | 0 | 0 | 0 | I | 0,0 | 0,0 | 0:00.00 | mm_percpu+ |
| 9 | root | 20 | 0 | 0 | 0 | 0 | S | 0,0 | 0,0 | 0:00.20 | ksoftirqd+ |
| 11 | root | 20 | 0 | 0 | 0 | 0 | I | 0,0 | 0,0 | 0:00.00 | rcu_bh |
| 12 | root | rt | 0 | 0 | 0 | 0 | S | 0,0 | 0,0 | 0:00.06 | migration+ |
| 13 | root | 20 | 0 | 0 | 0 | 0 | S | 0,0 | 0,0 | 0:00.00 | cpuhp/0 |

Fig. 10: CPU MobilNet_SSD



Con esta información, es correcto decir que MobileNet_SSD usa demasiada CPU, ya que está analizando imágenes y también realizando inferencias. En cambio, con Coral, la CPU sólo se preocupa del procesamiento de imágenes, evitando así sobreexigir el dispositivo.

Es necesario mencionar que el código implementado con Coral usa un modelo de MobileNet SSD V2⁵, pero de manera externa a la CPU de la Raspberry Pi.

⁵ <https://machinethink.net/blog/mobilenet-v2/>

3.2 Webcam LifeCam HD-3000 USB => Pi Camera V2

Las descripciones ubicadas en las secciones [2.3.1], [2.3.2] y [2.5] son necesarias en el análisis del cambio.

Como bien se mencionan las características de la webcam y Raspberry PI, estas son incompatibles en cuanto al manejo de memoria RAM. Recordar que, en el peor de los casos, en la webcam se usaba 1 GB de RAM, mientras que la capacidad de Raspberry Pi es de 1 GB, por lo que la cámara nunca alcanzaría una eficiencia ideal. Puede servir como referencia tomar en cuenta el análisis de CPU de la sección anterior, en la que aparecía el funcionamiento de la CPU en un estado normal, en la que es visible que la memoria total en aquella instancia es de 882.5 MB.

También, notar el manejo de procesador y frecuencia. Para el caso de la webcam, esta necesita al menos una CPU Dual Core de 1.6 GHz, mientras que la Raspberry PI ofrece una CPU Quad Core de 1.2 GHz. Nuevamente está presente el manejo de las capacidades de la placa sobre un dispositivo externo.

Si bien ambas cámaras proveen al dispositivo de imágenes con resoluciones altas (1080p), la velocidad con que se maneja esta información difiere de este carácter. Si esta diferencia se profundiza con detalle es posible notar que la webcam tiene una interfaz USB 2.0, es decir maneja 480 Mb/s (60 MB/s), pero a una tasa real práctica máxima de 280 Mb/s (35 MB/s)⁶; mientras que Pi Camera V2 posee una interfaz CSI, es decir, permite velocidades de datos de hasta 800 Mb/s (100 MB/s) por línea con 1 Gb/s (125 MB/s) establecido como límite práctico⁷.

Por último, si bien el análisis del párrafo anterior dejó en evidencia la diferencia de velocidades que pueden proporcionar las distintas interfaces, también es necesario hacer notar a que estructura del hardware arriban estas imágenes. En cuanto a los conectores USB, estos arriban directamente a la CPU, para luego poder ser compensada con la GPU; en cambio, la interfaz CSI arriba directamente a la GPU.

La computación acelerada por GPU permite asignarle a esta el trabajo de los aspectos de la aplicación donde la computación es más intensiva, mientras que el resto del código se ejecuta en la CPU. Desde la perspectiva del usuario, las aplicaciones se ejecutan de forma mucho más rápida⁸.

Mencionando esto, se hace claro el porqué del sobrecalentamiento del dispositivo en el momento en que se utilizaba el modelo de MobileNet_SSD.

⁶ https://es.wikipedia.org/wiki/Universal_Serial_Bus

⁷ <https://es.rasp-berry.net/q/cual-es-la-velocidad-de-la-interfaz-en-serie-de-la-camara-csi-y-su-cable-3108>

⁸ <https://www.nvidia.com/es-la/drivers/what-is-gpu-computing/>

3.3 CLAHE

En esta sección, gracias a los análisis previos, se deja en claro que el uso de Coral será permanente. Es por esto que se procederá a efectuar pruebas para detectar personas con y sin el manejo del algoritmo CLAHE, demostrando el porqué de su uso.

Como base se usará una imagen que presenta poca luminosidad.



Fig. 11: Imagen referencial.

La imagen referencial pasó por un proceso de smoothing (blur) con un kernel de 11x11, con el fin de obtener una imagen difícil y ruidosa en detectar; para luego ser procesada bajo las inferencias de Coral.

Se obtuvieron los siguientes resultados:

- Sin CLAHE:

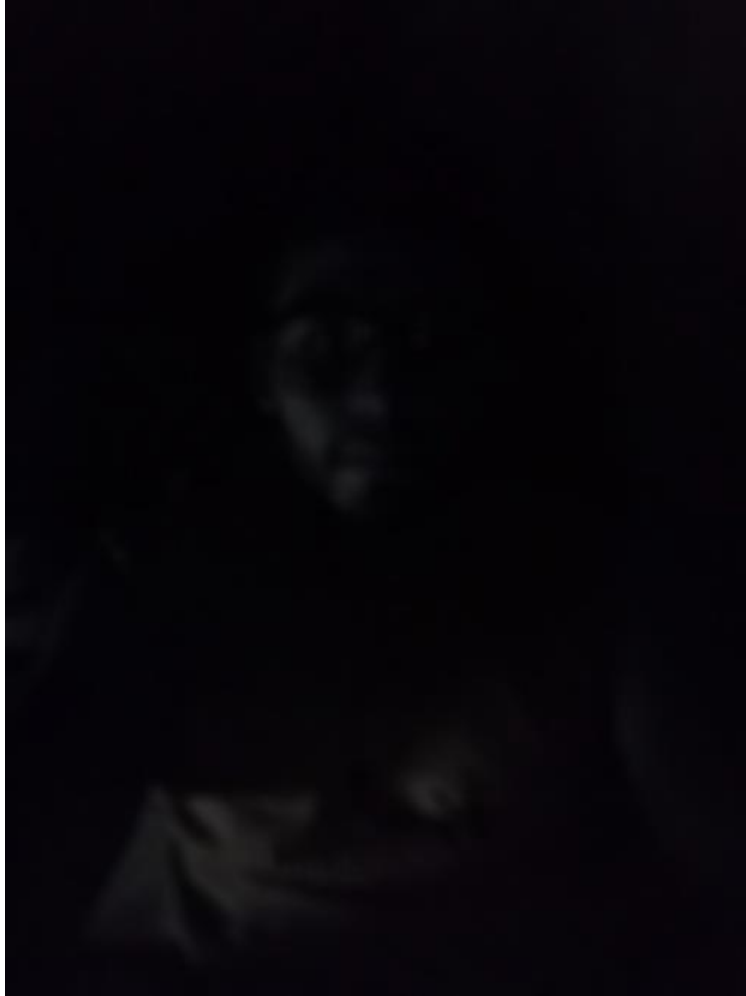


Fig. 12: Imagen con blur de 11x11, sin algoritmo CLAHE.

Es notoria la no detección, a pesar de la utilización de Coral.

- Con CLAHE:



Fig. 13: Imagen con blur de 11x11, con algoritmo CLAHE.

Esta figura demuestra el efecto tras la utilización del algoritmo. El resultado muestra como se resaltan bordes y colores, permitiendo así la detección de personas.

También, permite verificar que de alguna forma está detectando más de una persona. Esto puede ser debido al excesivo ruido entregado a la imagen, previo a la detección con el TPU de Google.

Estos resultados permiten llegar a conclusiones precipitadas en base a los videos anexados en el archivo "Avances_DetPax", subcarpeta "Coral". Aun así, es necesario tener en mente que los videos se miden hasta el minuto 2:03, y, también recordando el 200% base que se obtuvo de la sección [3.1].

Por lo tanto, no es posible llegar a estimaciones estadísticas confiables con tan pocas mediciones. Para esto, se necesitan más pruebas.

Sin embargo, el poder de perfeccionar la detección de personas bajo condiciones de borrosidad y baja luminosidad es evidente.

4. Mejoras

1. Temperatura:

Gracias a la sección [3.1], se pudo ver que el dispositivo puede presentar situaciones de sobrecalentamiento.

Ya que la CPU de la Raspberry Pi en el mejor de los casos alcanza un 108% aproximado, es evidente que en algún determinado momento incurrirá en una subida de temperatura. Es más, si se considera que la CPU está a más del 100% de trabajo, está usando los 1.2GHz del procesador, por lo tanto, la temperatura aumentará por sobre los 80°C, incurriendo así en la aparición de la señal de sobrecalentamiento del sistema⁹. El problema con esta situación es que la Raspberry comenzará a reducir su frecuencia para reducir el calor emitido por la CPU. Su velocidad será gradualmente reducida, bajando de los 600MHz si la placa alcanza una temperatura sobre los 85°C¹⁰.

Para esto, se acudió a incorporar elementos refrigerantes:

- Disipadores:
 - Especificaciones:
 - Set de 3 disipadores de calor (para CPU, chip LAN y chip memoria RAM)
 - Cada disipador viene con una cinta adhesiva térmica
 - Resistencia térmica de 27°C/W

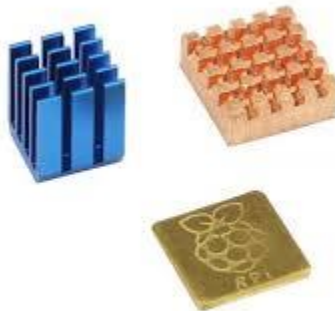


Fig. 14: Conjunto de disipadores.

- Ventilador: Para este caso, se presentaron 2 opciones:

⁹ <https://www.raspberrypi.org/documentation/configuration/warning-icons.md>

¹⁰ <https://www.zdnet.com/article/no-your-raspberry-pi-3-wont-overheat-in-everyday-use-says-its-creator/>

- Ventilador grande: Este ventilador se caracteriza por ser el más grande para Raspberry Pi, además de tener tornillos que permitirían instalarlo en alguna superficie cercana o anexada a la placa.



Fig. 15: Ventilador grande.

- Ventilador doble: Este ventilador posee un disipador acoplado a su base, y también incluye disipadores extras. Posee la ventaja de ser de un tamaño más reducido, pudiendo instalarse sobre la placa.



Fig. 16: Ventilador doble.

La pregunta que se presenta es: ¿Será necesaria la inclusión de ambos elementos, disipadores y ventiladores? La respuesta es sí. Esto se debe a que los disipadores no dan abasto en una situación de máxima frecuencia. La comprobación de esta situación arrojó que cuando se instalan solo disipadores, la Raspberry logra mantener una temperatura de 82.7°C. Con esto, se hace evidente la aparición de un ventilador en el dispositivo, pero la duda que se presenta es: ¿Cuál será el indicado? La decisión estuvo influenciada por la estructura de la caja que contiene al DetPax.



Fig. 17: Dispositivo DetPax.

Con la figura 17, es notoria la estrechez disponible, por lo cual, se decidió por implementar el segundo ventilador, el doble. Con esta decisión, se ahorra la idea de tener que fabricar otra caja con impresora 3D.



Fig. 18: Raspberry Pi con ventilador doble instalado.

Entonces, en base a la decisión, la utilización del ventilador doble es necesaria para no rondar el límite máximo de calor del sistema. El ventilador demostró que alcanza una temperatura máxima de 66.6°C bajo condiciones exigentes¹¹.

Sin embargo, el manejo de cualquier ventilador produce un problema en la arquitectura actual del DetPax. Como es visible en la figura 17, la Raspberry Pi posee una pantalla touchscreen de 3.5 pulgadas, la cual esta conectada a los puertos GPIO de la placa.

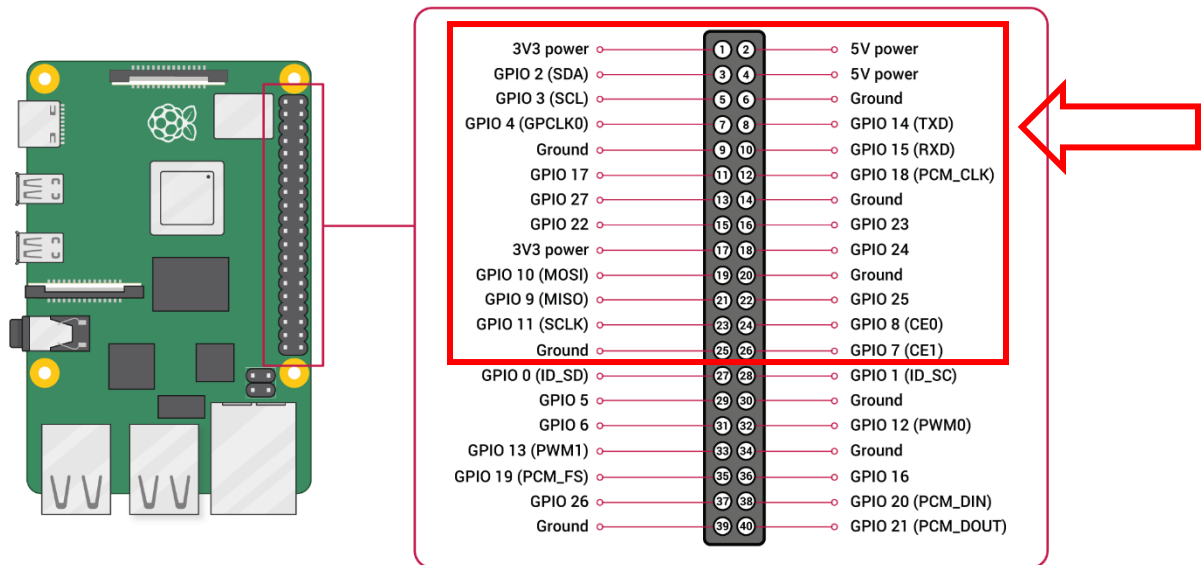


Fig. 19: Conexión de pantalla a Raspberry Pi.

¹¹ <https://www.youtube.com/watch?v=GpEP37-UG8c>

Mientras que los ventiladores se conectan a los siguientes puertos GPIO:

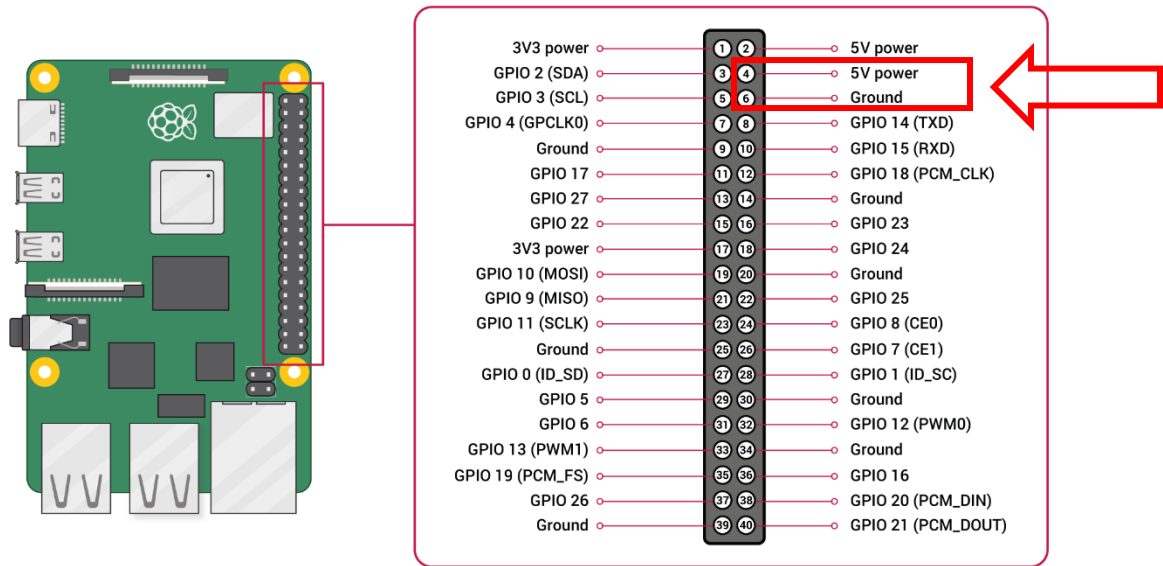


Fig. 20: Conexión de ventilador a Raspberry Pi.

En consecuencia, se presenta una disyuntiva en cuanto a la implementación de dispositivos que ayudan en el manejo y desempeño del dispositivo.

Por esta razón, tanto el ventilador como los disipadores, aún no son instalados.

2. Interfaz gráfica:

Se implementó en el código la librería e interfaz gráfica Tkinter de Python.

La finalidad de este código es preguntar al usuario si quiere que el streaming sea mostrado en la pantalla, o simplemente terminar el programa.

El fin de esta implementación era ahorrar recursos de CPU, mientras que el video resultante es grabado para su posterior análisis.

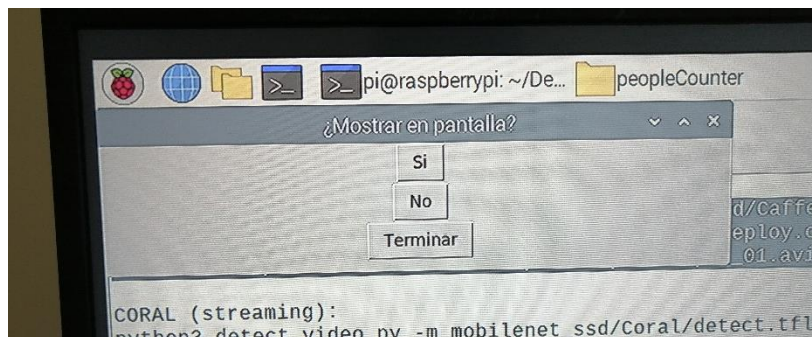


Fig. 21: Interfaz de Tkinter.

5. Propuestas

1. Propuesta “Pantalla – Interfaz DSI”:

Frente al problema expuesto al final de la sección [4], es posible resolverlo mediante el cambio de pantalla. Si bien la pantalla anterior se conectaba mediante los puertos GPIO, es posible obtener pantallas con las mismas características, pero con una interfaz DSI. Cabe notar que en este punto no se está buscando una optimización de rendimiento, sino que una acomodación de la arquitectura.



Fig. 22: Conexión con interfaz DSI de pantalla – Raspberry Pi.

Esta modificación permitiría el uso en paralelo del ventilador y pantalla.

Sin embargo, esta solución puede recaer sobre la modificación de la caja que protege al dispositivo. Se pueden buscar soluciones alternas.

2. Propuesta “Raspberry pi 4”:

Uno de los tópicos importantes del informe ha sido el manejo y procesamiento de datos. Si bien, el acelerador de Coral permitió un avance importante en la detección, el DetPax

presenta un “cuello de botella” con relación a este dispositivo. Esto se debe a que Raspberry Pi 3B posee conectores USB 2.0, mientras que Coral usa un USB 3.0¹².

USB 3.0 tiene una velocidad teórica de transmisión de hasta 4,8 Gbit/s o 600MB/s (SuperSpeed USB SS), que es diez veces más rápido que USB 2.0 (480 Mbit/s o 60 MB/s). USB 3.0 reduce significativamente el tiempo requerido para la transmisión de datos, reduce el consumo de energía y es compatible con USB 2.0¹³.

Es por esta razón que se estima una implementación más conveniente con una Raspberry Pi 4 modelo B, la cual presenta 2 puertos USB 3.0, entre otras características más.

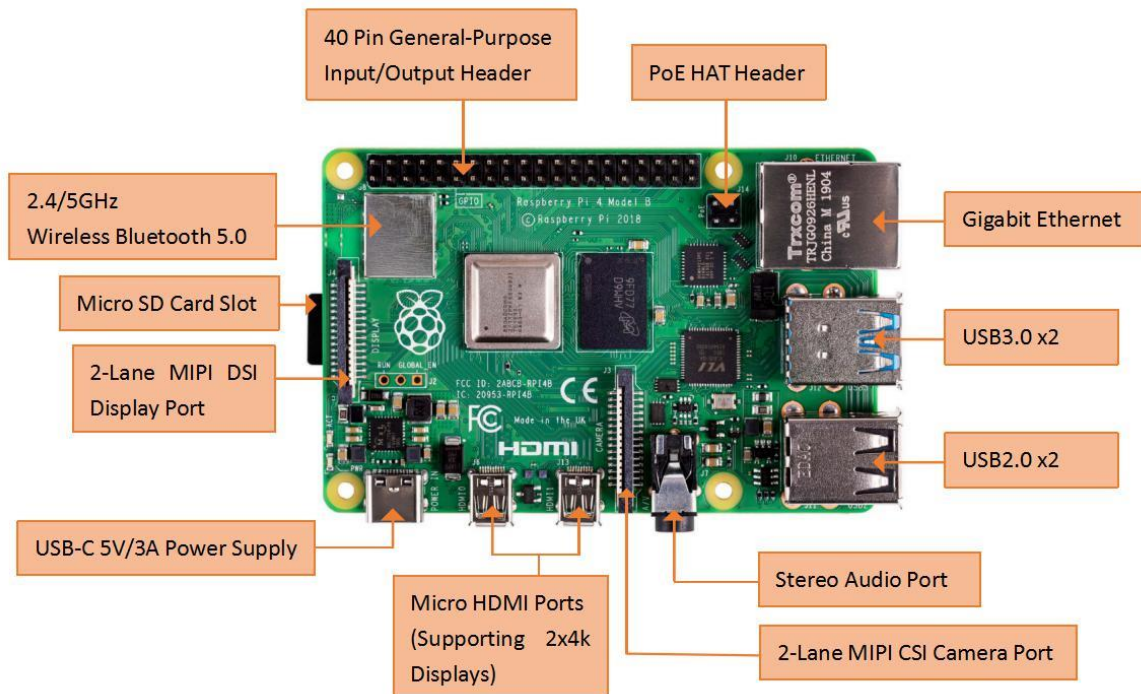


Fig. 23: Raspberry Pi 4B.

¹² <https://coral.ai/docs/accelerator/datasheet/>

¹³ https://es.wikipedia.org/wiki/USB_3.0



6. Conclusión

En vista de los análisis y resultados obtenidos, es viable una mejoría en el proyecto, siempre y cuando se puedan seguir implementando actualizaciones tanto de hardware como de software.

En cuanto a las mejorías en hardware, se podría revisar la implementación de los elementos propuestos.

En cambio, en las mejorías de software, se deben realizar pruebas aparte de videos, sino que como streaming; y, también realizar pruebas con distintos modelos de Edge TPU para verificar si existen distintos cambios.

Es necesaria la realización de pruebas en condiciones reales, para poder decidir cuales son las ubicaciones ideales, es decir, ángulo, distancia, altura, etc. Incluso, estas pruebas servirían para realizar investigaciones estadísticas, y así incurrir en estudios verídicos del desempeño del dispositivo.



7. Referencias

1. <https://honingds.com/blog/ssd-single-shot-object-detection-mobilenet-opencv/>
2. <https://www.pcfactory.cl/producto/17516-microsoft-webcam-hd-3000>
3. https://www.raspberrypi.org/documentation/hardware/computemodule/datasheets/rpi_DATA_CM3plus_1p0.pdf
4. <https://www.programcreek.com/python/example/89353/cv2.createCLAHE>
5. https://docs.opencv.org/3.4/de/d25/imgproc_color_conversions.html
6. https://docs.opencv.org/master/d5/daf/tutorial_py_histogram_equalization.html
7. https://en.wikipedia.org/wiki/Adaptive_histogram_equalization#Contrast_Limited_AHE
8. <https://www.raspberrypi.org/products/raspberry-pi-3-model-b/>
9. <https://www.mcielectronics.cl/shop/product/raspberry-pi-3-modelo-b-19766>
10. <https://coral.ai/docs/reference/edgetpu.detection.engine/>
11. <https://www.amazon.es/Raspberry-Pi-Modelo-Quad-Core-Cortex-A53/dp/B01CD5VC92>
12. <https://www.raspberrypi.org/forums/viewtopic.php?t=85899>
13. <https://altronics.cl/set-disipadores-raspberry-pi3-bplus>
14. <https://coral.ai/docs/accelerator/get-started/>
15. https://www.dfrobot.com/product-1875.html?tracking=5d230df38b98e&gclid=EAlaIQobChMI1YHnxd676AIVig-RCh3EWQsJEAAAYASAAEgJlpfD_BwE
16. <https://coral.ai/models/>