UNIVERSIDAD DEL SINÚ
Elías Bechara Zainúm
Seccional Cartagena

DESARROLLO DE UN SISTEMA BÁSICO DE INFERENCIA DE INTELIGENCIA ARTIFICIAL A BAJO COSTO UTILIZANDO UNA TPU GOOGLE CORAL Y UNA RASPBERRY PI.

CARLOS EDUARDO OSPINO MARTÍNEZ

UNIVERSIDAD DEL SINÚ ELÍAS BECHARÁ ZAINÚM SECCIONAL CARTAGENA
ESCUELA DE INGENIERÍA DE SISTEMAS
CARTAGENA-COLOMBIA
2020

UNIVERSIDAD DEL SINÚ
Elías Bechara Zainúm
Seccional Cartagena

DESARROLLO DE UN SISTEMA BÁSICO DE INFERENCIA DE INTELIGENCIA ARTIFICIAL A BAJO COSTO UTILIZANDO UNA TPU GOOGLE CORAL Y UNA RASPBERRY PI.

CARLOS EDUARDO OSPINO MARTÍNEZ

Asesor disciplinar
LUIS MURILLO FERNÁNDEZ

Asesor metodológico
EUGENIA ARRIETA RODRÍGUEZ

UNIVERSIDAD DEL SINÚ ELÍAS BECHARÁ ZAINÚM SECCIONAL CARTAGENA
ESCUELA DE INGENIERÍA DE SISTEMAS
CARTAGENA-COLOMBIA
2020

Acta de Calificación y aprobación

Notas de aceptación

_____

_____

_____

_____

Director de escuela

_____

Director de investigaciones

_____

Firma de jurado

_____

Firma de jurado

Cartagena de indias, 2020

Contenido

Tabla de Figuras

Lista de Tablas

RESUMEN

Este proyecto presenta el estudio de las funcionalidades de la TPU Coral a partir del análisis de los algoritmos de prueba. El resultado esperado es un sistema básico de inferencia de inteligencia artificial basado en la implementación de una TPU GOOGLE CORAL Y una Raspberry Pi, la cual nos lleva a indagar sobre temas y tecnologías que actualmente facilitan el estudio de reconocimiento de objetos a un bajo costo en diversos campos como la medicina, el deporte, animaciones, entretenimiento, entre otras.

Palabras Claves – Algoritmos de inferencia, Análisis Coste – Beneficios, Google Coral, Inferencia, Inteligencia Artificial, Retardo de Fotograma.

INTRODUCCIÓN

La inteligencia artificial (IA) engloba numerosos campos que facilitan el estudio de la misma, mediante la visión artificial un sistema inteligente es capaz de extraer información de un entorno para su interpretación mediante el uso de la computadora, esta es utilizada para muchas aplicaciones, entre ellas se pueden destacar el acceso a tecnologías para la captura de movimiento (Mocap, del inglés motion capture). Estos procesos permiten automatizar el proceso de obtención de información basadas en la captura de la imagen (cámaras de vídeo, cámaras web), útiles para el desarrollo de proyectos de cine, televisión, videojuegos, estudios clínicos, biomecánicos o para fines pedagógicos.

Hoy en día existen sistemas avanzados para la captura de movimiento del cuerpo, sin embargo, la mayoría es de uso restringido a condiciones de laboratorio con un costo elevado, sin mencionar que se requiere de personal altamente capacitado para manipular este tipo de sistemas. Entre los cuales se encuentran los sistemas ópticos estacionarios y portables [1], que consisten en una red de cámaras, usualmente infrarrojas, vídeo cámaras, sensores y marcadores (algunos sistemas no requieren marcadores) que convierten la información real en datos digitales para su tratamiento en entornos virtuales, calibradas entre sí en un espacio confinado o laboratorio, sin embargo, entre las limitaciones se encuentran el alto costo de estos sistemas para su utilización en diversos entornos de trabajo, la colocación de los marcadores en la piel requieren de una cantidad significativa de tiempo y del conocimiento de un experto.

En cuanto a implementación para el procesamiento de los datos en un dispositivo final, existen infinidades de equipos con alta tecnología (computadoras cuánticas, procesador de núcleo múltiple), que son de mucha utilidad para implementar este tipo de arquitectura que requieren un alto consumo, pero estos tienden ser de costos

elevados, lo que conlleva a que sean poco accesibles para ser implementando en espacios educativos, ya que estos no carecen de estos recursos. En los últimos años con su gran demanda, se ha logrado que los costos de implementación de estos sistemas bajen considerablemente, pero estos resultados también demuestran que al no contar con recursos adecuados disponen de equipos que no son indicados, por su bajo rendimiento, dificultando el uso y saturación de la información.

El Raspberry Pi es una solución económica para la aplicación de distintos algoritmos y soluciones para el desarrollo de sistemas que deben realizar procesamiento de imágenes. Este proyecto está encaminado al desarrollo de un sistema básico de inferencia de inteligencia artificial a bajo costo utilizando una TPU GOOGLE coral y una Raspberry Pi.

1. DISEÑO METODOLÓGICO

Dentro de este capítulo se detalla la metodología empleada para obtener la información que se necesita en la elaboración del presente proyecto. Se especifican los elementos necesarios para llevar a cabo el análisis del funcionamiento de la TPU Google Coral, entre estos se identifican descripción del problema, justificación, formulación del problema, objetivo general y específico, alcance, estado del arte, marcos de referencia y metodología a utilizar para el desarrollo de este.

1.1. Descripción del problema

En los últimos años la inteligencia artificial se ha transformado en una herramienta transcendental que ha generado cambios sustanciales en nuestro día a día, se perfila como uno de los mayores avances tecnológicos del siglo XXI. Es difícil encontrar una industria que no haya visto su modelo de negocio afectado por la influencia de la automatización gracias al desarrollo de este tipo de inteligencia. Se denomina Inteligencia Artificial a la facultad de razonamiento que tiene un agente no vivo, como es el caso de los robots. Cabe destacarse que además del poder de razonar, estos dispositivos son capaces de desarrollar muchas conductas y actividades especialmente humanas como puede ser resolver un problema dado, practicar un deporte, entre otros. Podemos entender una inteligencia artificial como aquellos algoritmos que se materializan en programas informáticos que, a su vez, corren sobre un hardware determinado, y que persiguen imitar el modo de funcionamiento del cerebro humano.

Google lanzo una unidad de hardware para aceleración de inferencia llamado TPU (Tensor processing unit) el propósito de esta USB CORAL es permitir que los dispositivos como microprocesadores exploten en el campo de la inteligencia artificial, es un coprocesador USB que realiza inferencias de aprendizaje automático a los sistemas existentes a partir de las librerías TensorFlow, Funciona con

Raspberry Pi y otros sistemas Linux. En la actualidad en el campo laboral de los ingenieros de sistemas es necesario el conocimiento de IA (Inteligencia Artificial) ya que ha tenido un gran impacto y actualmente la mayor parte de los desarrollos de software y hardware incluyen temas de esta herramienta que está causando gran efecto en el mundo económico y en la sociedad.

La Inteligencia Artificial está provocando una revolución en las herramientas de aprendizaje, ofreciendo métodos educativos más intuitivos y flexibles. De esta manera, los dispositivos inteligentes son capaces de dar formación de medida a cada estudiante. Integrar las tecnologías en la educación supone grandes beneficios para el aprendizaje de los alumnos.

1.2. Justificación

Se han desarrollado recientemente unidades de bajo costo de inteligencia artificial como la TPU Google Coral que permiten desarrollar sistemas de inferencias de IA (Inteligencia Artificial) que utilizando hardware genérico e incorporando estos coprocesadores permiten construir sistema hardware-software de inteligencia artificial a un costo relativamente bajo y muy fáciles de construir.

Una solución a este problema planteado es que mediante el plan de estudio de ingeniería de sistema en la Universidad del Sinú incorporen herramientas hardware en la enseñanza del uso de la inteligencia artificial que mediante todas estas herramientas ayudará a tener un conocimiento profundo en IA, ya que cada día va evolucionando y con esta evolución las empresas van incrementando el uso de esta tecnología. La inteligencia artificial no reemplazará el trabajo de los desarrolladores si no que enriquecerá cada uno de los conocimientos obtenidos.

1.3. Formulación del problema

¿Es posible desarrollar un sistema de inferencia de inteligencia artificial a bajo costo utilizando una TPU Google Coral que permita ser utilizada como una herramienta de aprendizaje de sistemas de inteligencia artificial en la Universidad del Sinú?

1.4. Alcance

La finalidad de este proyecto es desarrollar un sistema básico de inferencia de inteligencia artificial para la USB TPU GOOGLE CORAL mediante algoritmos básicos de prueba.

El sistema estará basado en la conexión de una cámara Raspberry Pi y un Raspberry Pi, combinado con el acelerador USB de Coral que permite incorporar capacidades de inteligencia artificial. Por recursos disponibles para el desarrollo e implementación del sistema se concluye que se trabaja con una cámara Raspberry Pi, una Raspberry Pi y una USB TPU GOOGLE CORAL.

Para satisfacer los requisitos anteriormente mencionados, se tendrán en cuenta las herramientas que nos permitan cumplirlos debidamente, la selección de software realizada para este proyecto se ha escogido tratando de agilizar todo lo posible en su diseño y robustez. Las cuales son relacionadas en la siguiente tabla:

Tabla 1. Herramientas de Hardware y software a implementar, Fuente Autor

HARDWARE	
1 Cámara Raspberry Pi	1 Monitor
1 Raspberry Pi	1 TPU GOOGLE CORAL
SOFTWARE	
Lenguaje de programación	Python PI
Interfaz de usuario	Thonny
Sistema operativo	Raspbian
Librería	TensorFlow Lite

1.5. Objetivos

Objetivo general

Desarrollar un sistema básico de inferencia de inteligencia artificial para la USB TPU mediante algoritmos básicos de prueba.

Objetivos específicos

- Analizar el funcionamiento de la USB TPU Google CORAL mediante la documentación de sus manuales para la puesta de funcionamiento.

- Implementar un sistema de inferencia a partir de la USB TPU Google CORAL para la aceleración de sistemas de Inteligencia Artificial.

- Verificar la funcionalidad de la TPU, a partir del análisis de los algoritmos de prueba para evaluación de su desempeño.

1.6. Estado del arte

Existen distintas investigaciones y proyectos relacionados con la captura de movimiento y en base del USB TPU GOOGLE CORAL, a continuación, se mencionan los de mayor relevancia, los cuales aportan bases significativas para el desarrollo de este proyecto.

El primer trabajo corresponde a un sistema de captura de movimientos (sistema cámaras Optitrack y software Mocap Arena) realizado en el año 2015, en San Cristóbal – España. Este proyecto fue presentado por estudiantes de la Universidad de La Laguna, la configuración de la arquitectura fue basada en la utilización de 6 cámaras que fueron distribuidas en forma de un hexágono alrededor del área en el cual se realizan la toma de datos, y a una altura de entre 2 y 2,5 metros. Las cámaras estaban en forma de retrato (horizontal) o en apaisado (vertical) dependiendo de las necesidades a la hora de tomar los datos [2].

El sistema realizo la captura de una serie de movimientos, con la participación de distintas personas. Los datos recopilados están almacenados y se encuentran disponibles para la consulta y descarga en la web [3]. Entre las pruebas realizadas se encuentran: prueba de bateo, movimiento de saludo con una mano, negación (movimiento con la cabeza de lado a lado), Afirmación (movimiento con la cabeza de arriba abajo), caminar a baja velocidad, caminar en círculos, caminar y volver, saltos (vertical), Boxeo (serie de puñetazos), entre otros.

Otro trabajo importante es “Motion Capture”, es un software que permite la captura de movimiento, con una a dos cámaras Kinect o cámaras de video, no es exigente con el lugar de trabajo y la iluminación. No es necesario disponer de un traje especial con marcadores reflectantes. El flujo de trabajo también es muy eficiente y sencillo. Después de hacer una calibración del sistema, que con una simple tabla es suficiente, el programa empieza a reconocer la estructura del cuerpo. El software creará un esqueleto básico. Después el programa solo tratará de ir haciendo correcciones [4].

Existen otros paquetes de software que se encuentran en el mercado que realizan funciones similares en la captura del movimiento, entre estos podemos encontrar ProAnalyst y BioSTAGE. ProAnalyst (Xcitex), es para medir automáticamente objetos en movimiento con video. ProAnalyst le permite importar virtualmente cualquier video y rápidamente extraer y cuantificar el movimiento dentro de ese video. Utilizado ampliamente por la NASA, ingenieros, difusores, investigadores y atletas, ProAnalyst es el software ideal para cualquier cámara de video profesional, científico e industrial, y viceversa. Puede ser usado con cualquier cámara para medir cualquier objeto en movimiento. Los resultados del análisis pueden graficarse y revisarse instantáneamente, compararse con datos externos y exportarse a una variedad de formatos de salida para análisis o presentaciones adicionales [5].

BioSTAGE de Organic Motion de NUEVA YORK, sistemas de captura de movimiento sin marcadores multi-cámara por medio de un seguimiento 3D de alta precisión de datos en tiempo real, es una herramienta para los médicos e investigadores con la capacidad de rastrear y analizar con precisión los movimientos de muchas poblaciones diferentes de pacientes que pueden tener dificultades para usar un traje de cuerpo. BioSTAGE integra sus datos de movimiento de cuerpo completo en tiempo real con paquetes de software de análisis biomecánicos líderes. Ejemplo: Debido a que no tiene marcadores, podemos hacer que un paciente entre con facilidad y dentro de un corto período de tiempo, obtenga información exacta sobre el rango de movimiento del paciente y los movimientos totales del cuerpo. Esto no solo mejora drásticamente el proceso de diagnóstico con datos cuantificables, sino que también nos permite medir y seguir el progreso a lo largo del tiempo [6].

The Teachable Machine diseñado por Mike Tyka de Google (2019), La máquina de enseñanza aprende a identificar los objetos que se sostienen frente a ella. Estos

objetos pueden ser cualquier cosa: llaves, fruta, piezas de ajedrez o incluso dedos o caras. El usuario levanta los elementos frente al módulo de cámara Raspberry Pi y presiona un botón en la máquina de enseñanza. Luego, el dispositivo recuerda el objeto que se está sosteniendo; si lo vuelve a ver, se encenderá el LED correspondiente [7].

Comparación de rendimiento y potencia de los aceleradores USB para inferencia con MLPerf [9], de la Universidad Complutense de Madrid (2020). En este trabajo, evaluaron la capacidad de los nuevos aceleradores de inferencia basados en USB para cumplir con los requisitos planteados por las aplicaciones en términos de tiempo de inferencia y consumo de energía. Su objetivo era la adaptación del benchmark MLPerf para manejar dos nuevos dispositivos: Google Coral Edge TPU e Intel NCS2. Además, presentaron una infraestructura de hardware que produce mediciones detalladas de energía para dispositivos alimentados por USB. Los resultados experimentales revelaron ideas interesantes en términos de rendimiento y eficiencia energética para ambas arquitecturas [10].

Accelerator-Aware diseño de redes neurales utilizando AutoML (2020), este proyecto consiste en presentar una clase de modelos de visión por computadora diseñados con búsqueda de arquitectura neuronal con reconocimiento de hardware para ejecutarse en Edge TPU, el acelerador de hardware de red neuronal de Google para dispositivos de borde de baja potencia. Para el Edge TPU en dispositivos Coral, estos modelos permiten el rendimiento de clasificación de imágenes en tiempo real, al tiempo que logran una precisión que generalmente se ve solo con modelos más grandes y pesados en cómputo que se ejecutan en centros de datos. En el TPU Edge de Pixel 4, estos modelos mejoran la compensación de la latencia de precisión sobre los modelos móviles SoTA existentes [11].

A continuación, se detallan otros proyectos que son relacionados con nuestro objetivo, pero teniendo en cuenta que el lanzamiento de la TPU Coral fue realizado a principios del año 2019, no se encuentran proyectos enfocados en la implementación de esta. Por lo tanto, se considera pertinente la investigación de

dispositivos similares como es el Movidius Neural Compute Stick, que también fue desarrollado para realizar aplicaciones de inferencia de aprendizaje profundo, de bajo consumo y bajo consumo de energía de Intel.

Prototipo funcional para clasificación de imágenes con salida de audio en un sistema embebido con red neuronal convolucional, proyecto presentado como trabajo de grado de la Universidad Autónoma Metropolitana en México (2018). Realizaron un prototipo basado en la Raspberry Pi 3, que realiza clasificación de imágenes con reproducción de audio. La clasificación es realizada por la CNN (convolutional neural networks) GoogleNet, la cual es entrenada fuera de línea, implementada en un NCS (**Neural Compute Stick)** e integrada a la tarjeta Raspberry Pi 3. La Raspberry captura 18 fotogramas cada 3 segundos, tiempo en el cual, el NCS realiza 18 clasificaciones de las imágenes instantáneas. Si durante los 3 segundo, la salida del NCS es la misma clase, la Raspberry traduce en audio la etiqueta de la clase para describir el objeto encontrado en la imagen y guarda en disco una copia de la imagen junto con su etiqueta [8].

1.7. Marcos de referencia

1.7.1. Marco Teórico

Clasificación de sistemas de captura de movimiento.

En la actualidad son numerosos los sistemas de captura de movimientos (mocap) y las tecnologías relacionadas con estos, las cuales consisten en el análisis y grabación de los datos de un movimiento. En la ***figura 1***, se describen brevemente algunos de estos y posteriormente serán detallados.

Figura 1. Sistemas de Movimiento, Tomado de [12]

Pueden clasificarse según su tecnología en [12]:

- Electromecánicos, la captura del movimiento se realiza usando sensores mecánicos. En el proceso de captura de movimiento, la persona viste unos trajes especiales, adaptables al cuerpo humano. Estos trajes son generalmente estructuras rígidas compuestas de barras metálicas o plásticas, unidas mediante potenciómetros colocados en las principales

articulaciones. Básicamente, el actor coloca la estructura en su cuerpo y mientras se mueve, el traje se adapta a los movimientos que este realiza. La desventaja de estos sistemas, es la incapacidad de medir traslaciones globales (miden las posiciones relativas de los miembros, pero no el desplazamiento del actor en el escenario). Por otro lado, dicho sistema supone que la mayoría de los huesos humanos están unidos por articulaciones de un grado de libertad con centro instantáneo de rotación fijo, pero no tiene en cuenta rotaciones complejas que se producen en las articulaciones humanas; las estructuras suelen ser rígidas y restringen el movimiento del actor.

- Electromagnéticos, se dispone de una colección de sensores electromagnéticos que miden la relación espacial con un transmisor cercano. Los sensores se colocan en el cuerpo y se conectan a una unidad electrónica central; están constituidos por tres espiras ortogonales que miden el flujo magnético, determinando posición y orientación del sensor. Un transmisor genera un campo electromagnético de baja frecuencia que los receptores detectan y transmiten a la unidad electrónica de control.
- Inerciales, se colocan sensores inerciales en distintas partes del cuerpo (acelerómetros triaxiales y giroscopios). Una ventaja es que se obtienen datos precisos de aceleración y orientación del individuo. Sin embargo, no es posible medir traslaciones globales y una desventaja es que estos sensores son muy sensibles a cambios en los campos magnéticos.
- Ópticos utilizan los datos recogidos por sensores de imagen para inferir la posición de un elemento en el espacio, utilizando una o más cámaras sincronizadas para proporcionar proyecciones simultáneas. Generalmente se usan marcadores pegados al actor, pero los sistemas más recientes permiten recoger datos confiables, rastreando superficies del sujeto identificadas dinámicamente. Estos sistemas entregan la posición cartesiana (*x,y,z*) de cada marcador en un marco de referencia inercial; la orientación de una superficie se calcula utilizando la posición relativa, de al menos, 3

marcadores. Los sistemas ópticos de captura de movimiento, permiten la grabación en tiempo real, con algunas limitaciones como son: el número de cámaras, marcadores y actores.

OpenCV (Open Source Computer Vision)

Es una librería multiplataforma originalmente desarrollada como proyecto por Intel para apoyar a los primeros compiladores Intel C++ y Microsoft Visual C++ en x86. Es muy utilizada actualmente para el procesamiento de imágenes y la visión artificial en general, también se ha utilizado en infinidad de aplicaciones como control de procesos, sistemas de seguridad con detección de movimiento, reconocimiento de objetos, robótica avanzada, etc. Según la página oficial de OpenCV, esta es una librería completamente libre y gratuita, pues se distribuye bajo licencia BSD (Berkeley Software Distribution), que permite que sea usada libremente en distintos proyectos con propósitos comerciales y de investigación, siempre y cuando, cumpla con las condiciones de la licencia. [13]

OpenCV tiene interfaces C/C++, Python y Java, y soporta Windows, Mac OS, GNU/Linux, iOS y Android. La biblioteca está escrita en un lenguaje C/C++ optimizado y orientado a la eficiencia computacional con un enfoque especial en aplicaciones en tiempo real. Con una estructura modular, lo que quiere decir que el paquete completo incluye varias bibliotecas compartidas. Los módulos que se incluyen dentro de este paquete son:

- **Core functionality (Funcionalidad básica):** En este módulo se definen las definiciones de las estructuras básicas, incluyendo la Matriz multidimensional 'Mat' y funciones básicas usadas por el resto de módulos.
- **Image processing (Procesamiento de imagen):** En este módulo se incluyen filtros de imagen lineales y no lineales, transformaciones geométricas de imagen, conversiones del espacio de colores, histogramas, etc.

- **Video (Vídeo):** Un módulo para el análisis de vídeo que incluye estimación de movimiento, substracción de fondo y algoritmos de seguimiento de objetos.
- **Calib3d (Calibración 3D):** Algoritmos básicos de geometrías de múltiple vista, calibración simple y estéreo, estimación de la posición de objetos, algoritmos de correspondencia estéreo y elementos de reconstrucción 3D.
- **Features2d (Características 2D):** Detectores de características salientes, descriptores y descriptores de coincidencias.
- **Objdetect (Detección de objetos):** Algoritmos de detección de objetos e instancias de clases predefinidas (por ejemplo: caras, ojos, personas, coches, etc.).
- **Highgui:** Una interfaz de uso fácil para simplificar la UI (Interfaz de usuario, del inglés: User Interface).
- **Videoio:** Una interfaz de uso fácil para captura de vídeo y códecs (codificador-decodificador) de vídeo.
- **Gpu:** Algoritmos de diferentes módulos de OpenCV con aceleración de GPU (Unidad de Procesamiento de Gráficos, del inglés: Graphics Processing Unit).

Raspberry Pi

Es un pequeño ordenador capaz, que puede ser utilizado para realizar muchas actividades que realiza un PC de escritorio, como hojas de cálculo, procesadores de texto y también reproduce vídeo de alta definición. [14]

Características:

La dimensión de la placa es de 8,5 cm por 53 cm.

- Chip integrado Broadcom BCM2835.
- Procesador ARM11
- Memoria RAM de 512 MB
- Salida de video y audio
- Conexión Ethernet 10/100
- Adaptador Wi-fi USB
- Salida Analógica de video RCA
- Pines de entrada y salida de propósito general
- Conector de alimentación microUSB
- Lector de tarjetas SD

Modelos de Raspberry Pi

- **Modelo A:** Este modelo solo contiene un puerto USB, carece de controlador Ethernet y cuesta menos que el modelo B. A pesar de que este no tiene un puerto RJ45, se puede conectar a una red usando un adaptador USB-Ethernet suministrado por el usuario. Contiene 256 MB de memoria de RAM.

- **Modelo B:** Este modelo tiene varias versiones, es un dispositivo de un tamaño diminuto (mide casi lo mismo que una tarjeta de crédito), pero en sus ínfimas dimensiones de 85,6 x 53,98 x 17 mm atesora grandes posibilidades, como la posibilidad de mostrar video 1080p o conectarse a redes y a internet y administrar dispositivos de domótica.

Figura 2 Raspberry PI 3 (izquierda) y Raspberry PI 1 (derecha), Tomada de [8]

TPU GOOGLE CORAL

La Google Coral se ha desarrollado únicamente con el objetivo de realizar tareas de machine learning que se puedan integrar en tareas de producción de manera rápida. Para ello, esta nueva SBC hace uso de la red neuronal TensorFlow Lite y del Módulo Edge TPU. Es en este módulo, que es de tipo SOM (System on Module) donde van ensamblados el procesador, la GPU, la RAM, el chip del WiFi y la memoria Flash. Pero, a diferencia de lo que sucede con las Raspberry Pi, este módulo puede ser reemplazado por otro diferente en cuanto a capacidades [15].

Las especificaciones técnicas del Módulo Edge TPU son:

- CPU: NXP i.MXM 8M SOC (Cortex-A53 quad core)
- GPU: integrada, GC7000 Lite Graphics
- Coprocesador: Google Edge TPU
- RAM: 1 GB LPDDR4
- Almacenamiento: Flash eMMC de 8 GB
- Conectividad: WiFi 2×2 MIMO de doble banda y Bluetooth 4.1
- Dimensiones: 48 x 40 x 5 mm

1.7.2. Marco conceptual

OpenCV: es una biblioteca libre de visión artificial, que cuenta con más de 2.500 algoritmos ya que son estos los encargados de hacer posible encontrar imágenes similares, identificar rostros, redes neuronales artificiales, soporte de máquinas vectoriales, calibrar cámaras, clasificar acciones humanas en vídeo y extraer modelos 3D entre muchas otras cosas más [16].

Python: es un lenguaje de scripting independiente de plataforma y orientado a objetos, preparado para realizar cualquier tipo de programa, desde aplicaciones Windows a servidores de red o incluso, páginas web. Es un lenguaje interpretado, lo que significa que no se necesita compilar el código fuente para poder ejecutarlo, lo que ofrece ventajas como la rapidez de desarrollo e inconvenientes como una menor velocidad [17].

Procesamientos de datos: acumulación y manipulación de elementos de datos para producir información significativa [18].

Raspberry Pi: mini ordenador de pequeño tamaño, bajo coste y bajo consumo. Ejecutan sistemas operativos basados en Linux y están íntimamente relacionados con el Open Software [19].

Inteligencia artificial: conjunto de disciplinas de software, lógica, informática y filosofía que están destinadas a hacer que los PC realicen funciones que se pensaba que eran exclusivamente humanas, como percibir el significado en el lenguaje escrito o hablado, aprender, reconocer expresiones faciales, etc [20].

Dataset: es una representación residente en memoria de los datos que proporciona un modelo de programación relacional coherente independientemente de la fuente de los datos que contiene [21].

Visión artificial: se basan en sensores digitales protegidos dentro de cámaras industriales con ópticas especializadas en adquirir imágenes, para que el hardware y el software puedan procesar, analizar y medir diferentes características para tomar decisiones [22].

Captura de movimiento: es el proceso de grabar los movimientos de un actor y recrearlos en modelos de personajes digitales [23].

Módulo Edge TPU: es una pequeña ASIC que se ha diseñado para proporcionar un gran rendimiento en aplicaciones destinadas al aprendizaje automático (Machine Learning) [15].

TensorFlow Lite: es una herramienta de Google, de código abierto, para hacer computación en paralelo, implementando además redes neuronales, así como otros métodos de aprendizaje en la Inteligencia Artificial. Ahora la nueva versión “light” permite inferencia de baja latencia y modelos de aprendizaje de máquina [24].

COCO: es un conjunto de datos de detección, segmentación y subtitulación de objetos a gran escala [25].

1.8. Metodología

1.8.1. Línea de investigación

Las líneas de investigación de inteligencia artificial y, de automatización y robótica, de la Universidad del Sinú brindan las herramientas necesarias para el desarrollo y avances tecnológicos a la optimización de procesos y generación de nuevos conocimientos. Por las razones anteriores, el desarrollo del sistema básico de inferencia de inteligencia artificial que se desea implementar estará apoyado en las líneas mencionadas, cuyo propósito es la implementación de una TPU GOOGLE CORAL Y una Raspberry Pi, apoyado en la tecnología de visión artificial, por medio del uso sistemático del conocimiento y la investigación [26].

Cabe mencionar que este proyecto tiene un impacto positivo a la Universidad del Sinú, afianzar los conocimientos en la disciplina de visión artificial, la cual permite la automatización del proceso de obtención de información de las propiedades físicas de objetos, a partir del análisis de imágenes captadas por cámara.

1.8.2. Tipo de investigación

El desarrollo de este trabajo acumula todas las características necesarias de una investigación de tipo aplicada, ya que serán utilizadas técnicas de procesamiento de imágenes y conocimientos basados en sistema de visión artificial que han sido adquiridos durante el estudio del programa de Ingeniería de Sistemas de la Universidad del Sinú.

El resultado esperado es un sistema básico de inferencia de inteligencia artificial basado en la implementación de una TPU GOOGLE CORAL Y una Raspberry Pi, la cual nos lleva a indagar sobre temas y tecnologías que actualmente facilitan el estudio de reconocimiento de objetos en diversos campos como la medicina, el deporte, animaciones, entretenimiento, entre otras.

1.8.3. Metodología

Al tratarse de un trabajo eminentemente constituido por algoritmos implementados en Software, se han estudiado desde un primer momento las diferentes metodologías orientadas al desarrollo de Software para que el tiempo invertido sea aprovechado de la mejor forma posible. En la actualidad existen numerosas metodologías de desarrollo de Software entre las que destacan: desarrollo incremental, prototipado, en cascada, en espiral, ágil, entre otras [27].

Pero es ésta última, la metodología ágil, permite un desarrollo del software en paralelo, con un objetivo común a todas las vías, pudiendo dividir el trabajo en equipos. A pesar de que en este caso el desarrollo no lo producen diferentes equipos, esta metodología es muy útil para estructurar el trabajo según diferentes funcionalidades independientes entre sí, que pueden ser desarrolladas, implementadas y testeadas individualmente, y, más tarde, en conjunto.

Sin embargo, es necesario concretar qué modalidad de este método se va a seguir, puesto que existen diferentes modos de actuar en función de las características deseadas, los recursos, el tipo de proyecto, etc.

En este caso se ha seleccionado la modalidad de **Extreme Programming** (del inglés: Programación Extrema) que reúne las siguientes características [27]:

- Desarrollo iterativo e incremental: realizar pequeños avances y mejoras sobre la versión anterior para ir aumentando las funcionalidades.
- Testing unitario continuo: realizadas frecuentemente para detectar los errores cuanto antes, incluyendo también pruebas de regresión, es decir, pruebas de software cuyo objetivo es encontrar errores que se produzcan al introducir modificaciones en versiones de código que ya habían sido probadas.

- Simplicidad en el código: Un programa simple es más fácil de analizar, revisar y corregir. Esta metodología apuesta por el desarrollo de código más simple e invertir un poco de trabajo extra si es necesario cambiarlo, en lugar de realizar un código más complejo y posiblemente no utilizarlo. Además, como se explicó anteriormente en el apartado 1.1 Presentación, la necesidad de desarrollar unos algoritmos sencillos y eficientes es primordial para obtener un programa eficaz y que cumpla con los objetivos fijados, sin consumir recursos innecesarios del procesador.

Las etapas que integran esta metodología y que se han llevado a cabo son las que se definen en la figura 3:

Figura 3 Etapas de la metodología XP, Fuente [16]

- **Planificación inicial:** Es la primera etapa del proyecto y en la que se realiza la planificación global y la división del desarrollo en pequeñas funcionalidades diferenciadas, es decir, cada uno de los objetivos descritos. Se definen las fechas límite, la dedicación y horas a invertir, y

criterios de evaluación y seguimiento para verificar el cumplimiento de los objetivos.

- **Planificación intermedia:** En cada ciclo de desarrollo de una funcionalidad se establece una planificación individual personalizada para ese fragmento del programa. Se definen los mismos elementos que en la planificación inicial, pero ajustados a este pequeño "proyecto", para cada uno de los objetivos se definen las tareas que permiten el cumplimiento de estos.

- **Diseño:** Se trata de una etapa principalmente teórica. Consiste en la búsqueda de información, análisis de las diferentes formas de llevar a cabo el trabajo, diseño de la arquitectura a desarrollar en forma de flujogramas que sirvan como guía, etc. Se realiza la investigación pertinente que permiten alcanzar los conocimientos necesarios para cumplimiento de las actividades, como son manuales de instalación de librerías, funcionamiento de algoritmos y puesta en marcha de los dispositivos (TPU y Raspberry Pi).

- **Programación:** Es la etapa que conlleva más tiempo ya que comprende la escritura del programa, algoritmos de reconocimiento, la implementación de los algoritmos de pruebas de TensorFlow Lite con y sin TPU, siguiendo los requisitos de simplicidad, eficiencia y eficacia, cumpliendo los objetivos establecidos de forma óptima.

- **Testing:** Dentro del testing y la validación de resultados existen dos acciones. La primera acción es el testing unitario, que se lleva a cabo junto con la etapa de programación asegurando que el código no contenga errores. La segunda acción comprende el test de aceptación o validación de los resultados, que consiste en verificar si el programa

desarrollado cumple los objetivos que se definieron en la planificación. Si los resultados no son los esperados es necesario repetir el ciclo, realizando de nuevo la planificación intermedia. Si, por el contrario, los resultados cumplen los objetivos definidos, se publica esa funcionalidad que estará lista para ser implementada posteriormente en el programa final.

CRONOGRAMA

El tiempo estimado para la formulación del proyecto y diseño del sistema están distribuidos de la siguiente manera de acuerdo al diagrama de Gantt.

Tabla 2 Cronograma, Fuente Autor

ACTIVIDAD/SEMANA	1	2	3	4	5	6	7	8	9	10
Revisión del estado del arte y requerimientos del proyecto.	X	X	X							
Análisis del funcionamiento de la USB TPU Google CORAL		X	X	X	X					
Implementación de un sistema de inferencia a partir de la USB TPU Google CORAL. para la aceleración de sistemas de IA.				X	X	X	X	X	X	
Verificación y pruebas del sistema								X	X	
Elaboración documento final		X	X	X	X	X	X	X	X	X

2. REQUISITOS DEL SISTEMA

En este capítulo se hace el levantamiento de los requerimientos de la arquitectura basado en la norma **IEEE-830** [28] que facilita la descripción general del sistema, con el fin de conocer las principales funciones que éste debe realizar, los datos asociados que afectan al desarrollo, sin entrar en excesivos detalles.

Para especificar los requisitos funcionales y no funcionales se utilizarán las siguientes plantillas descritas en la Tabla 3.

Tabla 3 Plantilla de Requerimientos funcionales, Fuente Autor

Identificador:	*Entidad-Número Requerimiento*
Nombre del requerimiento:	*Nombre del requerimiento*
Actor:	*Usuarios involucrados en el requerimiento*
Indispensable/ deseable:	*Un Requerimiento No Funcional es Indispensable cuando es necesario tenerlo en cuenta para la realización de algún requerimiento funcional.* *Un Requerimiento No Funcional es Deseable cuando la funcionalidad que proporciona puede ser implementada de forma opcional, y ningún requerimiento funcional depende de ella. Ej.: Indispensable*
Prioridad:	*Nivel de prioridad del requerimiento (Alta, media, baja)*
Visible:	*Si será visto por el usuario*
Autor:	*Autor del sistema*
Fecha de elaboración:	*Fecha elaboración del sistema*
Revisión:	*Tutor metodológico*
Última revisión:	*Última revisión*
Resumen:	*Descripción del requerimiento*
Eventos:	*Eventos que se manifiestan en el desarrollo del requerimiento*
Precondiciones:	*Estado en el cual el sistema debe encontrarse antes de que la funcionalidad que expresa el requerimiento se lleve a cabo.*
Post-condiciones:	*Cosas que deben ocurrir después*

2.1. Definición de requisitos

Los requisitos de la arquitectura fueron definidos acordes a las necesidades básicas funcionales de la misma, habiendo realizado la selección de las técnicas de visión artificial más apropiadas para alcanzar los objetivos del proyecto. Por lo cual se definieron los siguientes requisitos, los cuales serán relacionados en la tabla 4:

- Cuantificar los FPS (Fotogramas por Segundo) en la Raspberry sin la TPU
- Cuantificar los FPS en la Raspberry con la TPU
- Optimizar la capacidad de cómputo de la Raspberry Pi.
- Probar algoritmos predeterminados en la TPU.

2.2. Requisitos funcionales

A continuación se representa el análisis de requerimientos necesarios para el desarrollo de la plataforma de la arquitectura, los cuales serán descritos en la sección de **Anexo E**:

Tabla 4 Requisitos Funcionales, Fuente Autor

TABLA DE REQUISITOS FUNCIONALES	
RF-01	Cuantificar los FPS (fotogramas por segundo) en la Raspberry sin la TPU
RF-02	Cuantificar los FPS en la Raspberry con la TPU
RF-03	Optimizar la capacidad de cómputo de la Raspberry Pi.
RF-04	Probar algoritmos predeterminado en la TPU.

2.3. Requisitos no funcionales

Dentro de los requisitos que satisface el sistema, este cuenta con requisitos de seguridad, rendimiento, portabilidad, usabilidad y escalabilidad, a fin de proporcionar una arquitectura amigable, razonable y práctico, los cuales serán descritos en la sección de **Anexo F**:

Tabla 5 Requisitos No Funcionales, Fuente Autor

TABLA DE REQUISITOS NO FUNCIONALES	
N° REQUISITO	**NOMBRE DEL REQUISITO**
RNF-01	Entrenamiento óptimo de modelos.

3. DISEÑO DE LA ARQUITECTURA

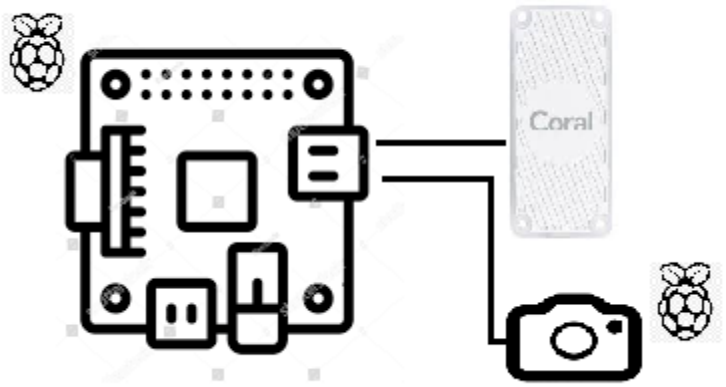
En este capítulo se explica el diseño de la arquitectura propuesta para la realización de este proyecto. Se hará una breve explicación del propósito de la arquitectura, las partes que la componen, así como las diversas etapas que desarrolla cada componente para procesar los datos requeridos.

3.1. Descripción general

La estructura general del sistema se muestra en la figura 4, estará basada en un dispositivo final (Raspberry Pi) que será el encargado de representar el resultado de los datos coprocesados por la TPU GOOGLE CORAL, la cual estará conectado a este, los datos serán capturados desde una cámara Raspberry Pi.

La TPU GOOGLE CORAL es la encargada de acelerar la inferencia de los modelos de aprendizaje automático.

Figura 4 Arquitectura del Sistema, Fuente Autor



3.2. Análisis de componentes

La implementación del sistema de inferencia utiliza los siguientes componentes descritos, los cuales se tienen en cuenta sus características principales y el costo de estos según los recursos estipulados para el desarrollo de este proyecto.

3.2.1. Raspberry Pi Model B V1.2.

El Raspberry instalado (Ver figura 5) es el encargado de procesar y enviar los datos, en esta pequeña placa se busca que el ordenador final realice menos esfuerzos en la entrega de los resultados. Sus características son similares a las de un pequeño PC, ofreciendo un rendimiento similar que estos, en algunas operaciones, pero por su bajo costo podemos tener una arquitectura distribuida y más amplia, y así tener varios nodos con capacidad de procesamiento para la captura del movimiento.

Figura 5 Raspberry Pi 3, Fuente Autor

3.2.2. Cámara

Para satisfacer las necesidades del proyecto, teniendo en cuenta la calidad de imagen, velocidad de captura y reducir costos del mismo, se analizan las tres posibles opciones que nos facilita el uso de la Raspberry Pi y se mencionan a continuación.

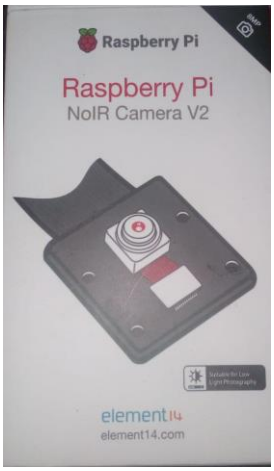
- Cámara USB: esta opción llega a ser una de las más atractivas, encontramos infinidades de ellas en el mercado desde costos bajos a los más altos, permitiendo conectar y transmitir la captura de imágenes y vídeos en la Raspberry Pi por medio de uno de sus puertos USB. Cabe resaltar que su calidad y capacidad de configuración de su módulo puede ser muy básica.

- Cámara por IP: Mediante este método, la cámara subirá una serie de fotos cuando detecte movimiento directamente al servidor ftp que este montado en una Raspberry Pi. Es decir, con este método tenemos la ventaja principal en conectividad, se evitan utilizar cables para las conexiones entre los equipos, pero puede presentar problemas de latencia, ya que es muy dependiente de los equipos conectados a la red.

- Cámara Raspberry Pi: La placa principal de la Raspberry Pi cuenta con un puerto digital dedicado para cámara. Se conecta en uno un pequeño socket que tiene la tarjeta Raspberry en su cara superior. Esta conexión, usa la interfaz dedicada CSI, la cual es óptima para conectar una cámara gracias a su alta capacidad de transmisión de datos. Existen diversos modelos de cámaras disponibles para utilizar. En este caso se realiza la prueba con un sensor óptico que tiene una resolución de 5 millones de pixeles (Mpx), y lleva un lente de foco fijo. La cámara es capaz de capturar imágenes de hasta 2592x1944 pixeles y soporta video en 1080p a 30 cuadros por segundo (FPS).

Por las características anteriormente expuestas y su facilidad de adquisición por bajos costo, se opta por usar **NoIR Camera V2** (Figura 6)**,** la cual facilita el desarrollo del proyecto.

Figura 6. NoIR Camera V2, Fuente Autor



3.2.3. TPU Google Coral

Coral USB Accelerator es un dispositivo USB que proporciona un Edge TPU como coprocesador para una computadora, acelerando la inferencia para los modelos de aprendizaje automático cuando está conectado a una computadora host Linux, Mac o Windows [29].

Figura 7 TPU GOOGLE CORAL, Fuente Autor

El módulo Coral USB Accelerator cuenta con un chip electrónico utilizado para agregar y acelerar funciones de aprendizaje automático en dispositivos SBC y PC cuando se realiza procesamiento local de inteligencia artificial, en este caso, brinda al Raspberry Pi la capacidad de mejorar su capacidad de cómputo al trabajar en entornos de aprendizaje automático. Se trata de un chip electrónico ASIC que está optimizado para ejecutar algoritmos de aprendizaje automático ligeros, una versión en miniatura del TPU refrigerado utilizado en los servidores en la nube de Google, se puede considerar este módulo como un coprocesador para inferencias en sistemas de cómputo de IA [30].

Figura 8 Chip Coral, Fuente [31]

Coral tiene la capacidad de ejecutar redes neuronales en el propio Raspberry PI, de forma que es posible incorporar, rápida y eficientemente, capacidades de inteligencia artificial en los proyectos sin que esto represente un peligro para la confidencialidad de los datos.

Para los desarrolladores, se dispone de una herramienta conocida como TensorFlow para crear redes neuronales y someterlas a procesos de aprendizaje, compilarlos y ejecutarlos en la tarjeta Edge TPU utilizando el software proporcionado. Una vez instalada la red compilada, todos los cálculos se realizan localmente en el circuito Edge TPU, sin enviar datos a la nube. Se elimina cualquier retraso en la nube, el rendimiento mejora y mantiene los datos del usuario localmente bajo control.

La plataforma de Coral también dispone de un conjunto de modelos preparados con pre-compilación y aprendizaje previo, que están debidamente optimizados para el chip electrónico Edge TPU. Estos son modelos flexibles que facilitan la programación y se adaptan a aplicaciones futuras.

Requisitos de uso de la TPU.

Para utilizar la TPU en una computadora requiere alguno de los siguientes sistemas operativos:

- Linux Debian 6.0 o superior, o cualquier derivado del mismo (como Ubuntu 10.0+), y una arquitectura de sistema x86-64 o ARM64 (se admite Raspberry Pi, pero solo hemos probado Raspberry Pi 3 Modelo B + y Raspberry Pi 4)
- MacOS 10.15, con MacPorts o Homebrew instalado
- Windows 10

Adicionalmente requiere:

- Un puerto USB disponible (para obtener el mejor rendimiento, un puerto USB 3.0)
- Python 3.5, 3.6 o 3.7

Ventajas de su uso.

A pesar de su pequeño tamaño y su reducido consumo, ofrece un gran rendimiento y permite desplegar alta precisión, entre sus ventajas se encuentran [32]:

- **Infraestructura completa:** complementa las TPU de Cloud y los servicios de Google Cloud para proporcionar una infraestructura de hardware y software completa, facilita el despliegue de las soluciones basadas en Inteligencia Artificial.

- **Alto rendimiento con un tamaño y consumo reducidos:** Gracias a su rendimiento, reducido tamaño y bajo consumo energético, Edge TPU posibilita el amplio despliegue de Inteligencia Artificial de alta calidad.

- **Diseño conjunto de hardware, software y algoritmos de Inteligencia Artificial:** Edge TPU no es una solución de hardware. Este circuito integrado combina hardware personalizado, software libre y algoritmos de Inteligencia Artificial vanguardistas para ofrecer soluciones específicas fáciles de desplegar y de alta calidad.

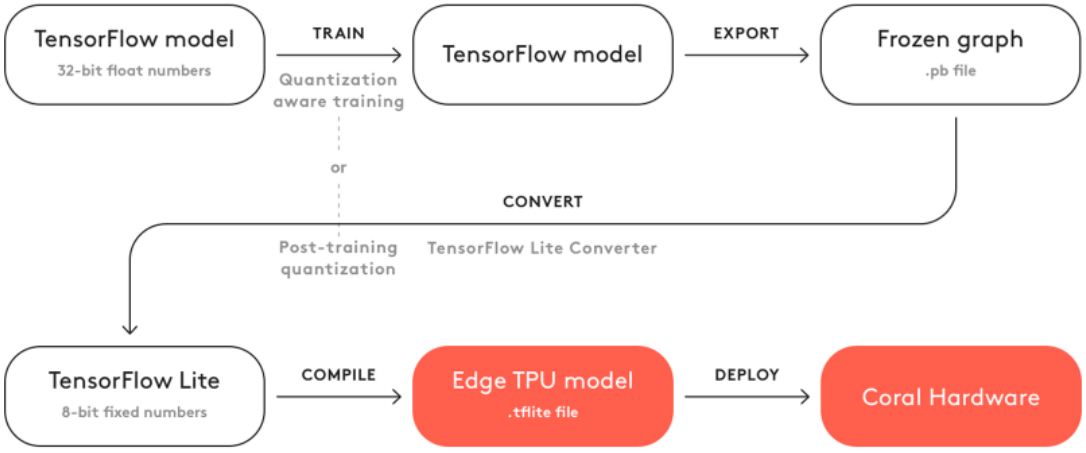
- **Amplia gama de aplicaciones:** El uso de Edge TPU se extiende a un número cada vez mayor de casos prácticos industriales, como el mantenimiento predictivo, la detección de anomalías, la visión artificial, la robótica o el reconocimiento de voz, entre muchos otros. Se puede utilizar en fábricas, operaciones on-premise, centros sanitarios, tiendas, espacios inteligentes, medios de transporte, etc.

3.3. Diseño de diagramas

3.3.1. Diagrama de flujo de trabajo de un modelo para TPU

Se ilustra el proceso básico para crear un modelo que sea compatible con Edge TPU. La mayor parte del flujo de trabajo utiliza herramientas estándar TensorFlow.

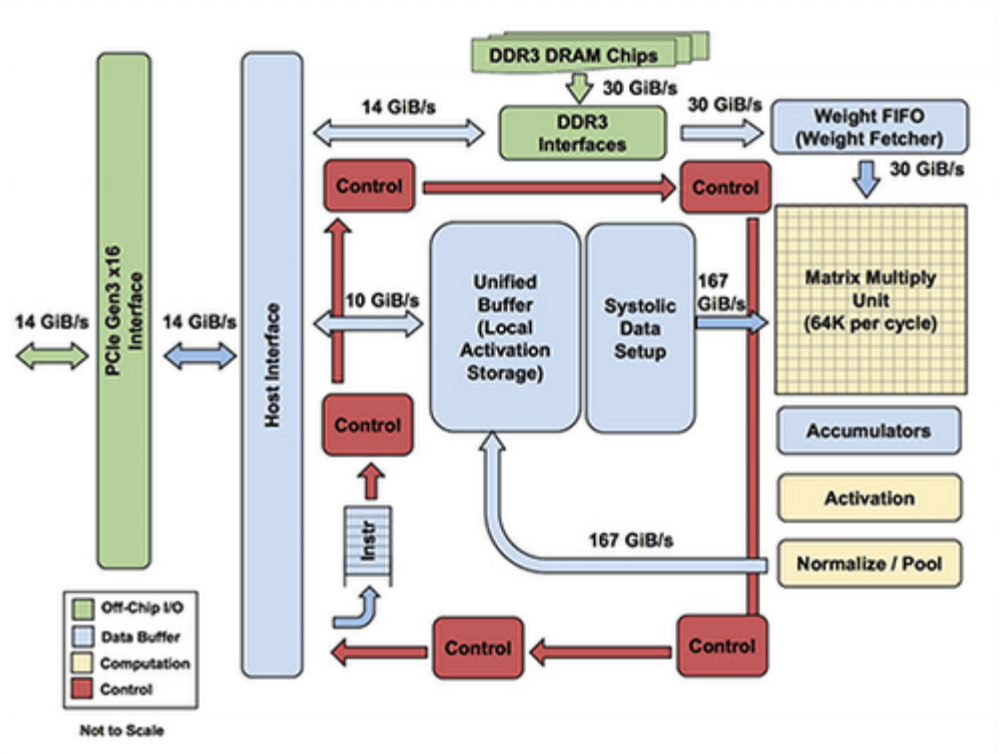
Figura 9 Diagrama de flujo de trabajo de un modelo, Fuente [33].



3.3.4. Diagrama del Esquema interno de TPU Google Cloud

En este diagrama se puede ver en detalle los componentes electrónicos de la TPU, diseñados para hacer exactamente las operaciones bit a bits necesarios.

Figura 10 Esquema Interno de TPU Coral, Fuente [34]



4. DESARROLLO

En este capítulo se realizan pruebas de rendimiento de una Raspberry Pi, con procesos de inferencia. Enfocados en tiempo de retrasos de la visualización de una imagen FPS (fotogramas por segundo) y menor carga de capacidad de cómputo.

En primer lugar, se describen las distintas líneas de desarrollo estudiadas para afrontar el problema, donde se van a determinar y analizar el uso de tecnologías que arrojan mejor resultado en base al seguimiento de las especificaciones de los requisitos plasmados en la recolección de requerimientos.

Posteriormente, se procede a describir las respectivas etapas de pruebas de la velocidad de fotogramas obtenida usando los siguientes modelos: primero un algoritmo básico en OpenCV para el reconocimiento de una persona partiendo de puntos esenciales como cabeza y hombros, segundo librerías de TensorFlow Lite y por último librerías de TensorFlow Lite con la TPU Coral.

4.3. Líneas de desarrollo estudiadas.

4.3.4. OpenCV

OpenCV es una biblioteca libre desarrollada originalmente por Intel. Escrita originalmente en C/C++, su mejor virtud es que es multiplataforma, se puede ejecutar en diferentes sistemas operativos (Linux, Windows, Mac OS X, Android e iOS). También se puede utilizar en diferentes lenguajes de programación como Java, Objective C, Python y C#. El principal objetivo de esta librería es ofrecer algoritmos de visión artificial que funcionen en tiempo real, en su mayoría basados en redes no neuronales [13].

Se centra principalmente hacia procesamiento de imagen en tiempo real como tal si encuentra las “Intel Integrated Performance Primitives” sobre el sistema, utilizará estas rutinas para acelerarse.

Contiene más de 500 funciones que abarcan una gran gama de áreas en el proceso de visión, como reconocimiento de objetos (reconocimiento facial), calibración de cámaras, visión estéreo y visión robótica.

4.3.5. TensorFlow

Es una librería matemática simbólica para operaciones de aprendizaje automático creada y desarrollada por Google, es una plataforma de código abierto con una potente tecnología de inteligencia artificial utilizada en el reconocimiento de imagen, de voz y en traducciones de idiomas. Esta aplicación puede ejecutarse en dispositivos móviles, así como en computadoras que no tienen procesadores gráficos dedicados.

El funcionamiento de TensorFlow es muy simple. Este software relaciona datos de la misma forma que lo hace un cerebro humano, relacionando imágenes con textos o reconociendo palabras. Los datos analizados por esta inteligencia artificial se almacenar para luego ser usados en programas y aplicaciones.

4.3.6. TensorFlow Lite

TensorFlow Lite es un marco optimizado para implementar modelos de aprendizaje profundo livianos en dispositivos SBC con recursos limitados (como en este caso, la Raspberry Pi). Los modelos TensorFlow Lite tienen un tiempo de inferencia más rápido y requieren menos potencia de procesamiento, por lo que puede usar para obtener un rendimiento más rápido en aplicaciones en tiempo real.

4.3.7. Selección de la línea de desarrollo.

La implementación de las líneas anteriores mencionadas da a lugar a un conjunto de prestaciones que puede ofrecer cada una de las tecnologías estudiadas. A la hora de escoger entre una u otra, se tiene en cuenta quien brinda un óptimo rendimiento en una Raspberry Pi.

En primer lugar, se estudia la cantidad de información y documentación que se tiene a la mano, ambas tecnologías cuentan con una comunidad muy activa al igual que numerosos tutoriales y ejemplos de desarrollo e implementación que facilitan el trabajo a expertos o aprendices en el tema de Inteligencia artificial.

En cuanto a la implementación de estas tecnologías en una Raspberry Pi, se analiza el tiempo de ejecución y capacidad de cómputo en esta. OpenCV es una librería completa que facilita los procesos de reconocimiento de rostros, TensorFlow fue diseñada para facilitar el desarrollo en dispositivos móviles, los cuales tienen menos recursos de cómputo. Por último, TensorFlow Lite tiene las mismas características de desarrollo que la anterior, pero esta es una versión más liviana por eso su nombre.

Finalmente, valorando los aspectos descritos anteriormente se opta por usar TensorFlow Lite como tecnología principal para desarrollar un sistema básico de inferencia de inteligencia artificial para la USB TPU mediante algoritmos básicos de prueba.

4.4. Configuración del entorno

Como se acaba de mencionar, la tecnología principal que se usa para desarrollar las pruebas es TensorFlow Lite (llamamos principal porque se realizaron algunas pruebas teniendo en cuenta OpenCV, como una muestra de su implementación). Para la puesta en marcha de las pruebas básicas, nuestro entorno de trabajo está

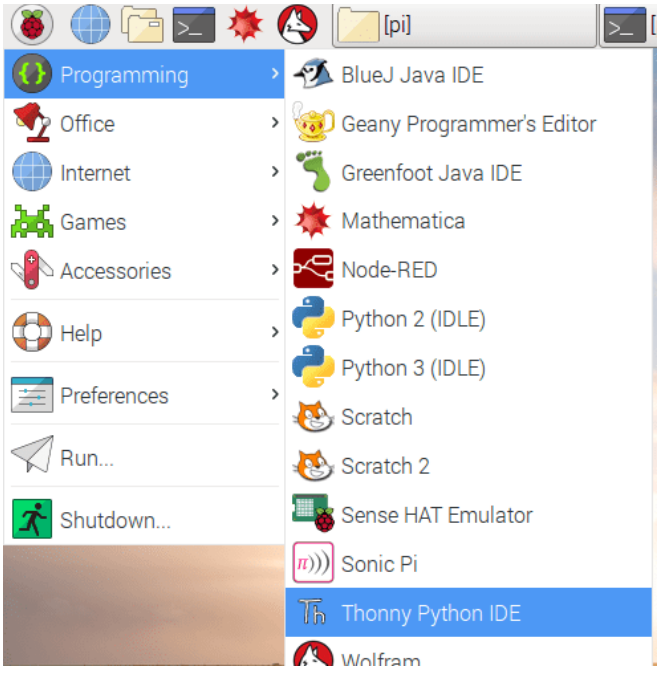
basado en Thonny como Entorno de Desarrollo Integrado - IDE (Integrated Development Envaronen) para ejecutar OpenCV y TensorFlow solo requiere ejecuciones en el modo de consola.

El sistema de inferencia se desarrolló en un entorno Linux, bajo el sistema operativo Raspbian (diseñado específicamente para administrar los diferentes modelos de Raspberry Pi) y un acelerador TPU Coral.

4.4.4. IDE Thonny

Thonny tiene diversas características básicas de Python. Por un lado, se puede destacar la inclusión de un depurador para ayudar al programador a corregir los errores y por otro ofrece acceso a la consola propia de Python. También se pueden destacar la visualización de variables, el inspector de objetos, AST y otras características, se lo podría considerar como una herramienta ideal para aquellos que quieran empezar a aprender a programar, ofreciendo además versiones para GNU/Linux, Windows y Mac [35].

Figura 11 Thonny Python IDE, Fuente Autor



4.4.5. OpenCV

Es necesario instalar la librería OpenCV en cada uno de los sistemas operativos, ya que esta será empleada en los procesos de reconocimiento de rostros. El proceso de instalación de esta librería es un tanto más complejo, en el apartado de implementación de los algoritmos de prueba se realiza una explicación detallada de cada procedimiento.

4.4.6. Python 3

El lenguaje de programación Python viene por defecto instalado en varios de los sistemas operativos conocidos, como son los casos de Windows y Raspbian, lo importante es verificar la versión que cada sistema tiene instalada. Para el desarrollo de este proyecto instalamos Python 3 [17].

4.4.7. TensorFlow Lite

Se puede instalar TensorFlow en dos modalidades distintas: modo CPU y modo GPU. El primero de ellos supone una instalación básica, en la que todas las ejecuciones se procesan directamente en la CPU. Por otro lado, el modo GPU da la posibilidad de acelerar los procesos realizados por TensorFlow, en el caso de disponer de un dispositivo con GPU dedicada. Como en la maquina utilizada (Raspberry Pi), se instaló el modo básico (CPU) y la optimización de los fotogramas está a cargo de la TPU Coral [24].

4.4.8. Acelerador TPU Coral

Es un dispositivo USB que proporciona un Edge TPU como coprocesador para la Raspberry Pi. Acelera la inferencia para sus modelos de aprendizaje automático cuando está conectado. Para el uso de esta fue necesario descargar el tiempo de

ejecución de Edge TPU y la biblioteca TensorFlow Lite en la computadora donde conectará el Acelerador USB, es decir la Raspberry Pi [36].

4.5. Implementación de algoritmos básicos de prueba.

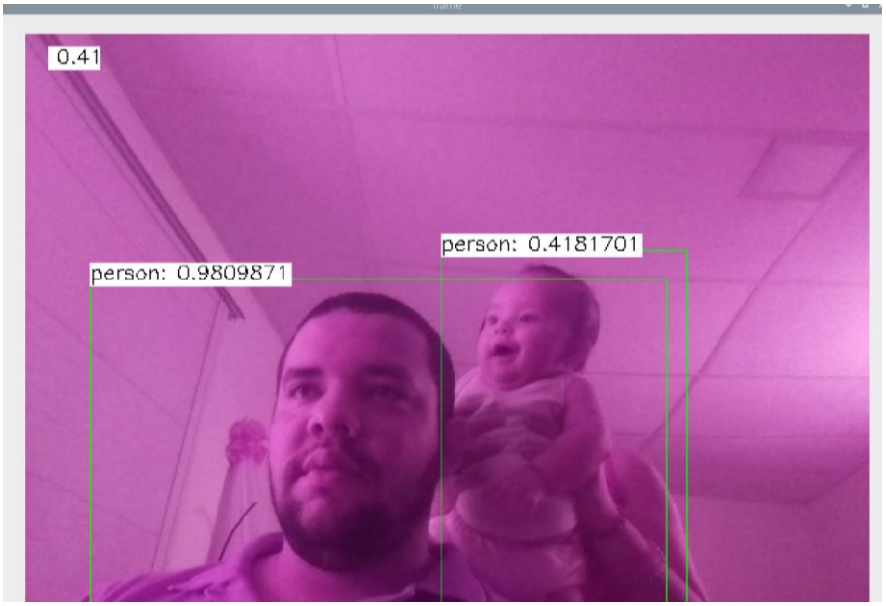
En este apartado se describe el paso a paso de los distintos tipos de algoritmos básicos de pruebas, desde la instalación de cada una de las herramientas, librerías y recursos que fueron fundamentales para el desarrollo de este proyecto.

4.5.4. Algoritmo en OpenCV

En el ANEXO D, se encuentra la parte de la instalación previa de OpenCV con sus ejecuciones y librerías para reconocimiento de imágenes. Este es el primer paso a tener en cuenta para la ejecución del algoritmo.

Con este algoritmo se tiene como objetivo el reconocimiento de un objeto. En el ANEXO A se explica en detalle el código empleado para este. En la siguiente figura, se observan los fotogramas por segundos que se alcanzaron con OpenCV.

Figura 12 Ejecución de algoritmo en OpenCV, Fuente Autor



4.5.5. TensorFlow Lite

Instrucciones sobre la configuración realizada con TensorFlow Lite en Raspberry Pi y su ejecución de modelos de detección de objetos. En el sistema operativo Raspbian. Los modelos TensorFlow Lite (TFLite) se ejecutan mucho más rápido que los modelos TensorFlow normales en la Raspberry Pi.

Estos son los pasos necesarios para configurar TensorFlow Lite:

- Se actualiza la Raspberry Pi.
- Se descarga el repositorio y se crea el entorno virtual [37].
- Se instala TensorFlow y OpenCV.
- Se configura el modelo de detección TensorFlow Lite.
- Por último, se ejecuta el modelo TensorFlow Lite.

Paso 1. Actualización de la Raspberry Pi.

La Raspberry se debe actualizar completamente, para esto en la terminal se procede con el siguiente comando:

```
sudo apt-get update
sudo apt-get dist-upgrade
```

Paso 2. Descargar repositorio para crear el entorno virtual.

Para este paso, nuestro entorno virtual es tomado de un repositorio existente [37].

```
git clone https://github.com/EdjeElectronics/TensorFlow-Lite-Object-Detection-on-Android-and-Raspberry-Pi.git
```

A continuación, se descarga una carpeta llamada TensorFlow-Lite-Object-Detection-on-Android-and-Raspberry-Pi. La cual se renombro a "tflite1":

```
mv TensorFlow-Lite-Object-Detection-on-Android-and-Raspberry-Pi tflite1
cd tflite1
```

En el siguiente paso se crea un entorno virtual llamado "tflite1-env". Se usa un entorno virtual para evitar cualquier conflicto entre versiones de las bibliotecas de paquetes que ya estén instaladas en la Raspberry Pi.

```
sudo pip3 install virtualenv
python3 -m venv tflite1-env
```

El paso anterior crea una carpeta llamada tflite1-env dentro del directorio tflite1. La carpeta tflite1-env contendrá todas las bibliotecas de paquetes para este entorno. A continuación, se activa el entorno:

```
source tflite1-env/bin/activate
```

Paso 3. Instalar las dependencias de TensorFlow Lite y OpenCV

A continuación, se instaló TensorFlow, OpenCV y todas las dependencias necesarias para ambos paquetes. OpenCV no es necesario para ejecutar TensorFlow Lite, pero los scripts de detección de objetos en este repositorio lo usan para capturar imágenes y dibujar resultados de detección en ellas.

Para facilitar las cosas, existe un script de shell (ANEXO B) que tiene automáticamente todos los paquetes y dependencias necesarios para la descarga y ejecución de estos:

```
bash get_pi_requirements.sh
```

Paso 4. Modelo de detección TensorFlow Lite

Google proporciona un modelo de Edge TPU de muestra, el cual se descargó con el comando:

```
wget
https://storage.googleapis.com/download.tensorflow.org/models/tflite/coco_ssd_mobilenet_v1_1.0_
quant_2018_06_29.zip
```

Se cambia la ubicación y se renombra el archivo:

unzip coco_ssd_mobilenet_v1_1.0_quant_2018_06_29.zip -d Sample_TFLite_model

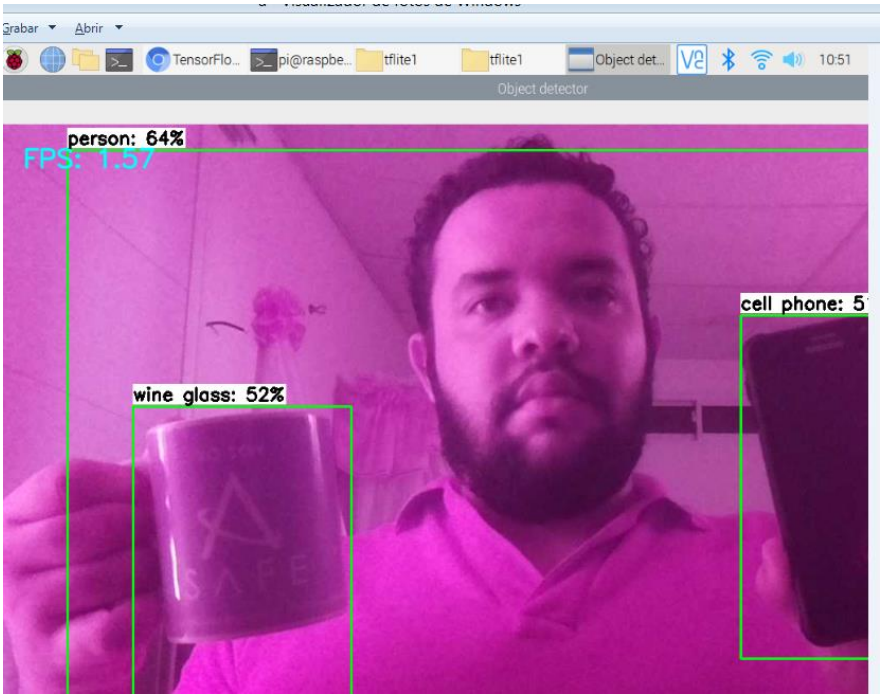
Paso 5. Ejecución del modelo TensorFlow Lite

Antes de empezar con este último paso, primero se debe liberar la memoria cerrando cualquier aplicación que no se esté utilizando. Además, verificar que la cámara de la Raspberry pi se encuentre conectada.

python3 TFLite_detection_webcam.py --modeldir=Sample_TFLite_model

Después de unos momentos de inicialización, aparece la ventana que muestra la alimentación de la cámara pi. Los objetos detectados muestran cuadros delimitadores y etiquetas en tiempo real (Ver figura 13).

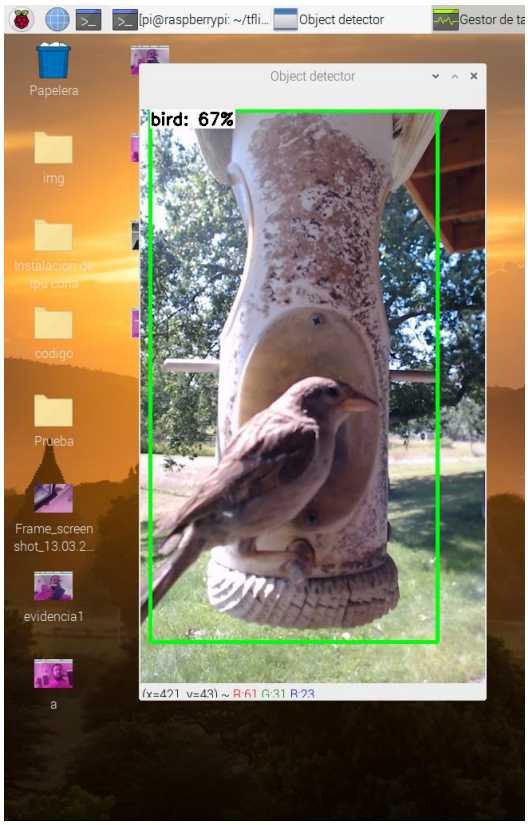
Figura 13 Ejecución TensorFlow Lite con cámara pi, Fuente Autor



Detección en video

python3 TFLite_detection_video.py --modeldir=Sample_TFLite_model

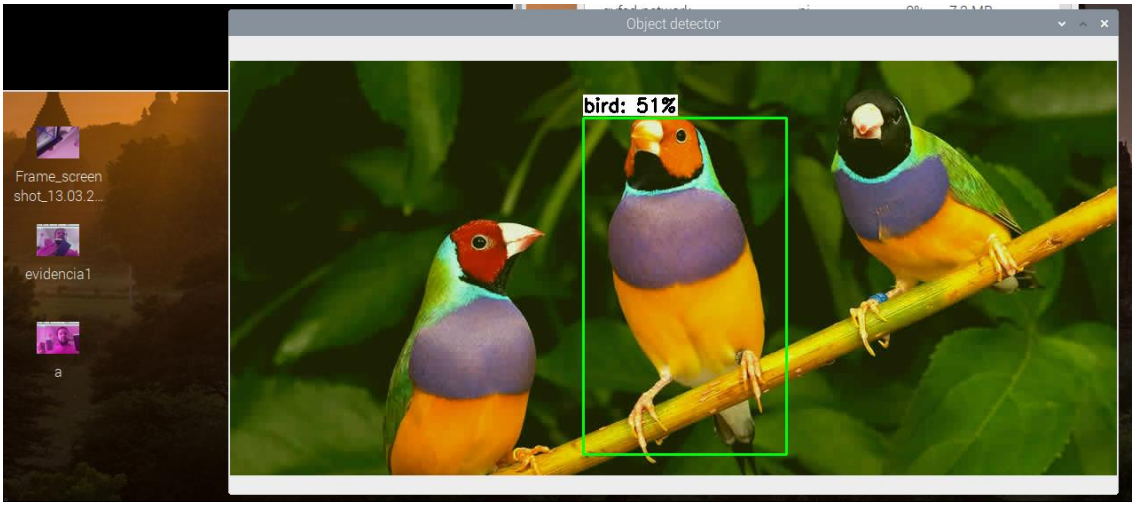
Figura 14 Ejecución TensorFlow Lite en video, Fuente Autor



Detección en imagen:

python3 TFLite_detection_image.py --modeldir=Sample_TFLite_model

Figura 15 Ejecución TensorFlow Lite en una imagen, Fuente Autor



4.5.6. TensorFlow Lite con TPU Coral

El Coral USB Acelerador es un accesorio de hardware USB para acelerar los modelos TensorFlow. Hace que los modelos de detección de objetos funcionen mucho más rápidos y es fácil de configurar. Estos son los pasos adicionales a los anteriores que se realizaron para configurar el Acelerador USB Coral:

- Instalar la biblioteca libedgetpu
- Configurar el modelo de detección Edge TPU
- ¡Ejecutar la detección con Edge TPU!

Paso 1. Instalar la biblioteca libedgetpu

Primero se activa el entorno virtual con los comandos:

```
cd /home/pi/tflite1
source tflite1-env/bin/activate
```

Agregar el repositorio de paquetes de Coral a la lista de distribución apt-get emitiendo el siguiente comando:

```
echo "deb https://packages.cloud.google.com/apt coral-edgetpu-stable main" | sudo tee
/etc/apt/sources.list.d/coral-edgetpu.list
curl https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -
sudo apt-get update
```

Con el siguiente comando se instala la biblioteca:

```
sudo apt-get install libedgetpu1-std
```

Paso 2. Configurar el modelo de detección Edge TPU

Los modelos Edge TPU son modelos TensorFlow Lite que se han compilado específicamente para ejecutarse en dispositivos Edge TPU como el Acelerador USB Coral. Residen en un archivo tflite y se usan de la misma manera que un modelo TF Lite normal.

Google proporciona un modelo de Edge TPU de muestra, el cual se descargó con el comando:

```
wget
https://dl.google.com/coral/canned_models/mobilenet_ssd_v2_coco_quant_postprocess_edgetpu.tf
lite
```

Se cambia la ubicación y se renombra el archivo:

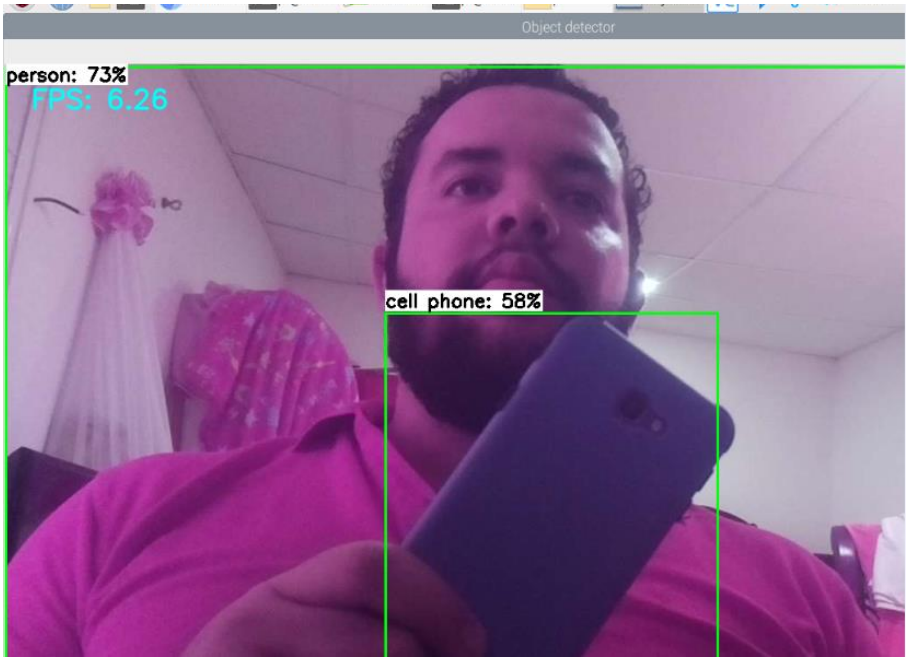
```
mv mobilenet_ssd_v2_coco_quant_postprocess_edgetpu.tflite Sample_TFLite_model/edgetpu.tflite
```

Paso 3. Ejecutar la detección con Edge TPU

Detección de cámara pi, Figura 16.

python3 TFLite_detection_webcam.py --modeldir=Sample_TFLite_model --edgetpu

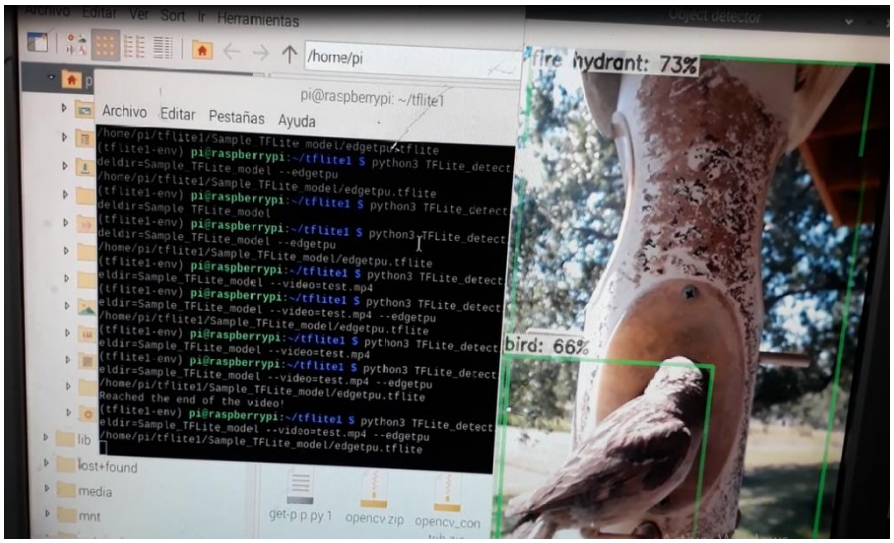
Figura 16 Detección de cámara con TPU, Fuente Autor



Detección en video, figura17:

python3 TFLite_detection_video.py --modeldir=Sample_TFLite_model --edgetpu

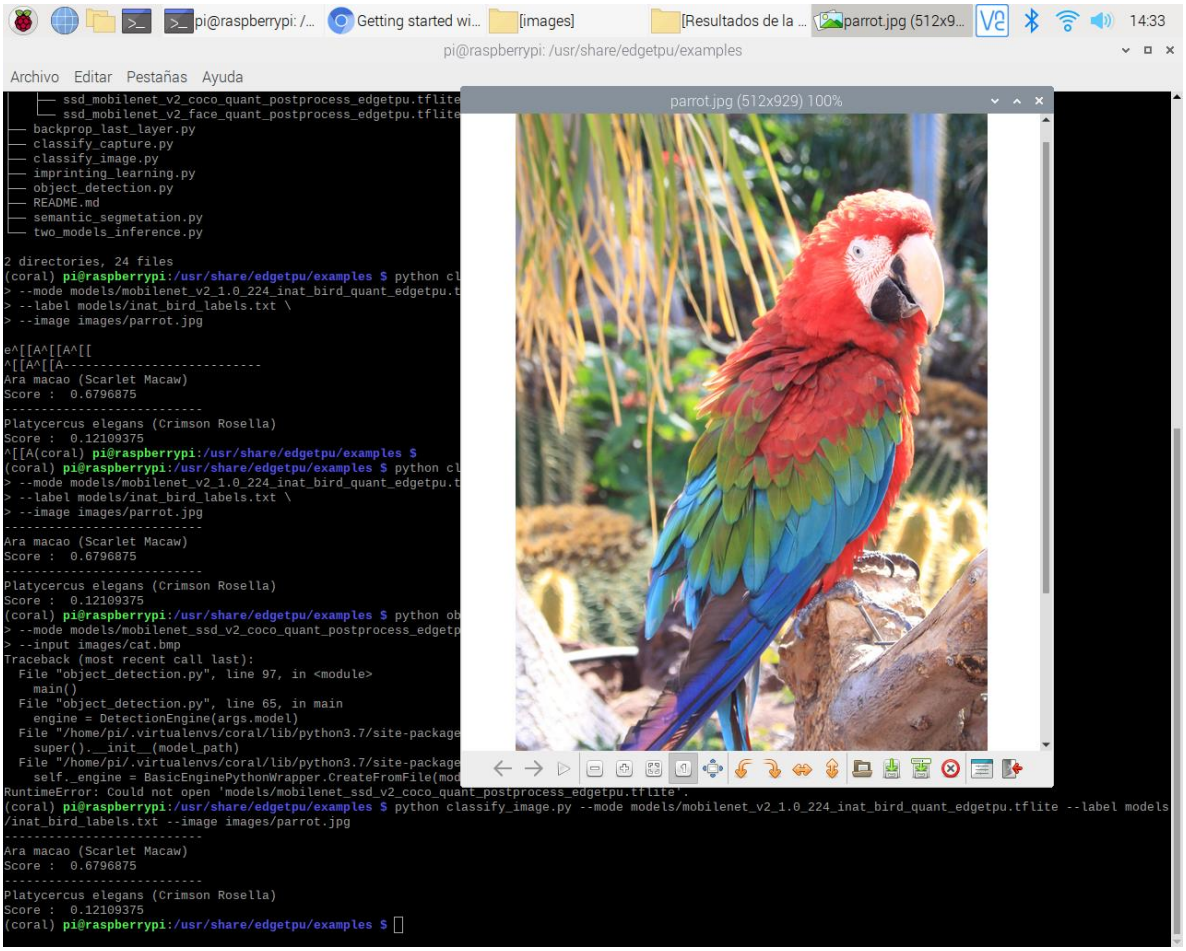
Figura 17 Detección en video, Fuente Autor



Detección en imagen, figura 18:

python3 TFLite_detection_image.py --modeldir=Sample_TFLite_model --edgetpu

Figura 18 Detección en imagen con TPU, Fuente Autor



4.5.7. Algoritmos de prueba de la TPU Coral.

En este espacio se observa la puesta en marcha de los modelos de aprendizaje pre-entrenados en la TPU Coral, a continuación, se detallan los pasos a seguir para su ejecución:

Paso 1. Instalar paquetes de ejemplos de EdgeTPU

$ sudo apt-get install edgetpu-examples

Paso 2. Agregar permisos de escritura al directorio de ejemplos

$ sudo chmod a + w / usr / share / edgetpu / examples

Paso 3. Estructura del proyecto: Activar el entorno y cambiar al directorio de ejemplos.

$ workon coral
$ cd / usr / share / edgetpu / examples

El directorio de ejemplos contiene directorios para imágenes y modelos junto con una selección de scripts de Python.

Directorio de imágenes:

- bird.bmp
- cat.bmp
- grace_hopper.bmp
- parrot.jpg
- sunflower.bmp

Directorio de modelos:

- coco_labels.txt
- deeplabv3_mnv2_pascal_quant_edgetpu.tflite

- inat_bird_labels.txt
- mobilenet_ssd_v1_coco_quant_postprocess_edgetpu.tflite
- mobilenet_ssd_v2_coco_quant_postprocess_edgetpu.tflite
- mobilenet_ssd_v2_face_quant_postprocess_edgetpu.tflite
- mobilenet_v2_1.0_224_inat_bird_quant_edgetpu.tflite

Directorio de scripts:

- backprop_last_layer.py
- classify_capture.py
- classify_image.py
- imprinting_learning.py
- object_detection.py
- semantic_segmetation.py
- two_models_inference.py

Para análisis de estos, se utilizan tres modelos de TensorFlow Lite:

1. Modelo de clasificación de imagen: este se usó con el script de clasificación de Python **classify_image.py**

   mobilenet_v2_1.0_224_inat_bird_quant_edgetpu.tflite

2. Modelo de detección de rostros: este se usó con el script de detección de objetos de Python **object_detection.py**

   mobilenet_ssd_v2_face_quant_postprocess_edgetpu.tflite

3. Modelo de detección de objetos: este se usó con el script de detección de objetos de Python **object_detection.py**

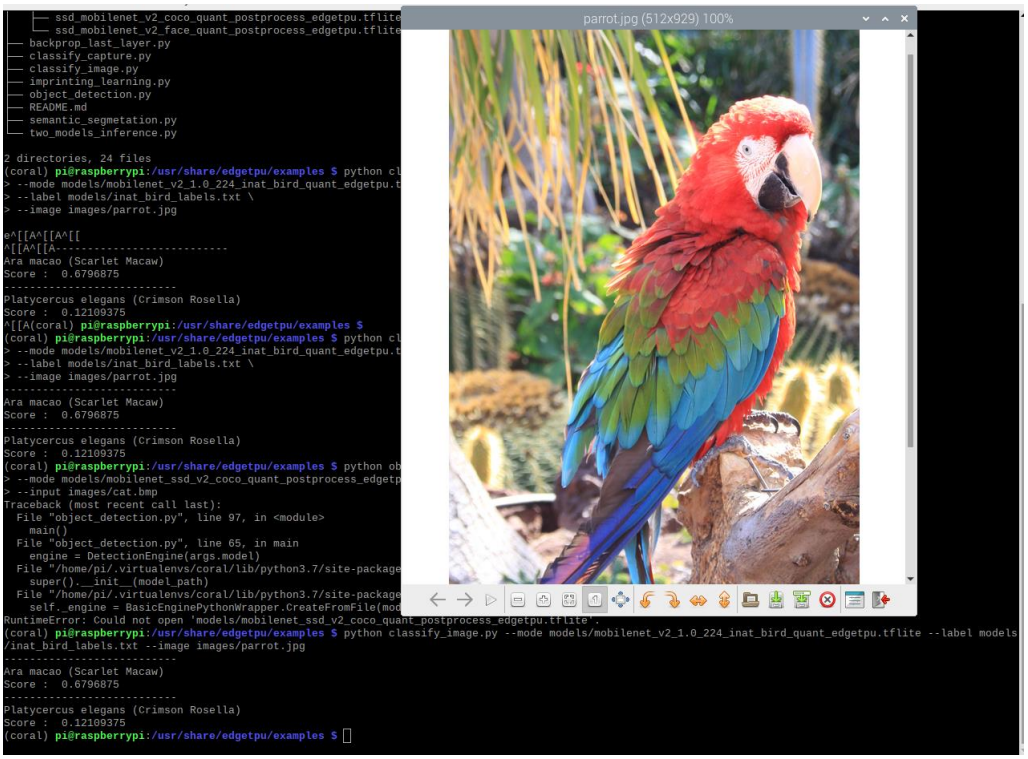
   mobilenet_ssd_v2_coco_quant_postprocess_edgetpu.tflite

Clasificación de imágenes

En este punto para realizar nuestro primer ejemplo, en la consola se ejecuta la siguiente instrucción:

```
$ python classify_image.py \
--mode models/mobilenet_v2_1.0_224_inat_bird_quant_edgetpu.tflite \
--label models/inat_bird_labels.txt \
--image images/parrot.jpg
```

Figura 19 Reconocimiento de imagen, Fuente Autor



Los resultados obtenidos en la ejecución (Figura 19)

Ara macao (Scarlet Macaw)

Score: 0.6796875

Platycercus elegans (Crimson Rosella)

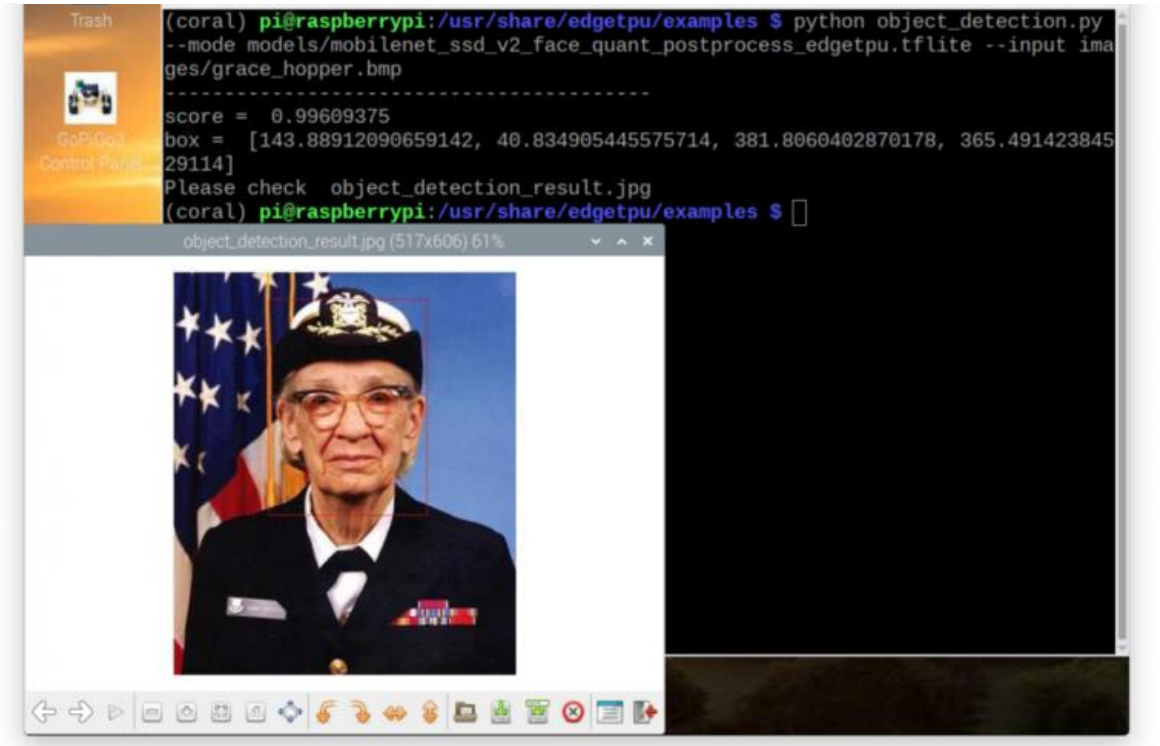
Score: 0.12109375

Detección de rostros

Ahora se realiza la detección de rostros con el Acelerador USB de Google Coral:

```
 $ python object_detection.py \
--mode models/mobilenet_ssd_v2_face_quant_postprocess_edgetpu.tflite \
--input images/grace_hopper.bmp
```

Figura 20 Detección de rostro, Fuente Autor



Los resultados obtenidos en la ejecución (Figura 20)

score = 0.99609375

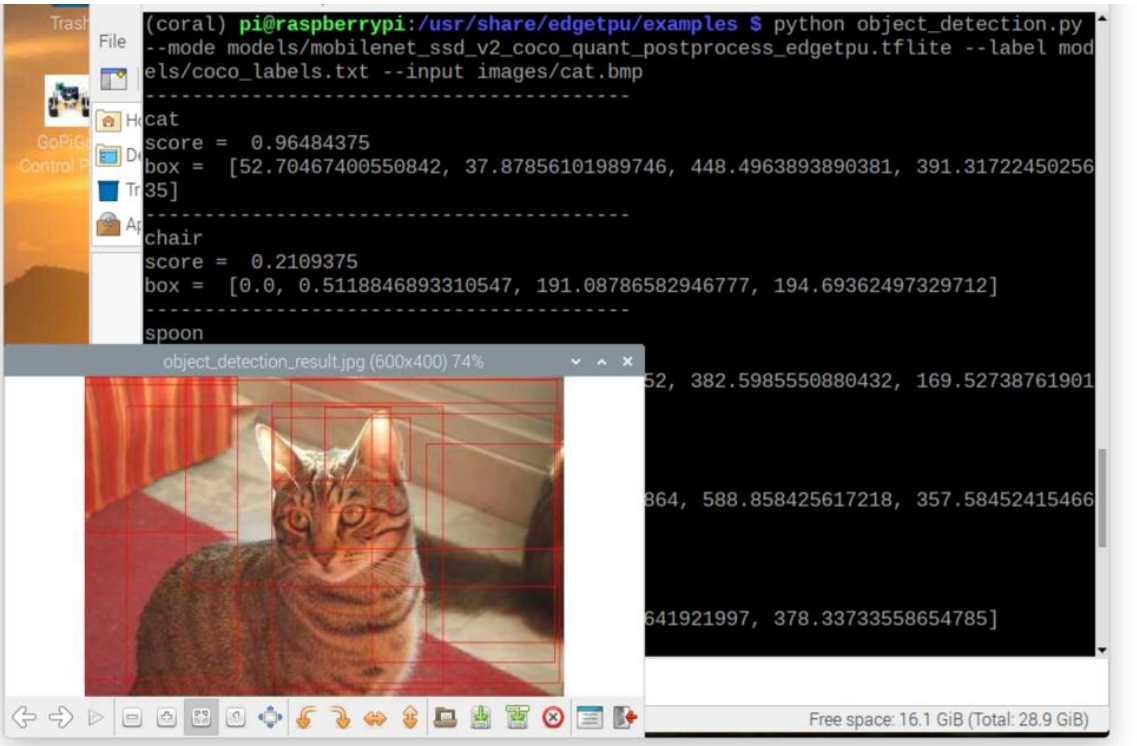
box = [143.88912090659142, 40.834905445575714, 381.8060402870178, 365.49142384529114]

Detección de objetos++

En el siguiente ejemplo se realizó la detección de objetos usando un modelo entrenado en el conjunto de datos COCO:

$ python object_detection.py \
--mode models/mobilenet_ssd_v2_coco_quant_postprocess_edgetpu.tflite \
--input images/cat.bmp

Figura 21 Detección de objeto, Fuente Autor



Los resultados obtenidos en la ejecución (Figura 21):

Cat:

score = 0.96484375

box = [52.70467400550842, 37.87856101989746, 448.4963893890381, 391.3172245025635]

Chair:

score = 0.2109375

box = [0.0, 0.5118846893310547, 191.08786582946777, 194.69362497329712]

5. PRUEBAS Y RESULTADOS

En este capítulo se comentan las pruebas realizadas que validan a este proyecto como resultado al problema definido previamente en el capítulo 1, y finalmente se comentan los resultados alcanzados durante su desarrollo.

5.3. Pruebas

Para cumplir con los objetivos planteados para el desarrollo de este proyecto se tienen en cuenta tres tipos de pruebas, los cuales garantizan el cumplimiento de los mismos:

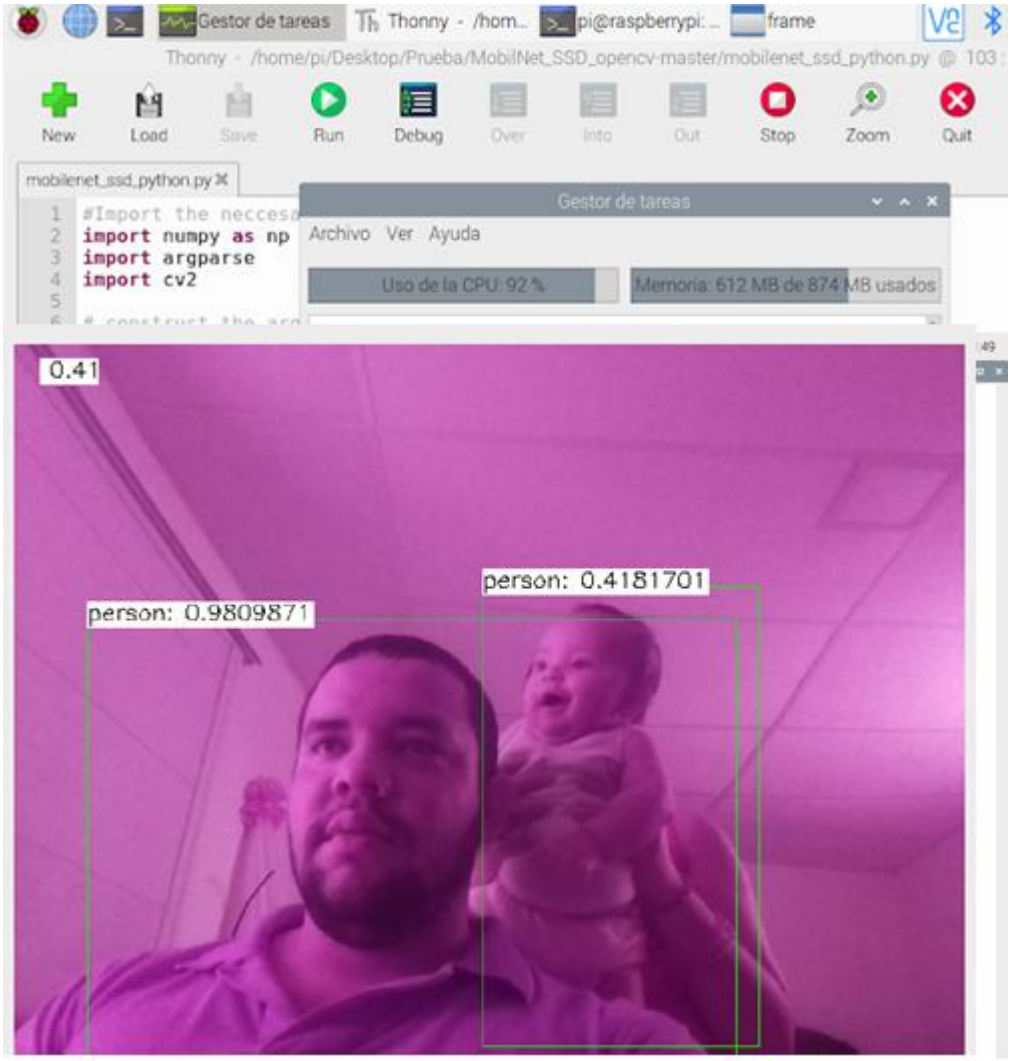
1) Algoritmo OpenCV:

El objetivo de esta prueba es demostrar el funcionamiento de una librería robusta ante un sistema con pocos recursos, esto conlleva a que la maquina se esfuerce en su capacidad de computo, dando a probar que no por ser la más robusta es la indicada para todo tipo de proyecto como es el caso con el uso de la Raspberry Pi.

Cabe resaltar que este algoritmo no es un objetivo del proyecto, pero se convierte en un punto importante a la hora de demostrar las mejoras que se obtienen con la tecnología TensorFlow Lite y esta a su vez combinada con la TPU Coral.

Este algoritmo parte del reconocimiento de objetos por medio de la cámara Raspberry Pi, para esta prueba se quiere reconocer a una persona adulta y un bebé. Al realizar la ejecución del algoritmo se obtiene un porcentaje de inferencia respectivamente los FPS y el rendimiento de la CPU (ver Figura 21).

Figura 22 Reconocimiento de una Persona, Fuente Autor

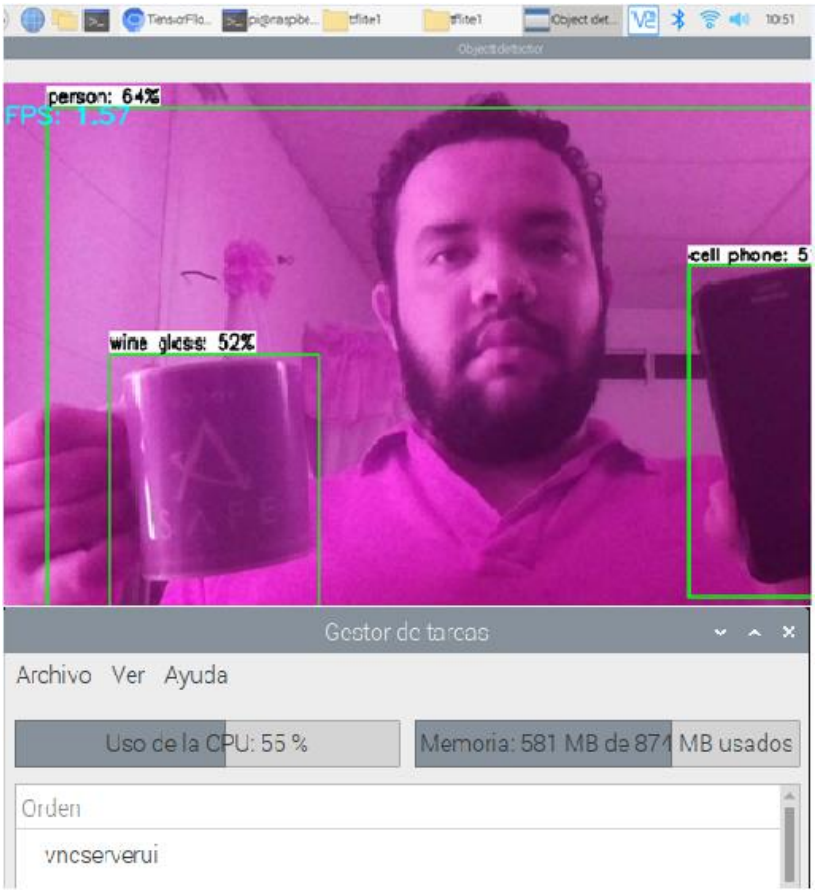


2) TensorFlow Lite

Se definen tres tipos de pruebas en esta tecnología, basados en los modelos de detección en una imagen, detección por medio de una cámara y por ultimo detección en un video. En esta sección serán realizadas las pruebas sin el uso de la TPU Coral, para poder realizar el análisis de rendimiento e inferencia con el uso de la TPU.

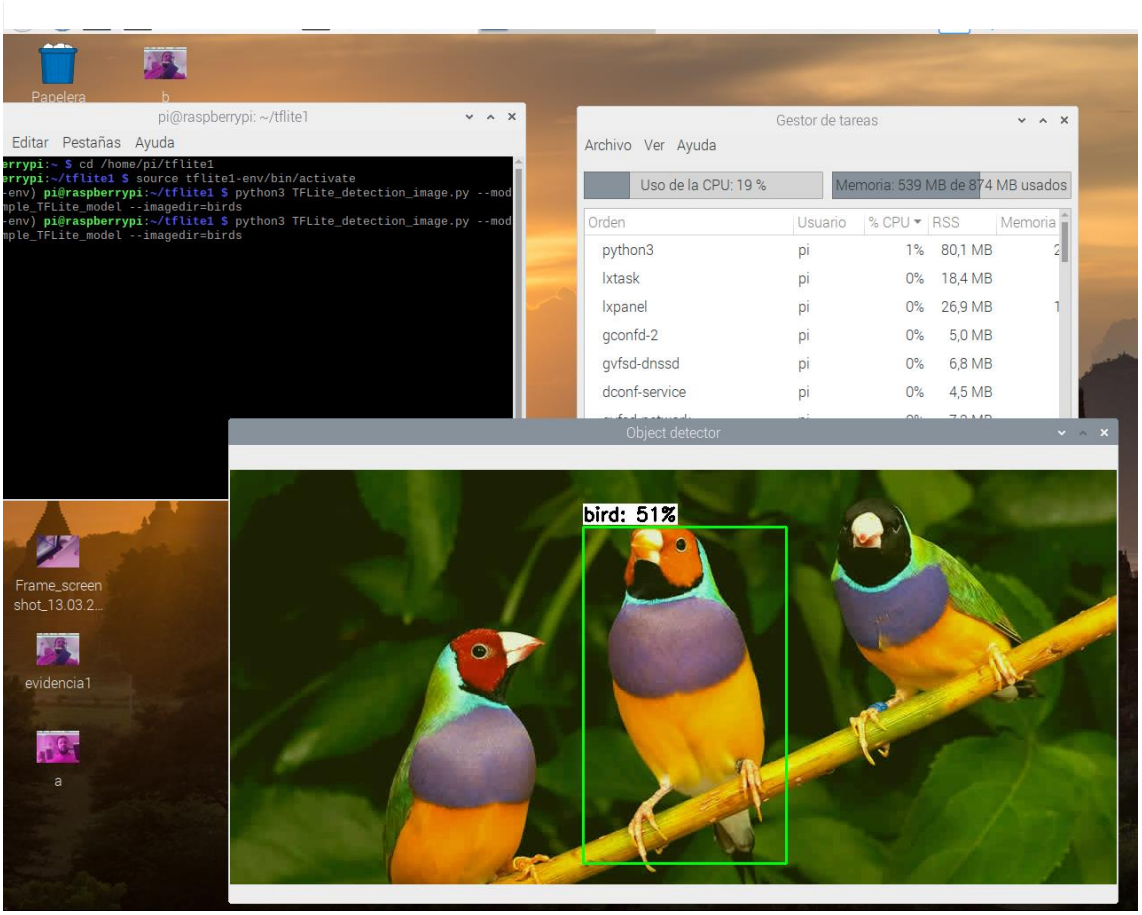
a) Detección con uso de la cámara pi: en esta prueba primero se procede a verificar el funcionamiento de la cámara y se realiza la ejecución para el reconocimiento de objetos, en este caso son: un teléfono celular, una persona y un vaso, los cuales arrojan los resultados: porcentaje de semejanza, FPS y el uso de CPU, se puede observar en la figura 23.

Figura 23 Detección con uso de la cámara Pi, Fuente Autor



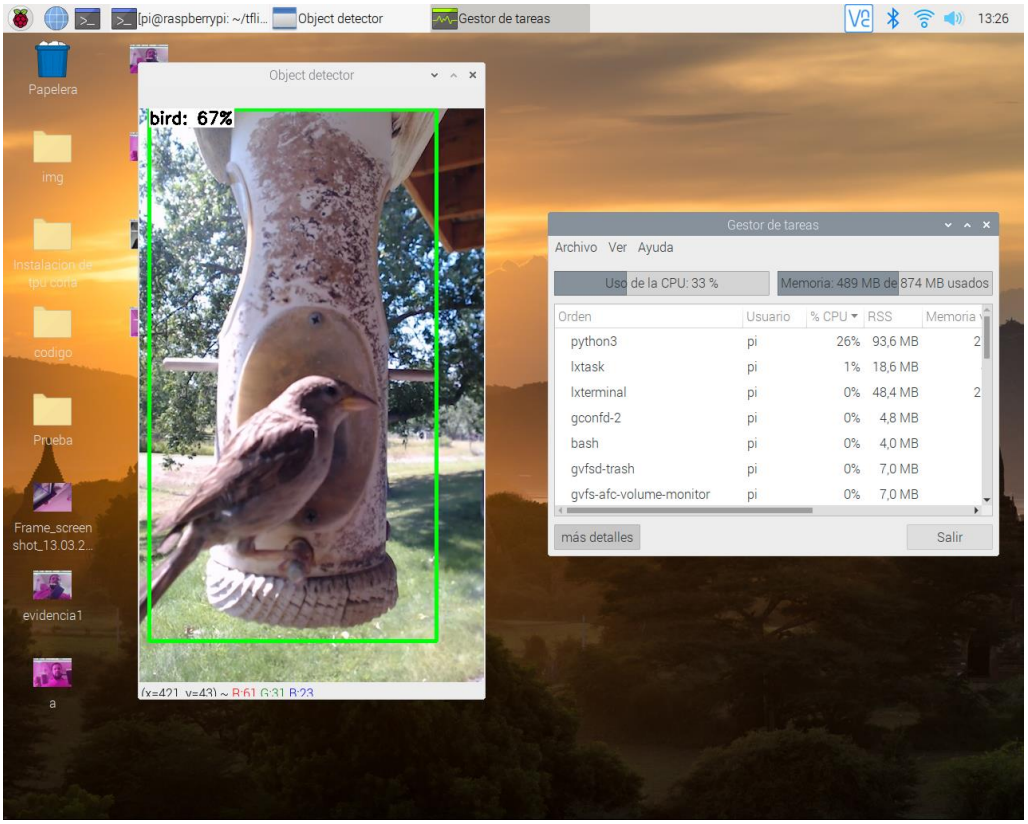
b) Detección en una imagen: en esta prueba se define el nombre del script, el dataset y por último la ruta de la imagen con su extensión (.jpg, .png, .jpeg), se procede a realizar la ejecución para el reconocimiento de objetos, en este caso se usó la imagen de un pájaro, los cuales arrojan los resultados: porcentaje de semejanza, tiempo de inferencia y uso de la CPU, se puede observar en la figura 24.

Figura 24 Prueba TensorFlow Lite, detección de imagen. Fuente Autor



c) Detección en un video: en esta prueba se define el nombre del script, el dataset y por último la ruta del video con su extensión (.avi, .mp4, .m4v, .mov), se procede a realizar la ejecución para el reconocimiento de objetos, en este caso se usó el archivo de video (test.mp4), los cuales arrojan los resultados: porcentaje de semejanza, tiempo de inferencia y uso de la CPU, se puede observar en la figura 25.

Figura 25 Prueba TensorFlow Lite, detección de video, Fuente Autor

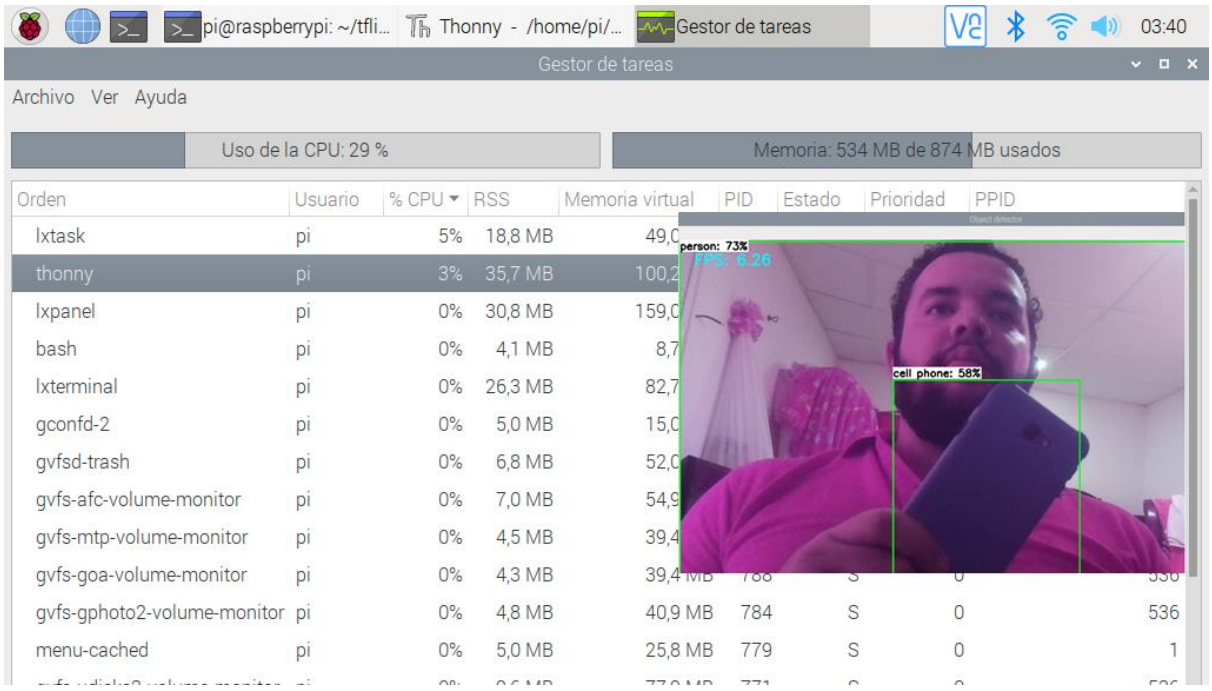


3) TensorFlow Lite con TPU

Se definen tres tipos de pruebas en esta tecnología, basados en los modelos de detección en una imagen, detección por medio de una cámara y por ultimo detección en un video. En esta sección serán realizadas las pruebas con el uso de la TPU Coral, para poder realizar el análisis de rendimiento e inferencia sin el uso de la TPU.

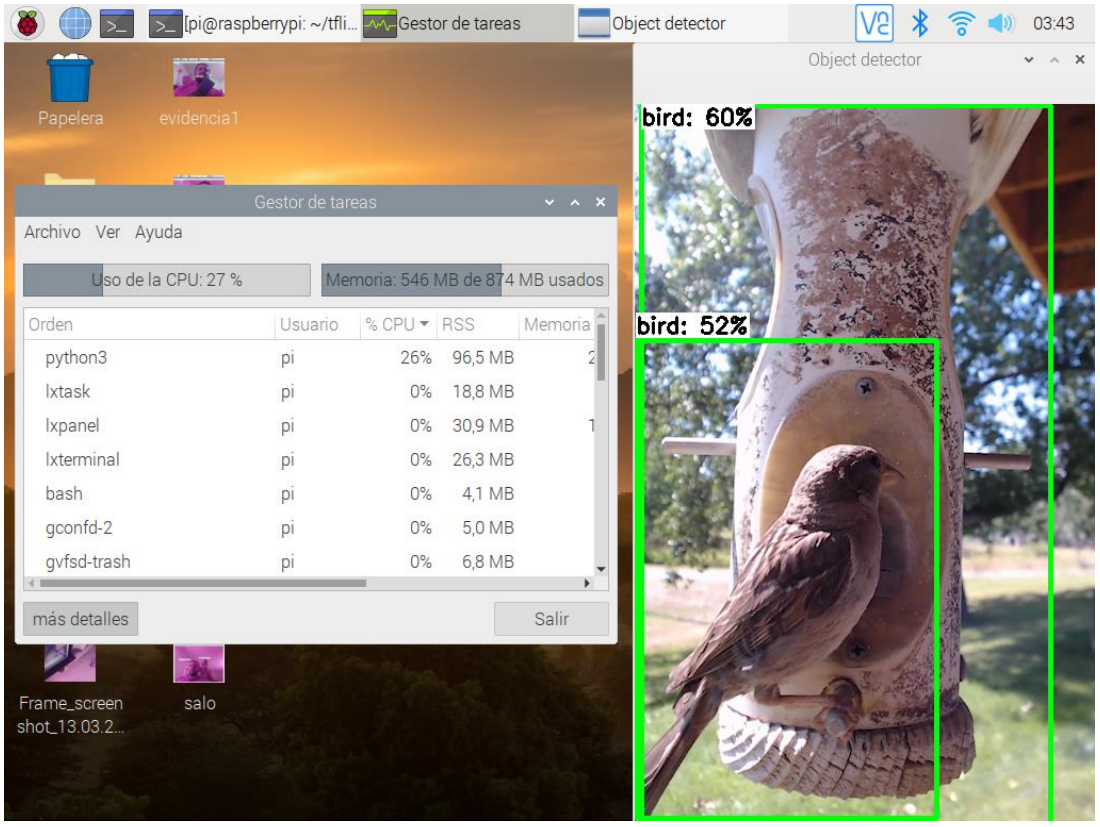
a) Detección con uso de la cámara pi: en esta prueba primero se procede a verificar el funcionamiento de la cámara y se realiza la ejecución para el reconocimiento de objetos teniendo en cuenta el comando de uso de la TPU (previamente explicado en el capítulo 4, apartado 4.3.3.), en este caso los objetos son: un teléfono celular, una persona y un vaso, los cuales arrojan los resultados: porcentaje de semejanza, FPS y el uso de CPU, se puede observar en la figura 26.

Figura 26 Prueba TensorFlow Lite con TPU, detección de cámara. Fuente Autor



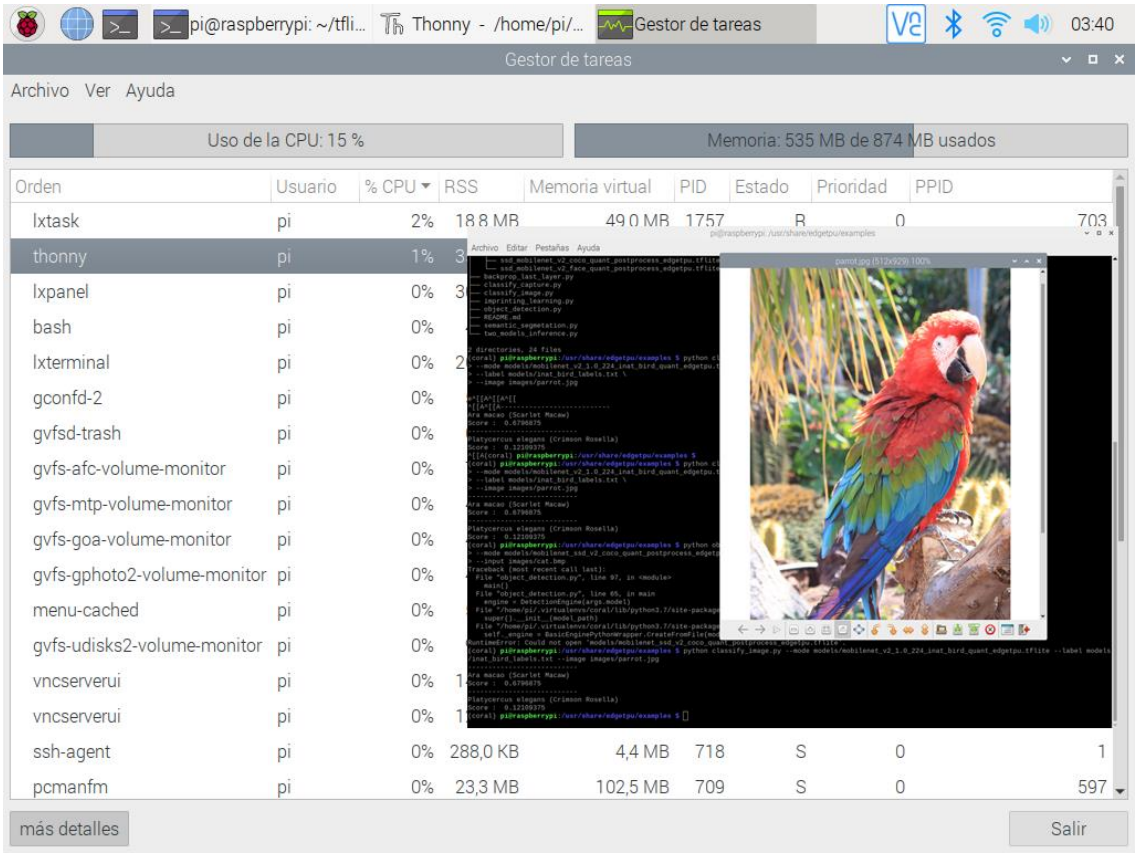
b) Detección en un video: en esta prueba se define el nombre del script, el dataset y por último la ruta del video con su extensión (.avi, .mp4, .m4v, .mov), se procede a realizar la ejecución para el reconocimiento de objetos teniendo en cuenta el comando de uso de la TPU (previamente explicado en el capítulo 4, apartado 4.3.3.), en este caso se usó el archivo de video (test.mp4), los cuales arrojan los resultados: porcentaje de semejanza, tiempo de inferencia y uso de la CPU, se puede observar en la figura 27.

Figura 27 Prueba TensorFlow Lite con TPU, detección de video. Fuente Autor



d) Detección en una imagen: en esta prueba se define el nombre del script, el dataset y por último la ruta de la imagen con su extensión (.jpg, .png, .jpeg), se procede a realizar la ejecución para el reconocimiento de objetos teniendo en cuenta el comando de uso de la TPU (previamente explicado en el capítulo 4, apartado 4.3.3.), en este caso se usó la imagen de un pájaro, los cuales arrojan los resultados: porcentaje de semejanza, tiempo de inferencia y uso de la CPU, se puede observar en la figura 28.

Figura 28 Prueba TensorFlow Lite con TPU, detección de imagen. Fuente Autor



5.4. Resultados

Se utilizaron los datos recolectados en el apartado anterior para el análisis de los resultados. Teniendo en cuenta los FPS, porcentaje de rendimiento de la CPU y una apreciación de los resultados obtenidos, los cuales son detallados en las siguientes tablas: la primera para los datos de imágenes fijas (tabla 6) y la segunda para imágenes de video (tabla 7).

Tabla 6 Resultados Obtenidos para imágenes fijas, Fuente Autor

	Tecnología	Prueba	Tiempo de inferencia (ms)	% CPU	Resultado
1	TensorFlow Lite	Detección por imagen	107	19%	Muy bueno
2	TensorFlow Lite con TPU	Detección por imagen	60	15%	Muy bueno

Tabla 7 Resultados Obtenidos para imágenes de video, Fuente Autor

	Tecnología	Prueba	Frame Rate (fps)	% CPU	Resultado
1	OpenCV	Reconocimiento por cámara Pi	0.41	92%	Pobre
2	TensorFlow Lite	Detección por video	1.71	33%	Aceptable
3	TensorFlow Lite con TPU	Detección por video	6.10	27%	Muy bueno
4	TensorFlow Lite	Detección con la cámara pi	1.57	55%	Aceptable
5	TensorFlow Lite con TPU	Detección con la cámara pi	6.26	29%	Muy bueno

Teniendo en cuenta los resultados descritos en la tabla anterior, se estipula un porcentaje de mejora para cada prueba, para esto se tienen en cuenta dos fórmulas, para las pruebas de **Detección por imagen** tenemos *(Tiempo de inferencia con TPU / Tiempo de inferencia sin TPU) * 100;* **Detección por video y Detección con la cámara pi** se requiere *(FPS con TPU / FPS sin TPU) * 100*. Estos resultados se relacionan en la tabla 8:

Tabla 8 Resumen de porcentaje de Mejoras, Fuente Autor

	Tecnología	Prueba	% de Mejoras
1	OpenCV	Reconocimiento por cámara Pi	**N.A**
2	TensorFlow Lite vs TensorFlow Lite con TPU	Detección por imagen	56,1%
3		Detección por video	256%
4		Detección con la cámara pi	298%

6. CONCLUSIONES

Al finalizar el trabajo de investigación, se logró desarrollar un sistema básico de inferencia de inteligencia artificial para la USB TPU mediante algoritmos básicos de prueba. Para llegar al cumplimiento de este objetivo se ejecutaron los tres objetivos específicos, los cuales se realizaron de la siguiente manera:

- El análisis del funcionamiento de la USB TPU Google CORAL mediante la documentación de sus manuales para la puesta de funcionamiento se realizó mediante la revisión y documentación de los procesos de instalación, prueba funcional, cargue de aplicaciones y evaluación funcional, que se encuentran descritas en este documento, sus anexos y manuales de funcionamiento, los cuales muestran de manera detallada como es el comportamiento de esta y que debemos tener en cuenta para su uso.

- En la implementación de un sistema de inferencia a partir de la USB TPU Google CORAL para la aceleración de sistemas de Inteligencia Artificial, a partir de las aplicaciones pre-entrenadas, disponibles y de una aplicación propia de reconocimiento de personas u objetos se logró la construcción de un prototipo funcional de varios sistemas de inferencia utilizando la USB TPU Google CORAL. Estos prototipos construidos y probados se utilizaron, algunos de ellos, para evaluar la mejora en la respuesta del sistema con el uso de la TPU. Esto se documentó en el capítulo 4 correspondiente a la implementación de los algoritmos, donde se explican las tecnologías utilizadas, su integración y además se detallan los pasos que se deben llevar a cabo para el cumplimiento de este.

- Para verificar la funcionalidad de la TPU, a partir del análisis de los algoritmos de prueba para evaluación de su desempeño, se realizó la implementación de varios de estos, utilizando solo la placa SBC Raspberry Pi, y agregando luego con el mismo algoritmo, el uso de la TPU, con lo cual se logró evidenciar las diferencias en rendimiento del sistema para el procesamiento de imágenes fijas y de secuencias de video, Como se ve en los resultados obtenidos, las mejoras de desempeño pueden ser hasta de casi tres veces con la adición de la TPU al sistema, y el uso de TensorFlow Lite.

7. RECOMENDACIONES

Entre las recomendaciones que realiza el autor para la implementación de un sistema de inferencia con uso de una TPU Coral, se encuentran:

- La TPU Coral cuenta con dos modos de instalación, la primera es la estándar libedgetpu1 –std y la versión acelerada libedgetpu1 –mx, brinda las siguientes características procesador de overclocks (acelerar los procesos), velocidad de fotogramas más rápido (FPS) y temperatura más caliente (hasta 38.7 ºC). Por las pruebas realizadas se recomienda el uso de la versión estándar, las mejoras no son relevantes ni sustanciales (la diferencia en FPS) y además el recalentamiento que produce la otra versión aumenta mucho la temperatura y el consumo del sistema.

- El rendimiento que puede ofrecer la TPU Coral depende del dispositivo base (en este caso la Raspberry Pi 3), por lo tanto para aprovechar completamente los recursos de esta se recomienda realizar los algoritmos de pruebas con otro dispositivo con mayores prestaciones, ejemplo: la Raspberry Pi 4.

ANEXOS

ANEXO A. ALGORITMO DE RECONOCIMIENTO

Importar librerías

Figura 29 Librerías Importadas, Fuente Autor

```
#Import the neccesary libraries
import numpy as np
import argparse
import cv2
```

Argumentos del filtro de reconocimiento de la cámara conectada a la Raspberry Pi.

Figura 30 Argumentos, Fuente Autor

```
# construct the argument parse
parser = argparse.ArgumentParser(
    description='Script to run MobileNet-SSD object detection network ')
parser.add_argument("--video", help="path to video file. If empty, camera's stream will be used")
parser.add_argument("--prototxt", default="MobileNetSSD_deploy.prototxt",
                                  help='Path to text network file: '
                                       'MobileNetSSD_deploy.prototxt for Caffe model or '
                                       )
parser.add_argument("--weights", default="MobileNetSSD_deploy.caffemodel",
                                 help='Path to weights: '
                                      'MobileNetSSD_deploy.caffemodel for Caffe model or '
                                      )
parser.add_argument("--thr", default=0.2, type=float, help="confidence threshold to filter out weak detections")
args = parser.parse_args()
```

Se definen los labels que se pueden reconocer por medio de este algoritmo, por medio del dataset previamente instalado por OpenCV (ANEXO D).

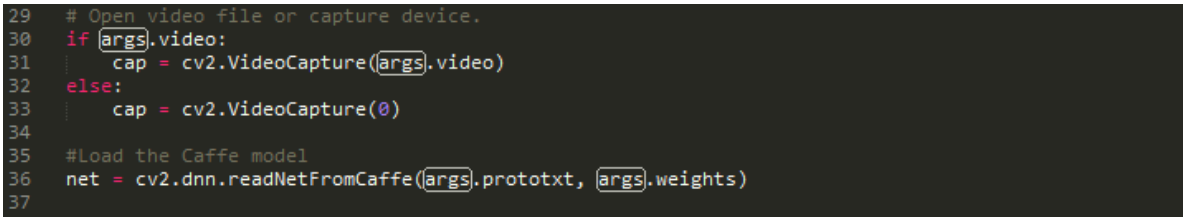
Figura 31 Definición de Labels, Fuente Autor

```
# Labels of Network.
classNames = { 0: 'background',
    1: 'aeroplane', 2: 'bicycle', 3: 'bird', 4: 'boat',
    5: 'bottle', 6: 'bus', 7: 'car', 8: 'cat', 9: 'chair',
    10: 'cow', 11: 'diningtable', 12: 'dog', 13: 'horse',
    14: 'motorbike', 15: 'person', 16: 'pottedplant',
    17: 'sheep', 18: 'sofa', 19: 'train', 20: 'tvmonitor' }

```

Se validan los argumentos, para identificar el tipo de label que reconoce el algoritmo:

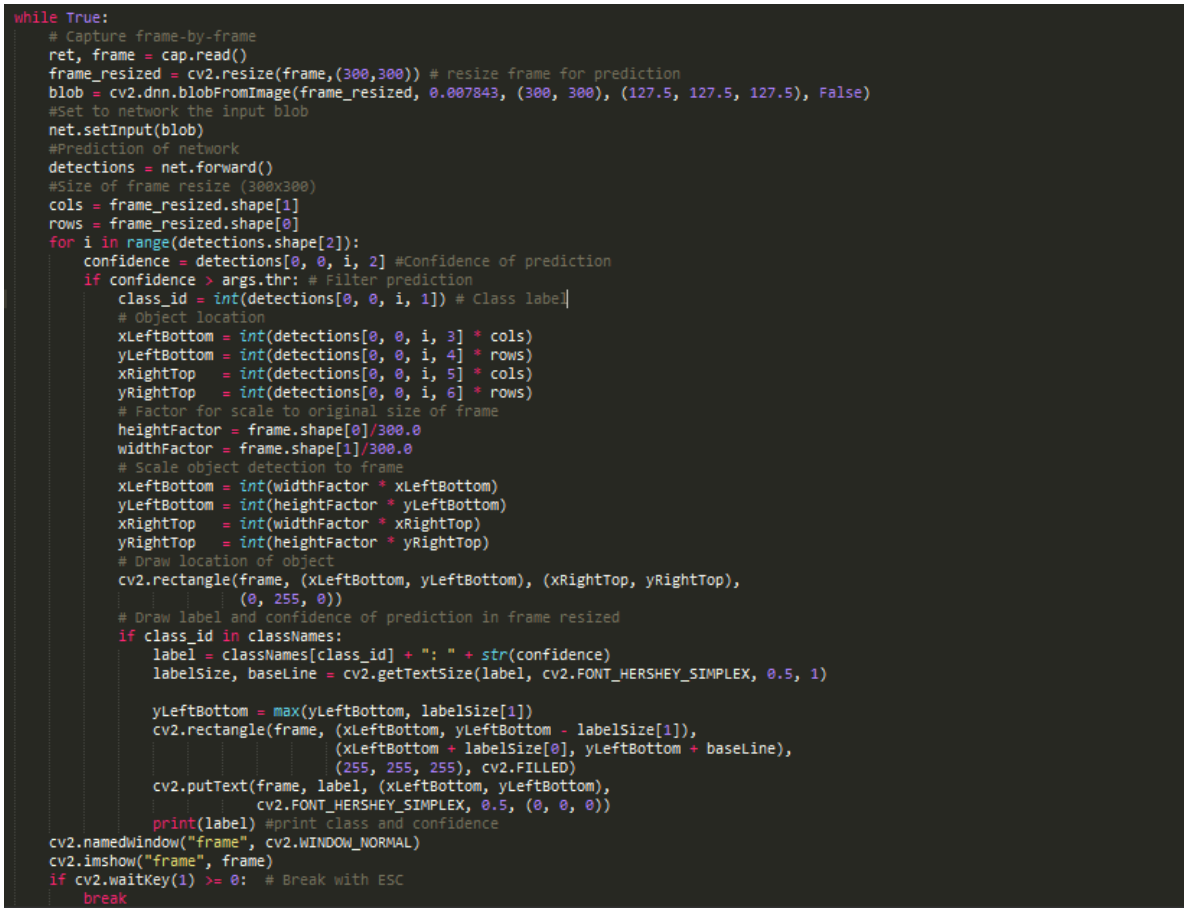
Figura 32 Validación de Argumentos, Fuente Autor



```
29  # Open video file or capture device.
30  if args.video:
31      cap = cv2.VideoCapture(args.video)
32  else:
33      cap = cv2.VideoCapture(0)
34
35  #Load the Caffe model
36  net = cv2.dnn.readNetFromCaffe(args.prototxt, args.weights)
37
```

Por último, se tiene la parte de ejecución del algoritmo. Se tienen en cuenta posición y dimensión del frame donde se visualiza la imagen capturada por la cámara, los objetos reconocidos por el algoritmo son marcados con un rectángulo y un porcentaje de probabilidad de aproximación.

Figura 33 Ejecución del Algoritmo, Fuente Autor



```
while True:
    # Capture frame-by-frame
    ret, frame = cap.read()
    frame_resized = cv2.resize(frame,(300,300)) # resize frame for prediction
    blob = cv2.dnn.blobFromImage(frame_resized, 0.007843, (300, 300), (127.5, 127.5, 127.5), False)
    #Set to network the input blob
    net.setInput(blob)
    #Prediction of network
    detections = net.forward()
    #Size of frame resize (300x300)
    cols = frame_resized.shape[1]
    rows = frame_resized.shape[0]
    for i in range(detections.shape[2]):
        confidence = detections[0, 0, i, 2] #Confidence of prediction
        if confidence > args.thr: # Filter prediction
            class_id = int(detections[0, 0, i, 1]) # Class label
            # Object location
            xLeftBottom = int(detections[0, 0, i, 3] * cols)
            yLeftBottom = int(detections[0, 0, i, 4] * rows)
            xRightTop   = int(detections[0, 0, i, 5] * cols)
            yRightTop   = int(detections[0, 0, i, 6] * rows)
            # Factor for scale to original size of frame
            heightFactor = frame.shape[0]/300.0
            widthFactor = frame.shape[1]/300.0
            # Scale object detection to frame
            xLeftBottom = int(widthFactor * xLeftBottom)
            yLeftBottom = int(heightFactor * yLeftBottom)
            xRightTop   = int(widthFactor * xRightTop)
            yRightTop   = int(heightFactor * yRightTop)
            # Draw location of object
            cv2.rectangle(frame, (xLeftBottom, yLeftBottom), (xRightTop, yRightTop),
                          (0, 255, 0))
            # Draw label and confidence of prediction in frame resized
            if class_id in classNames:
                label = classNames[class_id] + ": " + str(confidence)
                labelSize, baseLine = cv2.getTextSize(label, cv2.FONT_HERSHEY_SIMPLEX, 0.5, 1)

                yLeftBottom = max(yLeftBottom, labelSize[1])
                cv2.rectangle(frame, (xLeftBottom, yLeftBottom - labelSize[1]),
                                     (xLeftBottom + labelSize[0], yLeftBottom + baseLine),
                                     (255, 255, 255), cv2.FILLED)
                cv2.putText(frame, label, (xLeftBottom, yLeftBottom),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 0))
                print(label) #print class and confidence
    cv2.namedWindow("frame", cv2.WINDOW_NORMAL)
    cv2.imshow("frame", frame)
    if cv2.waitKey(1) >= 0:  # Break with ESC
        break
```

ANEXO B. ARCHIVO DE INSTALACIÓN Y EJECUCIÓN DE TENSORFLOW LITE Y OPENCV.

Archivo get_pi_requirements: script shell que contiene todos los paquetes y dependencias necesarios para la descarga y ejecución de TensorFlow Lite y OpenCV.

Figura 34 Archivo get_pi_requirements, Fuente Autor

get_pi_requirements.sh

```
#!/bin/bash

# Get packages required for OpenCV

sudo apt-get -y install libjpeg-dev libtiff5-dev libjasper-dev libpng12-dev
sudo apt-get -y install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev
sudo apt-get -y install libxvidcore-dev libx264-dev
sudo apt-get -y install qt4-dev-tools libatlas-base-dev

# Need to get an older version of OpenCV because version 4 has errors
pip3 install opencv-python==3.4.6.27

# Get packages required for TensorFlow
# Using the tflite_runtime packages available at https://www.tensorflow.org/lite/guide/python
# Will change to just 'pip3 install tensorflow' once newer versions of TF are added to piwheels

#pip3 install tensorflow

version=$(python -c 'import sys; print(".".join(map(str, sys.version_info[:2])))')

if [ $version == "3.7" ]; then
pip3 install https://dl.google.com/coral/python/tflite_runtime-2.1.0.post1-cp37-cp37m-linux_armv7l.whl
fi

if [ $version == "3.5" ]; then
pip3 install https://dl.google.com/coral/python/tflite_runtime-2.1.0.post1-cp35-cp35m-linux_armv7l.whl
fi

```

ANEXO C. RECONOCIMIENTO DE UNA EXTREMIDAD SUPERIOR

En el desarrollo de esta arquitectura se buscó cumplir con cada uno de los requisitos funcionales y no funcionales del software, de manera que se analizaron aspectos relacionados con el entorno y herramientas esenciales a usar.

Se usó Python como lenguaje de programación y el paradigma de programación estructurada, teniendo presente la ayuda de librerías que permiten realizar diferentes operaciones a través de sus métodos preestablecidos, como la librería OpenCV que permite el uso de métodos enfocados a la visión artificial.

Adquisición de la imagen

La adquisición de la imagen puede ser a través de un archivo de video almacenado en el equipo o a través de un dispositivo como lo son las cámaras web, los cuales pueden ser usados con la librería de visión artificial OpenCV, en este caso se trabajó con la cámara del Raspberry NoIR, por las razones expuestas en el capítulo anterior.

En primera instancia para empezar con el desarrollo y funcionamiento de la arquitectura se debe tener en cuenta la instalación de la librería de OpenCV, que será detallada en la sección del **Anexo C,** como sistema operativo, dependencias, entorno virtual (Python) y pruebas de instalación**.**

Librerías necesarias

Para el uso de las librerías, es necesario importarlas al código después de haberlas instalado en el equipo. En este prototipo se usó:

- Cv2: librería de OpenCV.

- Numpy: permite el uso de vectores y matrices.
- Math: permite el uso de funciones matemáticas.

```
import numpy as np
import cv2
import math
```

Captura

Para inicializar la captura de imagen se usa la función VideoCapture de la librería cv2, esta puede llevar como parámetro la dirección de un archivo de video almacenado en el equipo o el puerto índice del dispositivo de video; usualmente son los puertos -1, 0 y 1 según la cantidad de dispositivos conectados, predeterminadamente es el puerto 0.

```
captura = cv2.VideoCapture(0)
```

Detección del puño

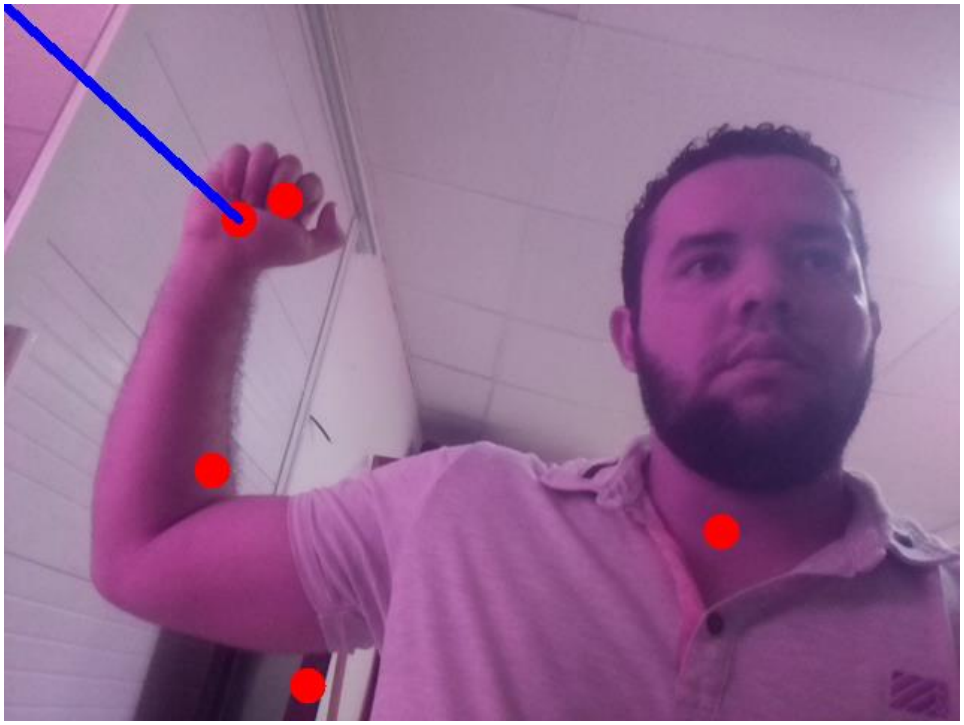
Para la detección se usa el algoritmo de Haar Cascade para puño con lo cual se obtiene la posición inicial de este. Para esto se añade el archivo clasificador con el método CascadeClassifier de cv2 y se requiere convertir el frame o fotograma a escala de grises para ejecutar sobre este el reconocimiento de las características del puño con el método detectMultiScale, y a partir de esto se halla el centro de la región que ocupa el puño en la imagen, ver Figura 9.

Es posible realizar la detección del puño de la siguiente forma:

```
fist_cascade = cv2.CascadeClassifier("fist.xml")
fist = fist_cascade.detectMultiScale(framegray, 1.3, 5)

Región del puño:
    (x, y, ancho, alto) = regionFist
```

Figura 35 Detección del puño, Fuente Autor



Seguimiento del puño

Una vez se obtuvo las coordenadas del puño, estas misma son las que se elegirán para el seguimiento del puño con el método de lucas kanade el cual se implementa con la función cv2.calcOpticalFlowPyrLK. Esta operación se realiza de la siguiente forma:

- Como práctica de la documentación de openCV se crea una variable que contiene parámetros para la función de lucas kanade.

```
lk_params = dict(
winSize  = (15,15),
maxLevel = 2,
criteria = (cv2.TERM_CRITERIA_EPS | cv2.TERM_CRITERIA_COUNT, 10, 0.03)
)
```

Donde dicha variable es un diccionario que contiene los siguientes parámetros:

- Winsize*:* es la ventana alrededor del píxel inicial sobre la que se aplicará el método de Lucas Kanade.
- MaxLevel: indica el número de pirámides que utilizará el método de Lucas Kanade.
- Criteria: especifica el rango de iteraciones que hará el método de Lucas Kanade. En este caso le estamos diciendo que haga como máximo 10 iteraciones, o cuando la ventana de búsqueda sea menor a 0.03. El Método de Lucas Kanade básicamente utiliza el método de Regresión por Mínimos Cuadrados para encontrar la nueva posición del píxel.
- Se Define una función con la cual a través de un evento de teclado se marcará el pixel central para el método de Lucas kanade.

```
def seleccionarPunto(x,y):
    global punto, punto_elegido,old_points
    punto= (x,y)
    old_points = np.array([[x, y]], dtype=np.float32)
```

El seguimiento del puño inicia con el reconocimiento de la cabeza y partiendo de este punto las extremidades, las cuales serán presentadas en las Figuras 9,10, 11, 12 y 13.

Figura 36 Reconocimiento de la cabeza, Fuente Autor

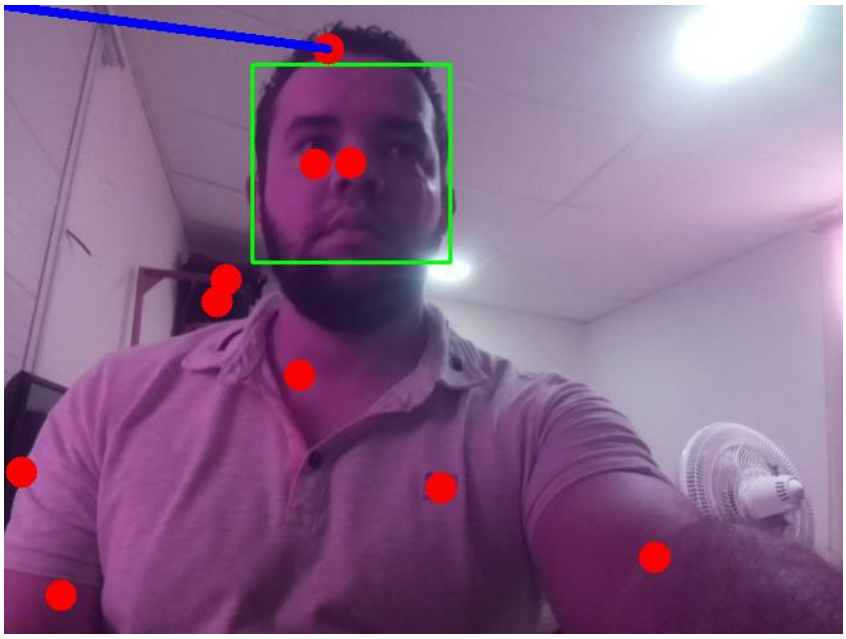


Figura 37 Reconocimiento de la extremidad partiendo de la cabeza,, Fuente Autor

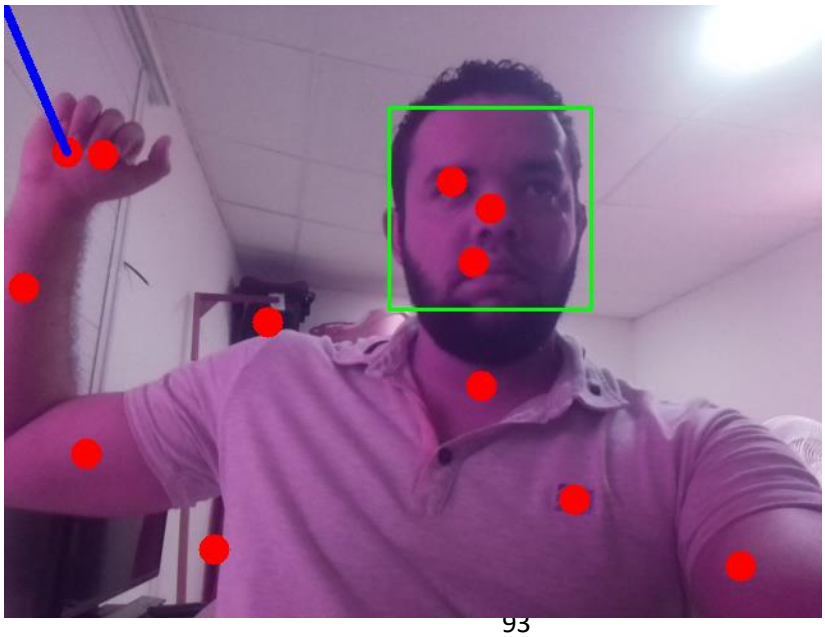


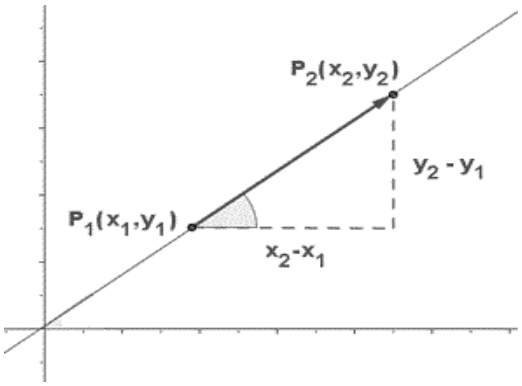
Figura 38 Reconocimiento de espalda, Fuente Autor

Cálculo de los ángulos

Para obtener los ángulos se calculó la pendiente de la recta que forma la unión de dos puntos, la pendiente mide la inclinación de una recta respecto al eje horizontal y la tangente inversa de esta es el ángulo en radianes; teniendo en cuenta que es una imagen en 2D y limita el conocimiento de varias posiciones del brazo, los ángulos representados serán cercanos a la realidad. La pendiente suele estar representada por la letra *m* y está dada por la siguiente formula:

$$m = \frac{\Delta y}{\Delta x} = \frac{y2 - y1}{x2 - x1}$$

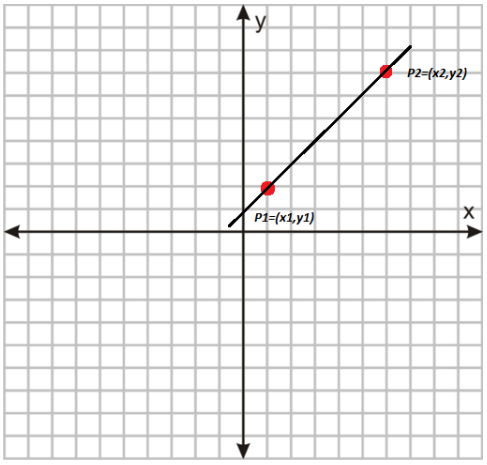
Figura 39 Representación de la pendiente en el plano, Fuente Autor



El ángulo Ɵ (teta) que se forma con el eje horizontal está dado por la siguiente formula: $\theta = \tan^{-1} m$. Si la pendiente(m) es mayor que cero se dice que esta es positiva, si es menor que cero se dice que es negativa, si es igual a cero la recta es paralela al eje horizontal(x) y si es indefina (división entre cero) la pendiente es paralela al eje vertical (Y).

Para realizar dicha operación se tomó la recta de los puntos codo a puño y hombro a codo. Esto se representa en el plano cartesiano con puntos ejemplo de la siguiente forma:

Figura 40 Recta en el plano cartesiano, Fuente Autor

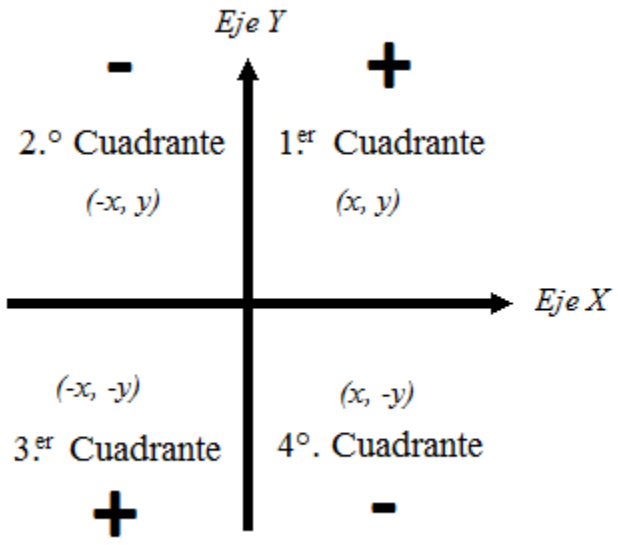


Donde la recta a tratar es codo a puño, las coordenadas en el punto uno son las coordenadas del punto del codo, y las coordenadas del punto dos, son las coordenadas del punto de puño. De igual forma se realiza con la recta hombro a codo, el punto uno corresponde al hombro y el punto dos al codo.

Ahora, puesto que no se realiza una translación de coordenadas para que el origen sea en un solo punto, el eje horizontal será trazado a partir del punto uno en ambos casos, es decir, que para cada recta se crea un eje horizontal donde el punto de origen es dado en las coordenadas del punto uno; en el caso de la recta codo a puño el origen está dado en las coordenadas del codo.

También es necesario tener en cuenta que el plano cartesiano está compuesto por cuatro cuadrantes y que tienen valores positivos y negativos como los muestra la siguiente figura:

Figura 41 Plano Cartesiano, Fuente Autor



Sabiendo lo anterior, al calcular entonces la tangente inversa de la pendiente que viene siendo el ángulo en radianes, este puede tomar valores positivos o negativos. Esto se ilustra mejor en la siguiente tabla:

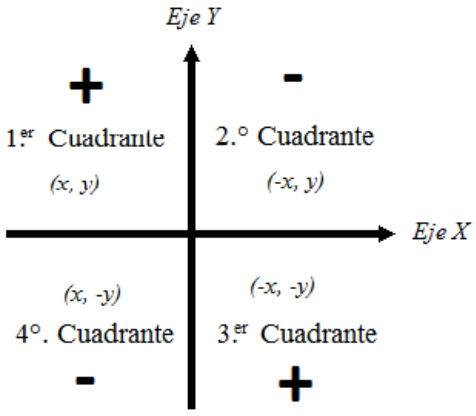
Tabla 9 Ángulos en radianes, Fuente Autor

(y2 – y1)	(x2 – x1)	Pendiente / Ángulo
+	+	+
-	+	-
-	-	+
+	-	-

Si la pendiente es positiva el ángulo también lo será, si es negativa el ángulo también lo será, puesto que el ángulo depende del cociente de la operación de la fórmula de la pendiente.

Ahora, durante el desarrollo del prototipo al calcular la tangente inversa de la pendiente esto nos da resultados de signos opuestos.

Figura 42 Tangente inversa, Fuente Autor



ANEXO D. INSTALACIÓN DE OPENCV

Se deberá tener instalado el sistema operativo **Raspbian Stretch** para aprovechar las nuevas características que ofrece este sistema comparado con **Raspbian Jessie** [38].

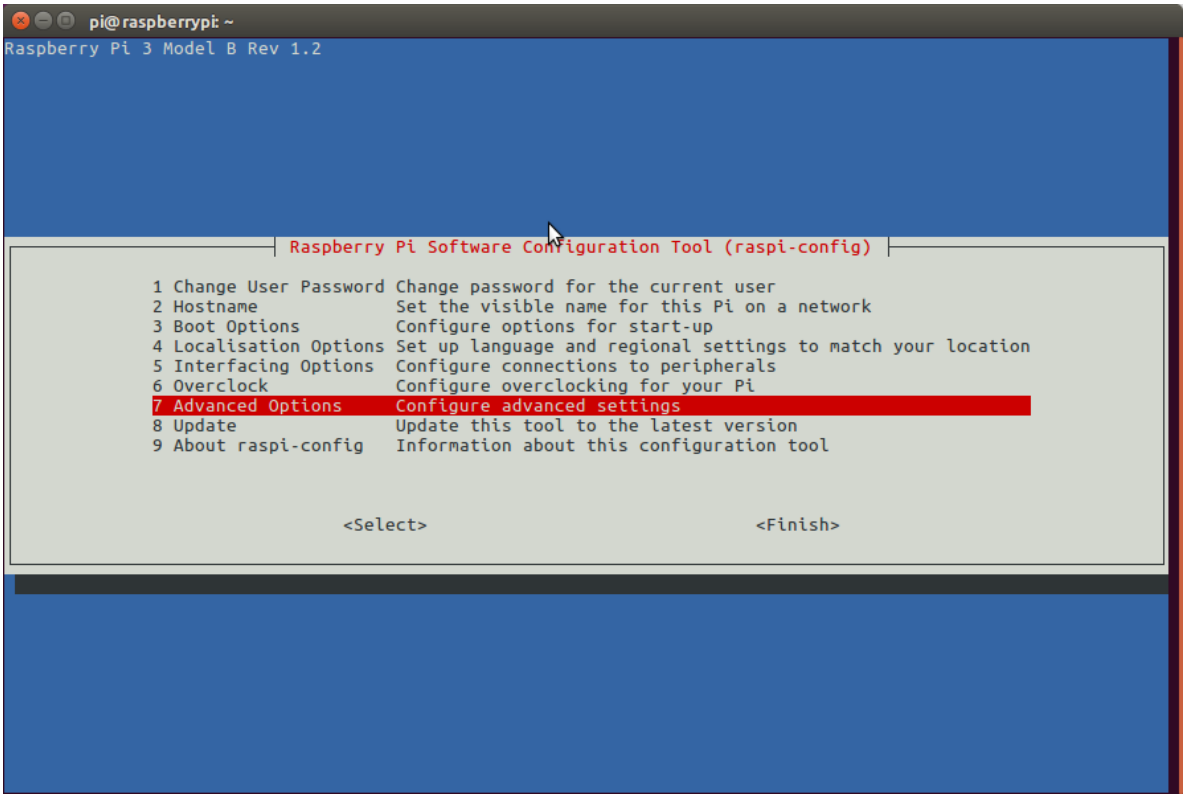
Paso # 1: Expanda el sistema de archivos

Lo primero que se debe hacer es expandir el sistema de archivos para incluir todo el espacio disponible en su tarjeta micro-SD, ver Figura 8,9, 10 y 11:

$ sudo raspi-config

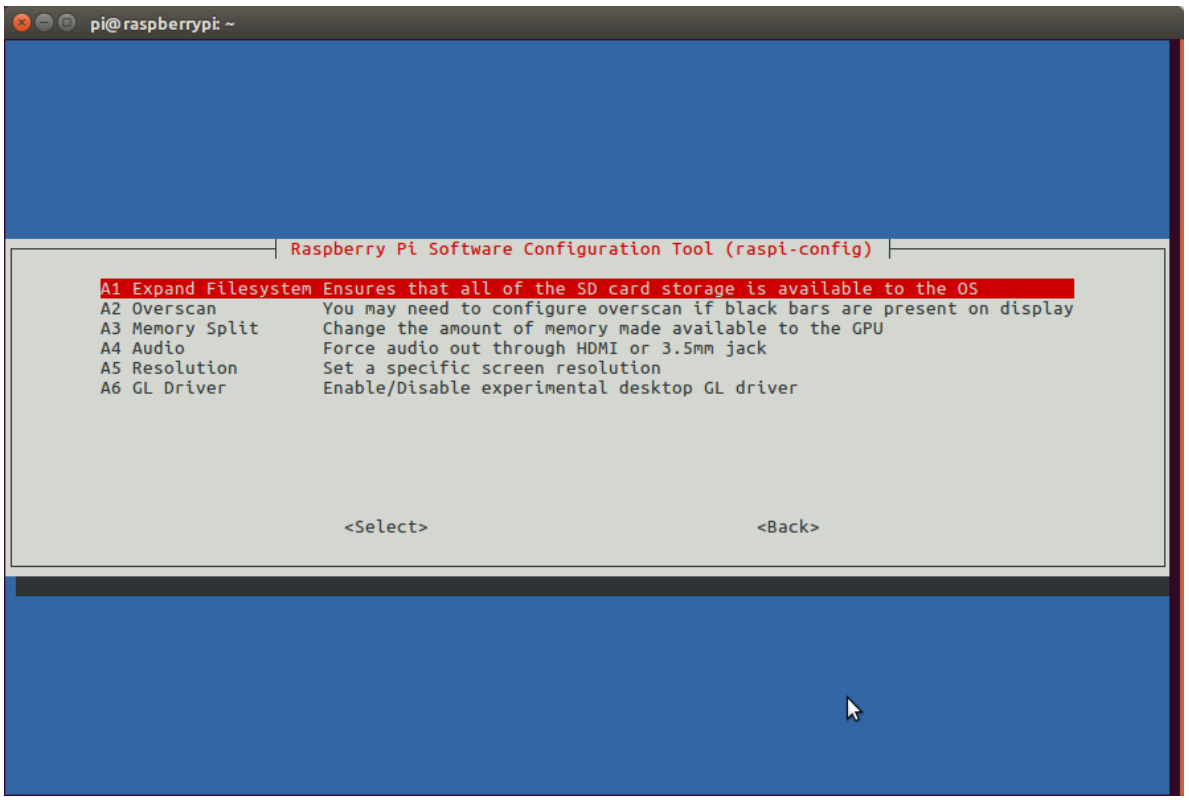
Y luego seleccione el elemento de menú "Opciones avanzadas"

Figura 43 Paso # 1, Fuente Autor



Seguido seleccionando "Expandir sistema de archivos":

Figura 44 Paso # 1, Fuente Autor



Una vez que se le solicite, debe seleccionar la primera opción, **"A1. Expanda Sistema de archivos "**, **presione Intro** en su teclado, flecha hacia abajo hasta el botón **" <Finish> "**, y luego reinicie su Pi; es posible que se le solicite reiniciar, pero si no puede ejecutar:

sudo reboot

Después de reiniciar, su sistema de archivos debería haberse ampliado para incluir todo el espacio disponible en su tarjeta micro-SD. Puede verificar que el disco se haya expandido ejecutando df - h y examinando la salida:

Figura 45 Paso # 1, Fuente Autor

```
$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root        30G  4.2G   24G  15% /
devtmpfs        434M     0  434M   0% /dev
tmpfs           438M     0  438M   0% /dev/shm
tmpfs           438M   12M  427M   3% /run
tmpfs           5.0M  4.0K  5.0M   1% /run/lock
tmpfs           438M     0  438M   0% /sys/fs/cgroup
/dev/mmcblk0p1   42M   21M   21M  51% /boot
tmpfs            88M     0   88M   0% /run/user/1000
```

Como puede ver, mi sistema de archivos Raspbian se ha ampliado para incluir los 32 GB de la tarjeta micro-SD.

Sin embargo, incluso con mi sistema de archivos expandido, ya he usado el 15% de mi tarjeta de 32GB.

Si está utilizando una tarjeta de 8GB, puede estar utilizando cerca del 50% del espacio disponible, por lo que una cosa simple es eliminar tanto el motor de LibreOffice como el de Wolfram para liberar espacio en su Pi:

```
$ sudo apt-get purge wolfram-engine
$ sudo apt-get purge libreoffice*
$ sudo apt-get clean
$ sudo apt-get autoremove
```

Figura 46 Paso # 1, Fuente Autor

Paso # 2: Instalar dependencias

El primer paso es actualizar y actualizar los paquetes existentes:

```
sudo apt-get update && sudo apt-get upgrade
```

Luego, necesitamos instalar algunas herramientas de desarrollador, incluido CMake, que nos ayuda a configurar el proceso de compilación de OpenCV:

```
sudo apt-get install build-essential cmake pkg-config
```

A continuación, necesitamos instalar algunos paquetes de E / S de imagen que nos permitan cargar varios formatos de archivo de imagen desde el disco. Ejemplos de tales formatos de archivo incluyen JPEG, PNG, TIFF, etc.

```
sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev libpng12-dev
```

Así como necesitamos paquetes de E / S de imagen, también necesitamos paquetes de E / S de video. Estas bibliotecas nos permiten leer varios formatos de archivos de video desde el disco, así como trabajar directamente con transmisiones de video:

```
sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev
sudo apt-get install libxvidcore-dev libx264-dev
```

La biblioteca OpenCV viene con un submódulo llamado highgui que se usa para mostrar imágenes en nuestra pantalla y construir GUI básicas. Para compilar el módulo highgui , necesitamos instalar la biblioteca de desarrollo GTK:

```
sudo apt-get install libgtk2.0-dev libgtk-3-dev
```

Muchas operaciones dentro de OpenCV (es decir, operaciones matriciales) se pueden optimizar aún más instalando algunas dependencias adicionales:

```
sudo apt-get install libatlas-base-dev gfortran
```

Estas bibliotecas de optimización son especialmente importantes para dispositivos con recursos limitados, como Raspberry Pi.

Por último, se deben instalar los archivos de encabezado Python 2.7 y Python 3 para que podamos compilar OpenCV con enlaces de Python:

```
sudo apt-get install python2.7-dev python3-dev
```

Paso # 3: Descargue el código fuente de OpenCV

Ahora que tenemos nuestras dependencias instaladas, tomemos el archivo 3.3.0 de OpenCV del repositorio oficial de OpenCV. Esta versión incluye el módulo dnn que discutimos en una publicación anterior donde hicimos Deep Learning con OpenCV (a medida que se publiquen versiones futuras de OpenCV. Puede reemplazar 3.3.0 con el último número de versión):

```
cd ~
wget -O opencv.zip https://github.com/Itseez/opencv/archive/3.3.0.zip
unzip opencv.zip
```

Queremos la *instalación completa* de OpenCV 3 (para tener acceso a funciones como SIFT y SURF, por ejemplo), por lo que también necesitamos tomar el repositorio opencv_contrib también:

```
wget -O opencv_contrib.zip
https://github.com/Itseez/opencv_contrib/archive/3.3.0.zip
unzip opencv_contrib.zip
```

Es posible que deba expandir el comando anterior utilizando el botón "<=>" durante su copia y pegue. El .zip en el 3.3.0.zip puede parecer estar cortado en algunos navegadores. La URL completa del archivo OpenCV 3.3.0 es:

https://github.com/Itseez/opencv_contrib/archive/3.3.0.zip

Paso # 4: ¿Python 2.7 o Python 3?

Antes de que podamos comenzar a compilar OpenCV en nuestro Raspberry Pi 3, primero debemos instalar pip, un administrador de paquetes de Python:

```
wget https://bootstrap.pypa.io/get-pip.py
sudo python get-pip.py
sudo python3 get-pip.py
```

Es **una práctica estándar** en la comunidad de Python utilizar entornos virtuales de algún tipo, por lo que le recomiendo que haga lo mismo:

```
sudo pip install virtualenv virtualenvwrapper
sudo rm -rf ~/.cache/pip
```

Ahora que se han instalado virtualenv y virtualenvwrapper, necesitamos actualizar nuestro ~ / .source de perfil para incluir las siguientes líneas en la parte inferior del archivo:

```
export WORKON_HOME=$HOME/.virtualenvs
```

```
export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python3
source /usr/local/bin/virtualenvwrapper.sh
```

Continúe y actualice el archivo para reflejar los cambios mencionados anteriormente.

De lo contrario, simplemente debe usar la redirección cat y output para manejar la actualización ~ /.profile:

```
echo -e "\n# virtualenv and virtualenvwrapper" >> ~/.profile
echo "export WORKON_HOME=$HOME/.virtualenvs" >> ~/.profile
echo "export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python3" >> ~/.profile
echo "source /usr/local/bin/virtualenvwrapper.sh" >> ~/.profile
```

Ahora que tenemos nuestro ~ / .profile actualizado, necesitamos volver a cargarlo para asegurarnos de que los cambios surtan efecto. Puede forzar una recarga de su ~ / . archivo de perfil por:

- Cerrar sesión y luego volver a iniciar sesión.
- Cerrar una instancia de terminal y abrir una nueva
- solo use el comando fuente :

```
source ~/.profile
```

Creando su entorno virtual Python

A continuación, creemos el entorno virtual de Python que usaremos para el desarrollo de la visión por computadora:

```
mkvirtualenv cv -p python2
```

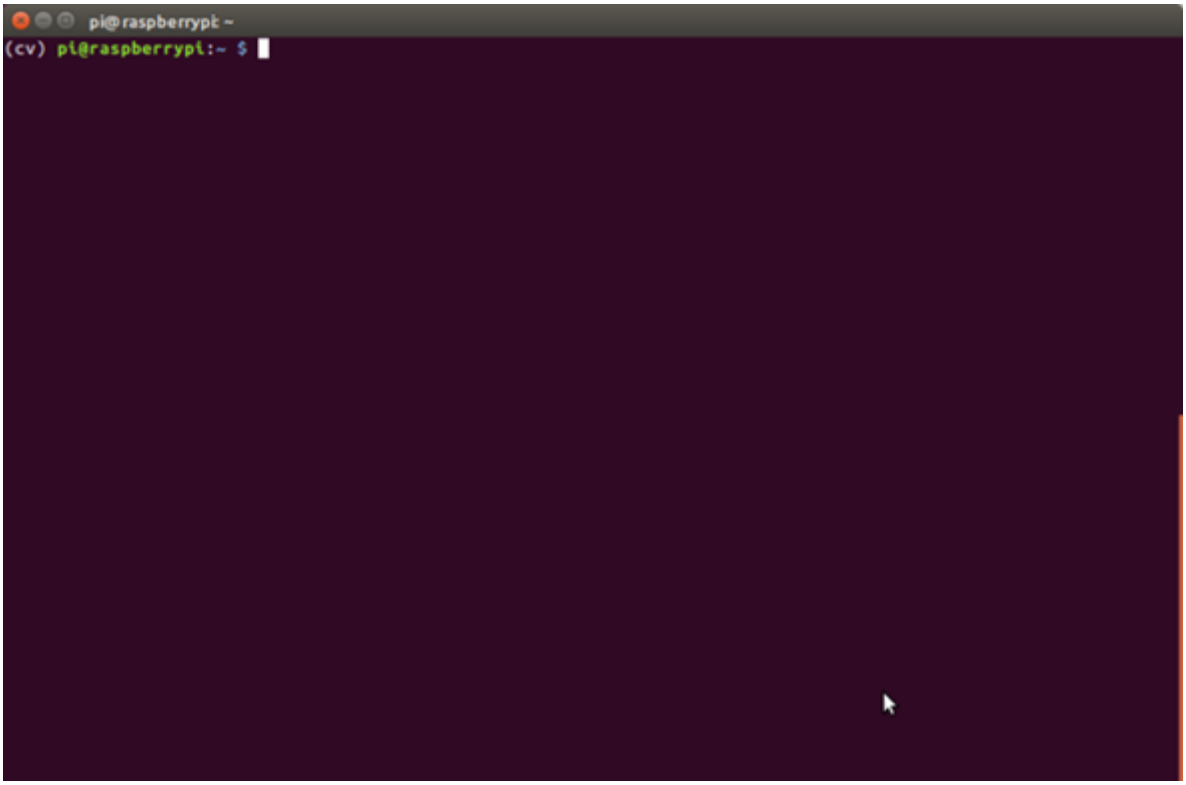
Cómo verificar si estás en el entorno virtual "cv"

Si alguna vez reinicia su Raspberry Pi; cerrar sesión y volver a iniciar sesión; o abrir una nueva terminal, deberá usar el comando workon para volver a acceder al entorno virtual de cv. El comando mkvirtualenv está destinado a ejecutarse solo una vez: para *crear* realmente el entorno virtual.
Después de eso, puede usar workon y se lo desplegará en su entorno virtual:

```
source ~/.profile
workon cv
```

Para validar y asegurarse de que se encuentra en el entorno virtual de cv, examine su línea de comando; si ve el texto (cv) que precede a su solicitud, entonces se **encuentra** en el entorno virtual de cv, (Ver Figura 12):

Figura 47 Entorno Virtual CV, Fuente Autor



Instalación Librería de NumPy

Nuestra única dependencia de Python es NumPy, un paquete de Python utilizado para el procesamiento numérico:

```
pip install numpy
```

Paso # 5: Instalar OpenCV

```
pip3 instalar opencv-python −−user
```

Paso # 6: Dependencias adicionales para OpenCV y la cámara:

sudo apt-get install libqtgui4

sudo modprobe bcm2835-v4l2

sudo apt-get install libqt4-test

ANEXO E. REQUISITOS FUNCIONALES

Tabla 10 RF-01, Fuente Autor

Identificador	**RF-01**
Nombre del requerimiento:	Cuantificar los FPS (fotogramas por segundo) en la Raspberry sin la TPU
Actor	
Indispensable/ deseable	Indispensable.
Prioridad.	Alta
Visible	Si
Autor	Carlos Eduardo Ospino Martínez
Fecha de elaboración:	28/08/2019
Revisión:	Luis Murillo
Última revisión:	20/05/2020
Resumen:	Se determinan la cantidad de fotogramas procesados con el script ejecutado con TensorFlow Lite.
Eventos:	1. Iniciada la Raspberry Pi, con su sistema operativo. 2. Cargar el modelo en la ruta instalada. 3. Conexión de la cámara Pi. 4. Ejecutar el script.
Precondiciones:	1. Exista el modelo a ejecutar. 2. Instalación de las librerías requeridas: TensorFlow Lite, Python y OpenCV.
Post-condiciones:	N/A

Tabla 11 RF-02, Fuente Autor

Identificador	**RF-02**
Nombre del requerimiento:	Cuantificar los FPS en la Raspberry con la TPU
Actor	
Indispensable/ deseable	Indispensable.
Prioridad.	Alta
Visible	Si
Autor	Carlos Eduardo Ospino Martínez
Fecha de elaboración:	28/08/2019
Revisión:	Luis Murillo
Última revisión:	28/08/2019
Resumen:	
Eventos:	
Precondiciones:	RF-01
Post-condiciones:	
Fecha de elaboración:	28/08/2019
Revisión:	Luis Murillo
Última revisión:	28/08/2019
Resumen:	Se determinan la cantidad de fotogramas procesados con el script ejecutado con TensorFlow Lite y la TPU Coral.
Eventos:	1. Iniciada la Raspberry Pi, con su sistema operativo. 2. Cargar el modelo en la ruta instalada. 3. Conexión de la cámara Pi. 4. Ejecutar el script.
Precondiciones:	1. Exista el modelo a ejecutar. 2. Instalación de las librerías requeridas: TensorFlow Lite, Python y OpenCV. 3. Descargar archivos de la TPU Coral.
Post-condiciones:	N/A

Tabla 12 RF-03, Fuente Autor

Identificador	**RF-03**
Nombre del requerimiento:	Optimizar la capacidad de cómputo de la Raspberry Pi.
Actor	
Indispensable/ deseable	Indispensable.
Prioridad.	Alta
Visible	Si
Autor	Carlos Eduardo Ospino Martínez
Fecha de elaboración:	28/08/2019
Revisión:	Luis Murillo
Última revisión:	
Resumen:	Se hace el análisis de múltiples librerías para identificar la más óptima, se selecciona como herramienta principal de trabajo TensorFlow Lite combinado con la TPU Coral.
Eventos:	1. Mejoras en el rendimiento de la Raspberry Pi.
Precondiciones:	RF-01, RF-02
Post-condiciones:	N/A

Tabla 13 RF-04, Fuente Autor

Identificador	**RF-04**
Nombre del requerimiento:	Probar algoritmos predeterminado en la TPU.
Actor	
Indispensable/ deseable	Indispensable.
Prioridad.	Alta
Visible	Si
Autor	Carlos Eduardo Ospino Martínez
Fecha de elaboración:	28/08/2019
Revisión:	Luis Murillo
Última revisión:	
Resumen:	Para determinar su funcionamiento se ejecutan los algoritmos de pruebas.
Eventos:	1. Se observa la puesta en marcha de los modelos de aprendizaje pre-entrenados en la TPU Coral.
Precondiciones:	1. Exista el modelo a ejecutar. 2. Instalación de las librerías requeridas: TensorFlow Lite, Python y OpenCV. 3. Descargar algoritmos de prueba de la TPU Coral.
Post-condiciones:	N/A

ANEXO F. REQUISITOS NO FUNCIONALES

Tabla 14 RNF-01, Fuente Autor

Identificador	RNF-01
Nombre del requerimiento:	Entrenamiento óptimo de modelos.
Indispensable/ deseable	Indispensable.
Prioridad.	Alta
Visible	Si
Autor	Carlos Eduardo Ospino Martínez
Fecha de elaboración:	28/08/2019
Revisión:	Luis Murillo
Última revisión:	
Resumen:	Los modelos no deben estar mal entrenados, lo cual garantiza y facilita el reconocimiento de estos.

Bibliografía

[1] Motyon Analysis solution, «stt-systems,» [En línea]. Available: https://www.stt-systems.com/motion-analysis/inertial-motion-capture/?gclid=EAIaIQobChMIxdbH55P44AIV1lqGCh0k6gIKEAAYASAAEgLgEfD_BwE. [Último acceso: 04 03 2020].

[2] S. M. M. Jonathan Rodríguez Marante, «Protocolo de trabajo y banco de pruebas para el uso del equipo de captura de movimiento MOCAP-OPTITRACK,» 06 2015. [En línea]. Available: https://riull.ull.es/xmlui/bitstream/handle/915/1111/Protocolo%20de%20trabajo%20y%20banco%20de%20pruebas%20para%20el%20uso%20del%20equipo%20de%20captura%20de%20movimientos%20MOCAP-Optitrack.pdf;sequence=1. [Último acceso: 04 03 2020].

[3] «ULL,» [En línea]. Available: http://atrae.webs.ull.es/capturadormovimiento/?page_id=30. [Último acceso: 04 03 2020].

[4] L. U. Irurtia, «euskadi,» [En línea]. Available: http://www.euskadi.eus/contenidos/noticia/errendimendua/es_def/adjuntos/Luis%20Unzueta.pdf. [Último acceso: 20 02 2020].

[5] Tienda Xcitex, [En línea]. Available: http://www.xcitex.com/proanalyst-motion-analysis-software.php. [Último acceso: 13 03 2020].

[6] MOVIMIENTO ORGÁNICO, INC, [En línea]. Available: https://www.businesswire.com/news/home/20090401005279/en/Organic-Motion%E2%80%99s-STAGE%E2%84%A2-BioSTAGE%E2%84%A2-Systems-Schools. [Último acceso: 12 02 2020].

[7] L. Hattersley., «magpi,» 06 03 2019. [En línea]. Available: https://magpi.raspberrypi.org/articles/teachable-machine-coral-usb-accelerator. [Último acceso: 04 04 2020].

[8] F. L. Saca, «PROTOTIPO FUNCIONAL PARA CLASIFICACIÓN DE,» Mexico, 2018.

[9] MLPerf, [En línea]. Available: https://mlperf.org/training-overview. [Último acceso: 24 05 2020].

[10] L. A. Libutti, «workshops,» [En línea]. Available: http://workshops.inf.ed.ac.uk/accml/papers/2020/AccML_2020_paper_4.pdf. [Último acceso: 02 06 2020].

[11] S. Gupta, «arxiv,» 2020. [En línea]. Available: https://arxiv.org/pdf/2003.02838.pdf. [Último acceso: 24 04 2020].

[12] SciELO, 14 08 2015. [En línea]. Available: http://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S0188-95322016000200149#B18. [Último acceso: 05 05 2020].

[13] «Tutor de programación,» [En línea]. Available: http://acodigo.blogspot.com/p/tutorial-opencv.html. [Último acceso: 11 03 2020].

[14] xataca, [En línea]. Available: http://www.xataka.com/tag/raspberry-pi. [Último acceso: 10 03 2019].

[15] J. D. d. Usera, «hardzone,» 31 03 2019. [En línea]. Available: https://hardzone.es/2019/03/31/google-coral-ordenador-raspberry-pi/. [Último acceso: 04 03 2020].

[16] A. R. Bazaga, 18 08 2015. [En línea]. Available: https://osl.ull.es/software-libre/opencv-libreria-vision-computador/. [Último acceso: 05 03 2020].

[17] M. A. Alvarez, 19 11 2003. [En línea]. Available: https://desarrolloweb.com/articulos/1325.php. [Último acceso: 03 03 2020].

[18] nextu, [En línea]. Available: https://www.nextu.com/blog/que-es-el-procesamiento-de-datos/. [Último acceso: 23 03 2020].

[19] L. Llamas, 15 10 2017. [En línea]. Available: https://www.luisllamas.es/que-es-raspberry-pi/. [Último acceso: 30 04 2020].

[20] Hewlett Packard, [En línea]. Available: https://www.hpe.com/lamerica/es/what-is/artificial-intelligence.html. [Último acceso: 13 04 2020].

[21] Microsoft, 30 03 2017. [En línea]. Available: https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/dataset-datatable-dataview/. [Último acceso: 16 04 2020].

[22] Cognex Corporation, [En línea]. Available: https://www.cognex.com/es-co/what-is/machine-vision/what-is-machine-vision. [Último acceso: 24 04 2020].

[23] H. Ortega, Desarrollo de un sistema de captura de movimiento:, Académica Española, 2013.

[24] M. L. Michelone, «UNOCERO,» 2017. [En línea]. Available: https://www.unocero.com/ciencia/tensorflow-lite-para-dispositivos-moviles/. [Último acceso: 03 03 2020].

[25] COCO, [En línea]. Available: http://cocodataset.org/#home. [Último acceso: 24 04 2020].

[26] D. Rodriguez, «Lifeder,» [En línea]. Available: https://www.lifeder.com/investigacion-aplicada/.

[27] «Agile Software Development,» [En línea]. Available: http://www.agile-process.org/. [Último acceso: 23 03 2020].

[28] I. 830. [En línea]. Available: https://www.fdi.ucm.es/profesor/gmendez/docs/is0809/ieee830.pdf. [Último acceso: 03 01 2020].

[29] Google, «coral,» [En línea]. Available: https://coral.ai/docs/accelerator/get-started/. [Último acceso: 28 02 2020].

[30] Mercado Negro, 30 01 2020. [En línea]. Available: https://www.mercadonegro.pe/marketing/como-funciona-coral-la-plataforma-de-inteligencia-artificial-de-google/. [Último acceso: 30 03 2020].

[31] «Parada Visual,» 12 01 2020. [En línea]. Available: https://www.paradavisual.com/google-y-su-nuevo-chip-coral-para-inteligencia-artificial/. [Último acceso: 13 03 2020].

[32] Google, 2020. [En línea]. Available: https://cloud.google.com/edge-tpu?hl=es. [Último acceso: 08 05 2020].

[33] CORAL, [En línea]. Available: https://coral.ai/docs/edgetpu/models-intro/#compatibility-overview. [Último acceso: 23 03 2020].

[34] S. Sterckval. [En línea]. Available: https://blog.raccoons.be/coral-tpu-jetson-nano-performance. [Último acceso: 05 03 2020].

[35] A. Annamaa, «Fedora,» 19 Febrero 2018. [En línea]. Available: https://fedoramagazine.org/learn-code-thonny-python-ide-beginners/. [Último acceso: 05 03 2020].

[36] coral, [En línea]. Available: https://coral.ai/. [Último acceso: 24 04 2020].

[37] GitHub, [En línea]. Available: https://github.com/EdjeElectronics/TensorFlow-Lite-Object-Detection-on-Android-and-Raspberry-Pi/blob/master/Raspberry_Pi_Guide.md. [Último acceso: 24 05 2020].

[38] pyimagesearch, «pyimagesearch,» 10 2019. [En línea]. Available: https://www.pyimagesearch.com/2017/09/04/raspbian-stretch-install-opencv-3-python-on-your-raspberry-pi/. [Último acceso: 04 03 2020].

[39] M. A. Alvarez, «Desarrollo Web,» 19 11 2003. [En línea]. Available: https://desarrolloweb.com/articulos/1325.php. [Último acceso: 04 03 2020].

[40] D. Rodriguez, «Lifeder,» [En línea]. Available: https://www.lifeder.com/investigacion-aplicada/. [Último acceso: 13 04 2020].

[41] D. Rodriguez, «Lifeder,» [En línea]. Available: https://www.lifeder.com/investigacion-aplicada/. [Último acceso: 30 05 2020].

[42] SciELO, 14 08 2015. [En línea]. Available: http://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S0188-95322016000200149. [Último acceso: 23 03 2020].

[43] BBC, «El inesperado éxito de Raspberry Pi,» [En línea]. Available: https://www.bbc.com/mundo/blogs/2013/07/130710_blog_un_mundo_feliz_raspberry_pi. [Último acceso: 03 03 2020].

[44] «Especificación de Requisitos según el estándar,» [En línea]. Available: https://www.fdi.ucm.es/profesor/gmendez/docs/is0809/ieee830.pdf. [Último acceso: 10 09 2019].

[45] hackster, «hackster,» hackster, 14 02 2020. [En línea]. Available: https://www.hackster.io/ansonhe1997/image-classification-with-raspberry-pi-and-google-coral-usb-0ed12f. [Último acceso: 03 03 2020].

[46] Tutor de Programación, «Tutor de Programación,» [En línea]. Available: http://acodigo.blogspot.com/p/tutorial-opencv.html. [Último acceso: 28 05 2020].

[47] «Revista mexicana de ingeniería biomédica,» 05 2016. [En línea]. Available: http://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S0188-95322016000200149. [Último acceso: 20 02 2019].