Geniia .com

2025

Curso de Robótica

Visión, control e inteligencia artificial.

Presented By

Ing. Santiago Sanchez

Descripción

del curso

Este curso integral explora el desarrollo de un sistema de brazo robótico inteligente, desde conceptos fundamentales hasta implementaciones avanzadas. Los estudiantes construirán un sistema de brazo robótico completo capaz de escanear el entorno, detectar objetos y realizar manipulaciones autónomas mediante visión artificial, cinemática, control por voz y modelos de lenguaje de gran tamaño.

Nivel 1.

Fundamentos basicos de robotica, y vision artificial.

Modulo 1: Arquitectura y configuración del sistema

- Introducción y descripción general del sistema.
  - Hoja de ruta y objetivos del curso.
  - Componentes de hardware (Raspberry Pi 5, VEX IQ).
  - Fundamentos de la arquitectura maestro-esclavo.
  - Descripción general de los servicios del sistema.

- Diseño del sistema.
  - Fundamentos y arquitectura modular.
  - Diagrama de flujo del sistema.
  - Comunicación entre modulos.

- Diseño de protocolos de comunicación.
  - Protocolos de comunicación.
  - Fundamentos de la comunicación serial.
  - Implementación de la comunicación serial.
  - Diseño de un protocolo robusto basado en JSON.

Descripción

del curso

Nivel 1.

Fundamentos basicos de robotica, y vision artificial.

Modulo 2: Movimiento básico y control

- Fundamentos basicos de movimiento del motor.
  - Fundamentos del control del motor.
  - Sistemas de control conjunto.
  - Secuencias de movimiento.

- Implementacion de servicios básicos de movimiento.
  - Servicio básico de posición segura.
  - Servicio básico de escaneo y perpcepción del entorno.
  - Servicio básico de pick & place.

Modulo 3: Visión artificial básica

- Introducción a la visión artificial.
  - Fundamentos de la visión artificial.
  - Conexión de camaras a la Raspberry PI.

- Fundamentos de la detección de objetos.
  - Introducción a la detección de objetos.
  - Introducción a la arquitectura YOLO.
  - Configuracion de modelos preentrenados.
  - Inferencia basica YOLO.
  - Conversión de modelos.

- Identificación de objetos.
  - Captura de imágenes.
  - Conceptos basicos de procesamiento y visualización.
  - Dibujar y guardar resultados.

Descripción

del curso

Nivel 1.

Fundamentos basicos de robotica, y vision artificial.

Modulo 4: Integración del sistema

- Proyecto final.
  - Integración de todos los servicios.
  - Construcción del sistema básico completo.
  - Demostración de capacidades basicas.
  - Identificacion de fallos y posibles mejoras.
  - Evaluación del desempeño.

Hardware

Vex IQ

Articulaciones y Juntas Cinemáticas.

Las articulaciones en robótica son los puntos de conexión que permiten el movimiento relativo entre dos partes de un robot. Las juntas cinemáticas describen cómo estas articulaciones facilitan el movimiento en términos de grados de libertad (DOF, por sus siglas en inglés), que indican las direcciones independientes en las que un componente puede moverse. En el caso del Armbot IQ, un robot educativo con un brazo robótico, las articulaciones son esenciales para permitir que el brazo y la garra se desplacen y manipulen objetos.

Tipos de Articulaciones:

Articulaciones Rotativas:

Estas permiten la rotación alrededor de un eje. En el Armbot IQ, es probable que se utilicen articulaciones rotativas para movimientos como:

- Girar la base del brazo en el plano horizontal.
- Girar el hombro para extender el brazo.
- Girar el "codo" del brazo para elevar o bajar la garra.

Este tipo de articulación es común en brazos robóticos debido a su simplicidad y versatilidad.

Articulaciones Prismaticas:

Estas permiten el movimiento lineal a lo largo de un eje (como un pistón). Aunque son menos frecuentes en robots pequeños como los de VEX IQ, podrían usarse para extensiones lineales del brazo en diseños más avanzados.

Hardware

Vex IQ

Relación de Engranajes.

Los engranajes son componentes fundamentales en el Armbot IQ, ya que transmiten el movimiento y la fuerza desde los motores a las articulaciones y la garra. La relación de engranajes define cómo se transforman la velocidad y el torque entre el engranaje conductor (conectado al motor) y el engranaje conducido (conectado a la parte móvil).

Cálculo de Relación de Engranajes:

La relación de engranajes se calcula como:
Relación = Número de dientes del engranaje conducido / Número de dientes del engranaje conductor.

- Ejemplo:
  - Si el engranaje conductor tiene 12 dientes y el conducido tiene 36 dientes:
  - Relación = 36 / 12 = 3:1.
  - Esto significa que el engranaje conductor debe girar tres veces para que el engranaje conducido complete una vuelta. Como resultado, la velocidad se reduce en un factor de 3, pero el torque aumenta en la misma proporción (ignorando pérdidas por fricción).

Aplicación en el Armbot IQ:

- Reducción de Velocidad y Aumento de Torque:
  - En el brazo robótico, los engranajes se usan para disminuir la velocidad de los motores y aumentar el torque, permitiendo levantar cargas más pesadas. Por ejemplo, la articulación del codo podría usar una relación de engranajes como 5:1 para proporcionar suficiente fuerza para elevar el brazo y la garra con un objeto.
- Cadena de Engranajes:
  - Es probable que el Armbot IQ emplee una serie de engranajes conectados (una "cadena de engranajes") para lograr la relación deseada entre el motor y la articulación.

Hardware

Vex IQ

Otros Aspectos Mecánicos.

Además de las articulaciones y los engranajes, hay varios elementos mecánicos clave en el diseño del Armbot IQ:

Estructura y Materiales.

- El chasis y las partes del brazo están fabricados en plástico resistente, lo que los hace ligeros pero duraderos.
- Las piezas se ensamblan con pines y conectores, facilitando el montaje y desmontaje, típico de los sistemas modulares VEX IQ.

Mecanismos de Transmisión.

- Engranajes: Son el método principal para transmitir movimiento en VEX IQ.
- Ejes metálicos: Conectan los engranajes y las articulaciones, asegurando rigidez y precisión en la transmisión del movimiento.
- Aunque las correas o cadenas son posibles en robótica, en VEX IQ los engranajes predominan por su simplicidad.

Arquitectura del Sistema

Módulo Principal.

Definición.

Este módulo actúa como el orquestador central del sistema, gestionando la interacción con el usuario y coordinando las acciones entre los demás módulos.

Robot:

- main_menu_loop: Muestra un menú interactivo al usuario con opciones como check, safety, scan, pick & place, o salir, y procesa los comandos ingresados.
- handle_scan_command: Inicia el servicio de escaneo, registra callbacks para procesar datos de escaneo y gestiona el proceso completo de escaneo.
- handle_pick_place_command: Ejecuta el servicio de pick & place, permitiendo al usuario seleccionar un objeto detectado y realizar la secuencia correspondiente.
- process_scan_results: Procesa y muestra los resultados del escaneo, organizando la información de los objetos detectados (ángulo, distancia, clase, confianza).
- select_object_interactively: Permite al usuario seleccionar un objeto de la lista de objetos escaneados de forma interactiva.
- execute_pick_sequence: Ejecuta la secuencia de movimientos para recoger un objeto, enviando comandos al brazo robótico.
- execute_place_sequence: Ejecuta la secuencia de movimientos para colocar un objeto en una zona designada según su clase.
- get_current_angles: Obtiene los ángulos actuales de los joints del brazo desde el módulo de comunicación.
- execute_movement: Envía comandos de movimiento al brazo y espera confirmaciones del módulo de comunicación.
- handle_movement_failure: Maneja fallos en las secuencias de movimiento, activando un protocolo de seguridad si es necesario.

Arquitectura del Sistema

Módulo Comunicación.

Definición.

Este módulo gestiona la comunicación serial entre el sistema principal y el hardware del brazo robótico, procesando mensajes enviados y recibidos.

CommunicationManager:

- connect: Establece la conexión serial con el puerto especificado (por ejemplo, /dev/ttyACM1).
- close: Cierra la conexión serial y detiene el hilo de lectura.
- send_message: Envía mensajes en formato JSON al robot, como comandos de servicio o movimiento.
- register_callback: Registra funciones callback para procesar tipos específicos de mensajes recibidos.
- _read_loop: Ejecuta un bucle continuo en un hilo separado para leer mensajes entrantes del robot.
- _process_message: Procesa los mensajes recibidos según su tipo (check, safety, scan, etc.) y ejecuta acciones o callbacks correspondientes.
- _handle_object_detection: Maneja la detección de objetos en tiempo real, capturando imágenes y procesándolas con el modelo YOLO.
- get_scan_data: Espera y devuelve los datos completos del escaneo una vez finalizado.
- wait_for_confirmation: Espera la confirmación de un movimiento específico desde el hardware.
- wait_for_angles_response: Espera la respuesta con los ángulos actuales de los joints del brazo.

Arquitectura del Sistema

Módulo Percepción.

Definición.

Este módulo se encarga de capturar imágenes y procesarlas para detectar objetos en el entorno del brazo robótico.

CameraManager:

- capture_image: Captura una imagen desde la cámara, la guarda en un archivo con una marca temporal y devuelve la ruta del archivo.

ImageProcessor:

- read_image_path: Lee una imagen desde una ruta, realiza la detección de objetos y, opcionalmente, dibuja los resultados en la imagen.
- process_image: Procesa la imagen usando el modelo YOLO para detectar objetos, filtrando por un umbral de confianza y seleccionando la mejor detección.
- _draw_detection: Dibuja las bounding boxes y etiquetas de los objetos detectados en la imagen.
- _save_drawn_image: Guarda la imagen procesada con las detecciones dibujadas en un nuevo archivo.

Arquitectura del Sistema

Módulo de Control del Brazo.

Definición.

Este módulo, implementado en el VEX Brain, controla directamente los motores y sensores del brazo robótico, procesando comandos y ejecutando servicios.

- CommunicationManager (VEX):
  - initialize: Inicializa el puerto serial para la comunicación con el sistema principal.
  - read_message: Lee mensajes JSON desde el puerto serial y los decodifica.
  - send_message: Envía mensajes JSON al sistema principal con información de estado o datos.
- SensorModule:
  - clear_screen: Limpia la pantalla del VEX Brain.
  - print_screen: Imprime texto en coordenadas específicas en la pantalla del Brain.
  - get_angle: Obtiene el ángulo actual del sensor inercial.
  - get_distance: Obtiene la distancia medida por un sensor de distancia (base o gripper).
  - get_object_size: Obtiene el tamaño del objeto detectado por un sensor de distancia.
  - is_bumper_pressed: Verifica si el bumper está presionado.
  - set_color: Establece el color del touchled según el estado del sistema (error, listo, etc.).
  - check_sensors: Verifica si todos los sensores necesarios están instalados.

Arquitectura del Sistema

Módulo de Control del Brazo.

Definición.

Este módulo, implementado en el VEX Brain, controla directamente los motores y sensores del brazo robótico, procesando comandos y ejecutando servicios.

- PerceptionModule:
  - process_sensor_distance: Procesa la distancia medida por un sensor para detectar objetos dentro de un rango específico.
- MappingModule:
  - process_object_detection: Procesa las detecciones de objetos para calcular su posición y tamaño, actualizando un mapa de objetos.
  - _save_object: Guarda la información de un objeto detectado en el mapa cuando termina su detección.
  - get_objects_map: Devuelve el mapa de objetos detectados y reinicia el mapa.
- ControlModule:
  - move_motor_to_angle: Mueve un motor (como el de la base) a un ángulo específico usando el sensor inercial.
  - get_position: Obtiene la posición actual de un motor en grados.
  - get_current: Obtiene la corriente consumida por un motor.
  - general_stop: Detiene todos los motores del brazo.
  - check_motors: Verifica si todos los motores están instalados.
- SafetyModule:
  - check_motors: Verifica la instalación de los motores.
  - check_sensors: Verifica la instalación de los sensores.
  - check_shoulder_safety: Monitorea y ajusta la seguridad del hombro, deteniendo o retrocediendo si el bumper está presionado.
  - gripper_action: Realiza acciones de seguridad con el gripper (abrir/cerrar) según el servicio en curso.

Arquitectura del Sistema

Módulo de Control del Brazo.

Definición.

Este módulo, implementado en el VEX Brain, controla directamente los motores y sensores del brazo robótico, procesando comandos y ejecutando servicios.

- RoboticServices:
  - run_service: Ejecuta servicios como check, safety o scan, coordinando las acciones de los otros submódulos.
  - reset_scan_variables: Reinicia las variables internas del servicio de escaneo.
  - _execute_start_scan: Inicia el proceso de escaneo moviendo el motor base.
  - _execute_scan_service: Ejecuta el servicio de escaneo, procesando datos de sensores y detectando objetos.
  - _pick_place_service: Gestiona los servicios de pick y place, ejecutando movimientos específicos para cada joint.
  - _execute_pick_place_sequence: Ejecuta la secuencia de movimientos para recoger o colocar un objeto.
  - process_message: Procesa mensajes recibidos del sistema principal y actualiza los estados del robot.
  - run: Bucle principal que lee mensajes y ejecuta los servicios correspondientes.

Comunicación

Protocolos de Comunicación

Un protocolo de comunicación es esencialmente un conjunto de reglas que determinan cómo los diferentes componentes de un sistema intercambian información. En nuestro caso, necesitamos que nuestro brazo robótico y nuestra computadora hablen el mismo idioma.
Profundicemos en los principales protocolos utilizados en robótica:

Comunicación Serial:

Transmite datos bit por bit a través de una única línea. Es como un camino de un solo carril donde los datos deben formar una fila y pasar uno tras otro. Es simple pero efectiva para muchas aplicaciones.

Comunicación I2C (Inter-Integrated Circuit):

Utiliza solo dos líneas: una para datos (SDA) y otra para la señal de reloj (SCL). Imagina una mesa redonda de negociaciones donde un moderador (maestro) da la palabra (reloj) a diversos participantes (esclavos) quienes comparten información cuando les toca su turno.

Comunicación SPI (Serial Peripheral Interface):

Utiliza cuatro líneas: MOSI (Master Out, Slave In), MISO (Master In, Slave Out), SCK (Clock) y SS (Slave Select). Es como un sistema de megafonía donde el administrador (maestro) selecciona a qué departamento (esclavo) hablar mediante una línea dedicada, y luego mantienen una conversación bidireccional exclusiva.

Comunicación CAN Bus (Controller Area Network):

Muy utilizado en automoción y robótica industrial. Piénsalo como un sistema de radio donde todos los dispositivos "escuchan" todos los mensajes, pero solo responden a los que llevan su "nombre" o identificador.

Comunicación

Protocolos de Comunicación

Comunicación Ethernet IP:

Para comunicaciones más robustas y a mayor escala. Es como tener una red de carreteras con múltiples carriles donde grandes cantidades de datos pueden viajar simultáneamente siguiendo reglas de tráfico establecidas.

Comunicación

Comunicación Serial.

Fundamentos:

- Piensa en la comunicación serial como enviar un tren de mercancías por una vía única. Cada vagón representa un bit, y todos deben pasar en secuencia ordenada.
- Al principio de cada tren (mensaje), enviamos un vagón especial llamado "bit de inicio" para alertar a la estación receptora.
- Al final, enviamos otro vagón especial, el "bit de parada", que indica "fin del mensaje".
- Si los trenes van demasiado rápido (baudrate incorrecto), la estación receptora no tendrá tiempo de descargar un vagón antes de que llegue el siguiente, creando confusión y pérdida de carga (datos).

Componentes esenciales:

- Puerto: El "nombre" del punto de conexión donde se conecta nuestro dispositivo..
- Baudrate: La velocidad de transmisión medida en bits por segundo. Usamos 115200 bps.
- Bits de datos: Generalmente 8 bits, forman un byte (un carácter).
- Bits de paridad: Opcional, pero útil para detectar errores.
- Bits de parada: Indica el fin de un byte.
- Control de flujo: Mecanismos como RTS/CTS o XON/XOFF que evitan la sobrecarga de datos.
- Timeout: Cuánto tiempo esperamos antes de considerar que la comunicación ha fallado.

Comunicación

Protocolo JSON.

Fundamentos:

Piensa en JSON como un formulario estandarizado universal. Cuando diferentes departamentos gubernamentales necesitan intercambiar información, utilizan formularios con campos predefinidos donde cada campo tiene un nombre claro (la clave) y un espacio para rellenar (el valor). Cualquier funcionario puede leer estos formularios independientemente de su departamento porque siguen una estructura común y organizada.

Componentes esenciales:

- **Estructura clara y legible:** JSON organiza la información en pares clave-valor, similar a un diccionario en Python. Como nuestra analogía del formulario, cada campo tiene un nombre (clave) y un contenido (valor).
- **Flexibilidad en los tipos de datos:** Podemos enviar diferentes tipos de información sin cambiar el formato:
  - **Números enteros (posiciones de motores):** "position": 90
  - **Números decimales (mediciones precisas):** "distance": 24.7
  - **Texto (mensajes de estado):** "status": "SUCCESS"
  - **Listas (secuencias de posiciones):** "waypoints": [10, 45, 90]
  - **Objetos anidados (configuraciones complejas):** "motor_config": {"speed": 50, "acceleration": 10}
- **Compatibilidad universal:** Prácticamente todos los lenguajes de programación tienen bibliotecas para manejar JSON, lo que facilita la comunicación entre diferentes sistemas (Python en la Raspberry, C++ en el VEX Brain).
- **Validación sencilla:** Es fácil verificar si un mensaje JSON tiene la estructura esperada, lo que nos ayuda a detectar errores rápidamente.
- **Tamaño compacto:** Ocupa menos espacio que XML y otros formatos, lo que es importante en comunicaciones seriales con ancho de banda limitado.

Comunicación

Protocolo JSON.

Estructura personalizada:

- Tipo de mensaje (messageType): Identifica qué servicio o acción se está solicitando.
  - Ejemplos: "MOVE_BASE", "MOVE_ARM", "GRAB_OBJECT", "GET_SENSOR"
- Datos (data): Contiene la información necesaria para ejecutar la acción.
  - Para "MOVE_BASE": ángulos de los motores
  - Para "GET_SENSOR": qué sensor consultar
  - Para "GRAB_OBJECT": fuerza de agarre

```json
{
  "messageType": "MOVE_BASE",
  "data": {
    "motorId": 1,
    "position": 90,
    "speed": 50
  }
}
```

Visión artificial

1. Visión Artificial en Robótica.

1.1 ¿Qué es la visión artificial?

La visión artificial es una disciplina de la inteligencia artificial que permite a las máquinas interpretar y extraer información del mundo visual a través de imágenes o videos. En robótica, actúa como el "sistema sensorial" que habilita la percepción del entorno, combinando hardware (cámaras) y software (algoritmos de procesamiento y aprendizaje profundo).

- Componentes básicos:Cámara: Captura fotogramas como entrada visual (resolución, tasa de frames, campo de visión).
- Procesador: Ejecuta algoritmos para analizar los datos visuales (CPU, GPU, o unidades especializadas como NPUs).

1.2 Aplicaciones en robótica.

- **Navegación y localización:**
  - Concepto: Los robots construyen mapas del entorno (SLAM: Simultaneous Localization and Mapping) usando cámaras y sensores LiDAR. Las imágenes se procesan para detectar puntos clave (keypoints) y estimar la pose del robot mediante algoritmos como ORB o SIFT.
  - Técnica: La visión estéreo o monocular genera mapas 3D al triangular distancias, mientras que el filtrado de Kalman o el bundle adjustment optimizan la localización.
  - Ejemplo: Un robot aspirador ajusta su trayectoria al detectar obstáculos en tiempo real.

Visión artificial

1.2 Aplicaciones en robótica.

- **Manipulación y ensamblaje:**
  - Concepto: La visión artificial calcula la posición 6D (x, y, z, roll, pitch, yaw) de objetos mediante detección y segmentación, permitiendo a los brazos robóticos adaptarse a variaciones.
  - Técnica: Redes neuronales convolucionales (CNNs) identifican objetos y sus propiedades, mientras que algoritmos de transformación (como la matriz de homografía) ajustan las coordenadas al sistema del robot.
  - Ejemplo: Un brazo robótico ensamblando componentes electrónicos con tolerancia a posiciones aleatorias.

- **Interacción humano - robot:**
  - Concepto: La visión artificial interpreta señales visuales humanas (rostros, gestos) para comandos o seguridad.
  - Técnica: Modelos como Haar Cascades o CNNs preentrenadas (ej. FaceNet) detectan rostros, mientras que redes como MediaPipe analizan gestos en tiempo real.
  - Ejemplo: Un robot humanoide ajustando su comportamiento según el lenguaje corporal del usuario.

2. Fundamentos del procesamiento de imágenes.

2.1 Representación de imágenes digitales.

Una imagen digital es un tensor tridimensional:

- Dimensiones: Alto (H) × Ancho (W) × Canales (C).
  - Ejemplo: Una imagen RGB de 1280×720 tiene H=720, W=1280, C=3.
- Valores: Cada píxel contiene intensidades (0-255 en 8 bits) por canal (R, G, B) o un solo valor en escala de grises.

Visión artificial

2.2 Técnicas de preprocesamiento.

- El preprocesamiento adapta las imágenes a las entradas esperadas por los modelos de visión artificial, optimizando precisión y eficiencia computacional.
  - **Redimensionamiento:**
    - Método: Interpolación bilineal o bicúbica para ajustar la resolución (ej. de 1280×720 a 640×640).
    - Impacto: Reduce la carga computacional al disminuir el número de píxeles procesados.
  - **Conversión de Color:**
    - Método: Transformación de RGB a escala de grises: I=0.299R+0.587G+0.114B I = 0.299R + 0.587G + 0.114B I=0.299R+0.587G+0.114B.
    - Impacto: Reduce los canales de 3 a 1, disminuyendo el tamaño del tensor en un factor de 3.
  - **Normalización:**
    - Método: Escalar valores de [0, 255] a [0, 1] o estandarizar con media y desviación (x′=(x−μ)/σ x' = (x - \mu) / \sigma x′=(x−μ)/σ).
    - Impacto: Mejora la convergencia de las redes neuronales al homogeneizar los datos.
  - **Reducción de Ruido:**
    - Método: Filtros como Gaussian Blur (G(x,y)=12πσ2e−x2+y22σ2 G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2 + y^2}{2\sigma^2}} G(x,y)=2πσ21e−2σ2x2+y2).
    - Impacto: Elimina artefactos que podrían confundir al modelo.
  - **Umbralización:**
    - Método: Aplicar un umbral T T T para binarizar: I(x,y)=255 I(x, y) = 255 I(x,y)=255 si I(x,y)>T I(x, y) > T I(x,y)>T, 0 0 0 si no.
    - Impacto: Simplifica la imagen para tareas como segmentación.
  - **Detección de Bordes:**
    - Método: Operadores como Sobel o Canny para resaltar gradientes (Gx=∂I∂x,Gy=∂I∂y G_x = \frac{\partial I}{\partial x}, G_y = \frac{\partial I}{\partial y} Gx=∂x∂I,Gy=∂y∂I).
    - Impacto: Enfatiza contornos, útil para detección de formas.

Visión artificial

3. Detección de objetos con YOLO.

3.1 Fundamentos de YOLO.

YOLO (You Only Look Once) es una familia de CNNs que realiza detección de objetos en una sola pasada (single forward pass), tratando la tarea como un problema de regresión:

- Salidas: Predice bounding boxes (x,y,w,h x, y, w, h x,y,w,h), confidence scores (P(objeto) P(objeto) P(objeto)) y clases (P(clase ∣ objeto) P(clase|objeto) P(clase ∣ objeto)).
- Arquitectura: Divide la imagen en una cuadrícula (ej. 7×7 en YOLOv1) y predice múltiples cajas por celda, refinadas con Non-Maximum Suppression (NMS).

3.2 Variantes de YOLO.

- YOLOv11 (2025): Incluye versiones como YOLOv11X (extra grande), YOLOv11L (grande), YOLOv11M (mediano), YOLOv11S (pequeño) y YOLOv11N (nano).
    - Diferencias: Equilibrio entre precisión (mAP) y velocidad (FPS). YOLOv11N es ideal para dispositivos edge como la Raspberry Pi 5 por su bajo costo computacional.

3.2 Formatos de despliegue.

- Onnx (Open Neural Network Exchange):
    - Descripción: Formato interoperable que convierte modelos de frameworks como PyTorch o TensorFlow a un estándar universal.
    - Ventajas: Portabilidad y compatibilidad con múltiples motores de inferencia.
    - Proceso: Exportación con torch.onnx.export() y optimización con ONNX Runtime.

Visión artificial

3.2 Formatos de despliegue.

- TensorRT:
  - Descripción: Framework de NVIDIA para optimizar modelos en GPUs.
  - Técnicas:
    - Cuantización: Reduce precisión (FP32 → FP16 o INT8) con calibración.
    - Fusión de capas: Combina operaciones (ej. Conv + BN + ReLU) en un solo kernel.
    - Selección de kernels: Usa algoritmos optimizados para CUDA.
  - Limitación: No aplicable a la Raspberry Pi (sin GPU NVIDIA).
- OpenVINO:
  - Descripción: Kit de Intel para CPUs, iGPUs y VPUs (Movidius).
  - Técnicas: Optimización de grafos y soporte para INT8.
  - Limitación: Menos relevante para ARM en la Raspberry Pi.
- MNN y NCNN:
  - Descripción: Frameworks ultraligeros para ARM y dispositivos edge (ver sección 4).

Visión artificial

4. Formato NCNN y MNN.

4.1 Hardware de la Raspberry PI 5.

- CPU: ARM Cortex-A76, 64 bits, 4 núcleos, 2.4 GHz, con soporte NEON (SIMD).
- GPU: VideoCore VII, compatible con OpenGL ES.
- RAM: LPDDR4X, hasta 8 GB.
- Limitación: Sin NPU/TPU dedicada, depende de CPU y GPU generalista.

4.2 Características de NCNN y MNN

- NCNN (Tencent): Framework de inferencia optimizado para ARM y móviles.
- MNN (Alibaba): Similar, con énfasis en eficiencia y flexibilidad.
- Filosofía: Ultraligeros (MB vs. GB de TensorFlow), enfocados en inferencia.

4.3 Optimizaciones clave.

- Instrucciones ARM NEON (SIMD):
  - Definición: Extensión SIMD que procesa vectores de datos (4×32 bits, 8×16 bits, etc.) en paralelo.
  - Uso: Acelera operaciones matriciales (ej. convoluciones: $y=w*x+b$ y = w * x + b $y=w*x+b$) al vectorizar cálculos.
  - Ejemplo: Una convolución 3×3 en una imagen se ejecuta 4 veces más rápido con NEON.

- Frameworks livianos:
  - Tamaño: NCNN (~1-2 MB), MNN (~2-3 MB) vs. TensorFlow (~GB).
  - Optimización: Código en C++ y assembly, sin dependencias innecesarias, compilación selectiva.

- Aceleración GPU con OpenGL ES:
  - Método: Mapea operaciones (ej. convoluciones) a shaders de GPU.
  - Implementación: NCNN/MNN convierten tensores en texturas y ejecutan kernels paralelos.
  - Beneficio: Descarga la CPU, aprovechando los ~100 núcleos de la VideoCore VII.

Visión artificial

4.3 Optimizaciones clave.

- Optimización de Memoria:
  - Memory Pooling: Preasigna bloques de memoria reutilizables, reduciendo overhead de malloc/free.
  - Operator Fusion: Combina operaciones (ej. Conv + ReLU) en un solo paso, minimizando escrituras intermedias.
  - Ejemplo: Reduce accesos a RAM de O(n) O(n) O(n) a O(1) O(1) O(1) por capa.

- Cuantización:
  - Definición: Reduce precisión (FP32 → FP16 o INT8).
  - Proceso:
    - Post-entrenamiento: Escala pesos/activaciones con xq=round(x/s) x_q = \text{round}(x / s) xq=round(x/s), donde s s s es un factor de escala.
    - Durante entrenamiento: Quantization-Aware Training (QAT) simula baja precisión.
- Impacto:
  - Tamaño: FP32 (4 bytes) → INT8 (1 byte) = 4x menor.
  - Velocidad: Menor ancho de banda de memoria, mejor uso de caché.
  - Precisión: Pérdida mínima (<1-2% mAP) con calibración adecuada.

- Enfoque en ARM/Linux:
  - Compatibilidad: Diseñados para Raspberry Pi OS, con soporte nativo para Cortex-A76.

Visión artificial

4.4 NCNN vs MNN

- Diferencias:
  - NCNN: Mayor énfasis en móviles, soporte Vulkan opcional.
  - MNN: Más flexible, soporta más formatos de entrada (ONNX, Caffe).
- Criterios de elección: Rendimiento (FPS), soporte de operadores, facilidad de conversión.

Geniia .com

Explicación completa en nuestro canal de YouTube:

¡Click aqui!

Gracias por visitar nuestro repositorio y esperamos que disfrute trabajando con nuestro codigo.