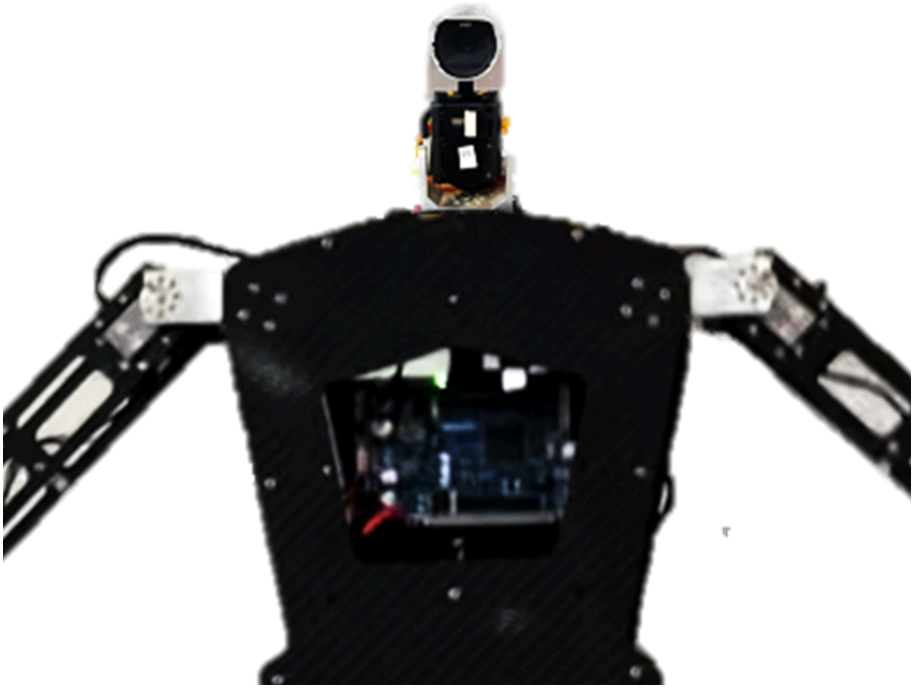
Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Estado de México

Robot Humanoide Bogobot 3

Visión Computacional



Unidad de Formación Integración de Robótica y Sistemas Inteligentes (TE3003B.501) a cargo del Dr. Alejandro Aceves López, del Dr. Miguel Ángel Gálvez, del Dr. Fransisco Ortiz Cerecedo y del Dr. Arturo Vargas Olivares.

Realizado por Rodrigo Mejía Jiménez y Alex Federico Núñez Escobar

1.0 Sistema operativo BOGO3

La Fitlet1 es un mini-PC compacto y robusto diseñado para aplicaciones industriales y sistemas embebidos. Esta plataforma es ideal para proyectos de visión por computadora y robótica debido a su capacidad de procesamiento eficiente y su tamaño reducido, lo que facilita su integración en entornos diversos y restringidos como en este caso, integrarlo al robot humanoide. A continuación, se presenta una descripción detallada del entorno de software utilizado en nuestro proyecto de detección de balones empleando la Fitlet1.

Entorno de Software

1. **Sistema Operativo:**
    - **Ubuntu 18.04.6 LTS (Long Term Support):** La versión 18.04 de Ubuntu ofrece un entorno estable y bien soportado, ideal para aplicaciones que requieren alta confiabilidad y soporte a largo plazo. Es compatible con una amplia gama de librerías y herramientas necesarias para el desarrollo de aplicaciones de visión por computadora y robótica.
2. **Sistema de Robótica:**
    - **ROS (Robot Operating System) Melodic Morenia:** Esta versión de ROS proporciona un conjunto completo de herramientas y librerías para el desarrollo de software robótico. Melodic es una de las distribuciones de ROS más utilizadas y soportadas, especialmente en proyectos de larga duración.
3. **Librerías Utilizadas:**
    - **OpenCV (CV2) versión 3.2.0:** OpenCV es una librería de visión por computadora ampliamente utilizada que proporciona una gran cantidad de funciones para el procesamiento de imágenes y videos. La versión 3.2.0 es estable y ofrece compatibilidad con diversas herramientas y algoritmos necesarios para la detección de objetos.
    - **rospy versión 1.14.13:** rospy es la librería de ROS para Python, que permite la integración y comunicación entre diferentes nodos en un sistema robótico. La versión 1.14.13 es compatible con ROS Melodic y proporciona una interfaz robusta para el desarrollo de aplicaciones en Python.
    - **numpy versión 1.13.3:** NumPy es una librería fundamental para la computación numérica en Python. Proporciona soporte para arreglos y matrices multidimensionales, así como una amplia colección de funciones matemáticas de alto nivel, lo cual es crucial para el procesamiento eficiente de datos en aplicaciones de visión por computadora.

4. **Lenguaje de Programación:**
    - **Python versión 2.7.17:** Aunque Python 2.7 ha sido descontinuado, muchas librerías y sistemas heredados, como ROS Melodic, aún dependen de esta versión. Python 2.7.17 es una versión madura y estable, adecuada para el desarrollo de aplicaciones que no requieren las características más recientes de Python 3.

2.0 Visualización de la Pelota

La visualización de imágenes es un componente fundamental en los sistemas de visión por computadora, ya que permite interpretar, analizar y verificar los resultados obtenidos durante el procesamiento de imágenes. Como en este caso que se requiere visualizar el balón de fútbol y realizar su correcto análisis.

La capacidad de visualizar las imágenes procesadas y los objetos detectados es crucial por varias razones. Primero, permite identificar y corregir posibles errores en el algoritmo de detección, como falsos positivos o negativos. Segundo, la visualización ayuda a ajustar y calibrar los parámetros del sistema para mejorar su precisión y robustez. Tercero, buscar maneras de mejorar el programa porque nos permite entender y analizar su comportamiento.

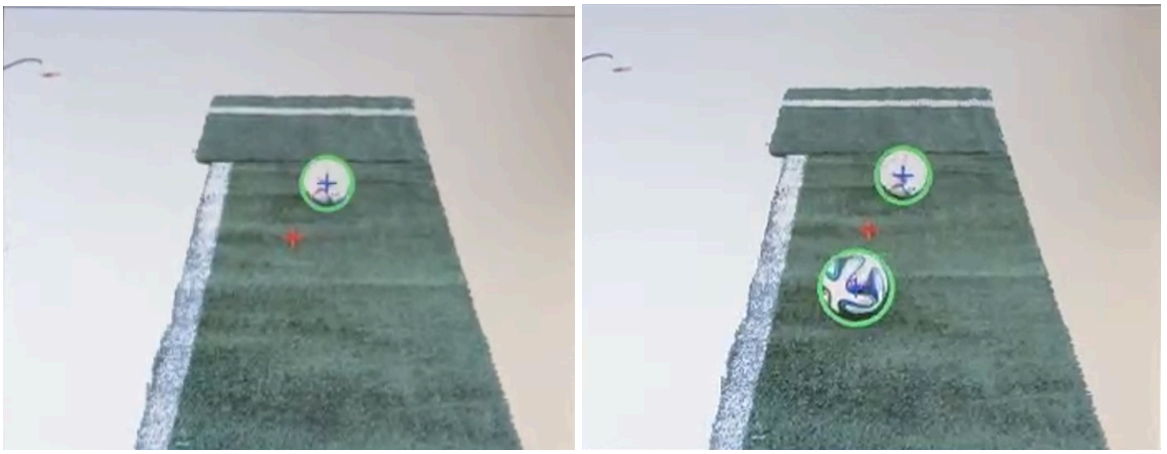


Figura #. Balones detectados

2.1 Preprocesamiento

El diagrama de flujo representa el proceso de detección de un balón utilizando técnicas de visión por computadora. El proceso comienza con la captura de video en tiempo real mediante una cámara externa, que proporciona un flujo continuo de imágenes a ser procesadas. Inicialmente, se ajusta el brillo de las imágenes capturadas para mejorar la visibilidad y resaltar los detalles importantes.

Posteriormente, se aplica un filtro de color verde para aislar y eliminar el fondo verde del campo, permitiendo que los objetos no verdes, como el balón, permanezcan visibles. Una vez filtrada la imagen, esta se convierte a escala de grises para simplificar el procesamiento y enfocarse en las intensidades de luz en lugar de la información de color.

La imagen en escala de grises se suaviza utilizando un desenfoque Gaussiano, lo que ayuda a reducir el ruido y eliminar detalles menores, facilitando la detección de bordes y formas. A continuación, se aplica una umbralización adaptativa para convertir la imagen suavizada en una imagen binaria, donde los píxeles de interés se destacan claramente contra el fondo.

Para mejorar la claridad y reducir el ruido, la imagen binaria pasa por un proceso de erosión, eliminando pequeños puntos blancos dispersos y limpiando la imagen. Luego, se aplica una dilatación para expandir las áreas blancas, reforzando las características de interés y haciendo que los bordes del balón sean más prominentes.

Finalmente, se utiliza la Transformada de Hough para detectar círculos en la imagen procesada. Esta técnica identifica los bordes circulares que representan al balón y marca su posición y tamaño en la imagen original.

Este enfoque secuencial y estructurado asegura un sistema robusto y eficiente para la detección precisa del balón, ideal para aplicaciones en tiempo real como la robótica y el análisis deportivo, donde la detección y el seguimiento de balones son esenciales. Además que da robustez y seguridad al estar detectando un objeto en una superficie irregular como el pasto.

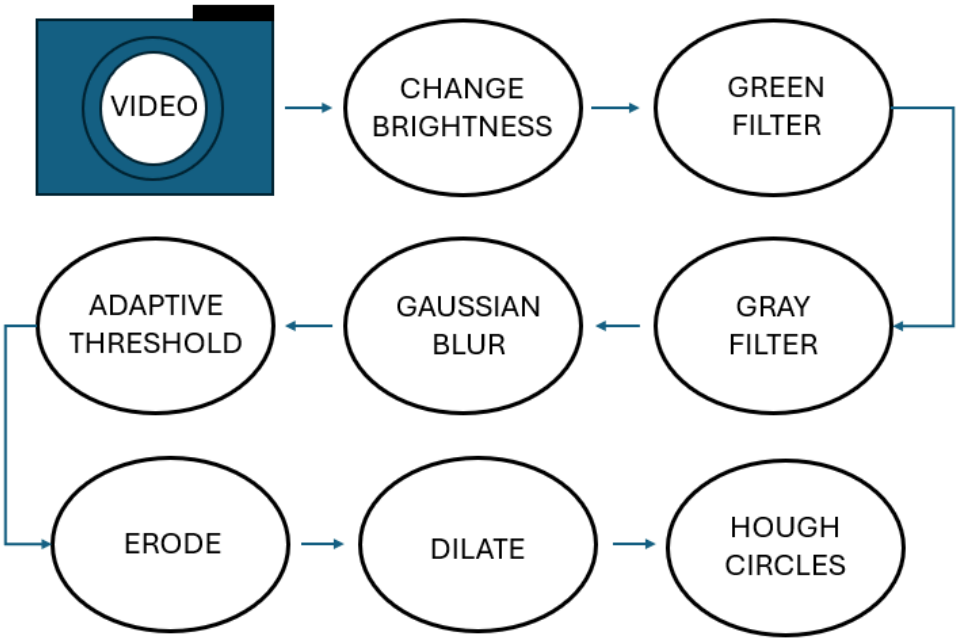


Figura #. Diagrama de flujo procesamiento de imagen

3.0 Controlador

Esta parte calcula el error entre el centro de cada círculo detectado y el centro de la cámara y se utiliza un controlador PID para ajustar la posición de los motores basándose en este error, con el objetivo de centrar la cámara en el círculo.

4.0 Problemas de sistema

Existen ciertos problemas en la detección cuando el balón se mueve muy rápido al igual que el controlador evita que gire tan rápido la cámara, por otra parte, el tamaño del balón es importante para una correcta funcionalidad del programa

5.0 Trabajo Futuro

El sistema de detección de balones desarrollado ha mostrado ser efectivo en condiciones controladas, pero existen varias áreas en las que se puede mejorar para incrementar su robustez, eficiencia y aplicabilidad en entornos más dinámicos y desafiantes. A continuación, se describen las direcciones clave para el trabajo futuro:

1. Mejora de la Visión para Reducción de Ruido y Frecuencia de Detección:

- Objetivo: Aumentar la precisión del sistema al reducir el ruido en las imágenes y mejorar la frecuencia de detección.
- Acciones Propuestas:
    - Implementar técnicas avanzadas de filtrado y procesamiento de imágenes, como el uso de filtros adaptativos y técnicas de aprendizaje profundo para segmentación de objetos.
    - Integrar algoritmos de seguimiento de objetos que permitan una detección continua y más fiable en presencia de variaciones de iluminación y movimiento rápido.
    - Utilizar técnicas de fusión de sensores para combinar datos de múltiples fuentes (por ejemplo, cámaras RGB-D y sensores de profundidad) para mejorar la robustez de la detección.

2. Mejora del Controlador para Mayor Estabilidad:

- Objetivo: Desarrollar un controlador más estable que pueda manejar mejor las variaciones en la detección y las condiciones del entorno.
- Acciones Propuestas:
    - Aplicar técnicas de control avanzado, como controladores predictivos basados en modelos (MPC) o controladores adaptativos, que puedan ajustarse dinámicamente a los cambios en el entorno y en las condiciones
    - Implementar estrategias de control tolerante a fallos que permitan al sistema mantener su funcionalidad incluso en presencia de detecciones erróneas o pérdida temporal de la señal.
    - Realizar pruebas exhaustivas en diferentes escenarios y condiciones para ajustar y calibrar los parámetros del controlador, garantizando una respuesta rápida y estable.

3. Mejora de la Metodología del Código para Evitar Redundancias:

- Objetivo: Optimizar la estructura y la eficiencia del código eliminando redundancias y mejorando la modularidad.
- Acciones Propuestas:
    - Realizar una revisión y refactorización del código para identificar y eliminar operaciones redundantes, asegurando que cada función y módulo del sistema se utilice de manera eficiente.
    - Adoptar principios de diseño de software como la programación modular y orientada a objetos, lo que facilitará la reutilización del código y la integración de nuevas funcionalidades.
    - Implementar pruebas automatizadas y herramientas de análisis estático para asegurar la calidad del código y detectar posibles errores o ineficiencias de manera temprana.

3. Comunicación:

- Objetivo: Conectar la parte de visión con la caminata del robot
- Acciones Propuestas:
    - Integrar el controlador del movimiento del robot con el controlador de la cámara para que el robot siga la pelota.