Sistema de Clasificación de Objetos con Visión por Computadora

INNOW Technology Fest 2024

Colegio Científico del Atlántico

Profesora Elkira Francis Hernández

Justin Lacayo Picado	Grevanny Quintana Manning	Martin Rojas Barrios
Lacayopicado@gmail.com	mahia092008@gmail.com	marfran1904@gmail.com

1. Resumen Ejecutivo

1.1. Objetivo del Proyecto

El propósito de este proyecto es permitir que un sistema electrónico programable llamado Raspberry pi 4 realice un proceso de visión por computadora que sea capaz de verificar el estado de un objeto esto con el fin de mejorar diversos procesos de manufactura, logrando con ello una mejor eficiencia en la automatización de procesos. Por esto mismo, nuestra problemática a resolver es la creación de un código eficiente que cumpla con todas estas expectativas. Debido a estas razones nuestra solución propuesta es utilizar múltiples funciones que se le pueden realizar a la imagen ya sea umbralización, saturación, Canny, entre otros, con el fin de detectar la forma y el color de un objeto, consiguiendo así la clasificación de este y una buena velocidad de respuesta del código.

1.2. Descripción del Sistema

En este sistema la clasificación de objetos se resume en determinar todos los contornos, los bordes y los colores predominantes de un objeto que se encuentra enfocado por una cámara y colocado en un fondo blanco para que con estas condiciones y esta información se descubra que tipo de objeto es y si el objeto metálico en cuestión esta corroído o con oxido.

2. Introducción

2.1. Antecedentes

La visión por computadora ha sido un avance revolucionario que va venido en desarrollo desde 1960 cuando se intentó replicar la vista humana de manera artificial, pero fue hasta en 2014 que dio un saltó revolucionario con la tecnología “deep learning” al lograr analizar un total de 15 millones de imágenes, Ahora bien en 2016 con ayuda de redes neuronales convulsiónales multicapa (CNN) este proceso se volvió casi en tiempo real y en 2020 el avance ya es tal, que se es capaz de tener visión en dispositivos con un hardware de menor coste. Esta tecnología representa un gran avance puesto que tiene múltiples ventajas: análisis médicos, reconocimiento facial, automatización de procesos, detección de errores en procesos de manufactura, por todo esto es que esto representa una gran importancia y avance tecnológico en el mundo.

2.2 Descripción del Problema

La necesidad de clasificar objetos en 2 categorías “buen estado” y “defectuoso”, se encuentra ligada a el propósito de mejorar los procesos de manufactura, logrando con esta visión por computadora detectar si en el proceso se produjo un objeto con errores no deseados. Lo anterior ocasiona que estos procesos sean más eficientes y por su parte ayuda a evitar la importación de objetos dañados, además de que con esto se podría rastrear el origen de estos errores. Es claro que esta tecnología tampoco es muy fácil de usar e implementar dado que existen muchos retos a superar: la lógica del código para la clasificación de los objetos, la eficiencia, velocidad de respuesta, hardware requerido. Mas, sin embargo, todos estos desafíos se pueden superar con relativa facilidad y los beneficios obtenidos son muy prometedores.

2.3. Justificación

Puesto que en este trabajo se realiza un procedimiento de visión por computadora, que logra clasificar objetos ya sea en “buen estado” o “mal estado”, se puede determinar que esto resulta muy relevante ya que con esa tecnología se da lugar a una mejora en la automatización de procesos y en el apartado de control de calidad. Por lo que para este proyecto se espera que la categorización de objetos sea precisa, adaptable y escalable para ser usada en ciertos procesos de manufactura.

3. Especificaciones Técnicas

3.1. Hardware Utilizado:

- Raspberry Pi 4 Modelo B.
- Cámara web Lauguham.
- Lampara.

3.2. Software Utilizado:

- OpenCV versión 4.10.0
- Python versión 3.12.2
- Librerías adicionales Numpy y Time
- Visual Studio Code 1.92

4. Descripción del Algoritmo

4.1. Proceso de Captura de Imágenes

Al iniciar el programa se ejecuta un bucle que obtiene cada fotograma de video proporcionado por la cámara web, si es que está conectada, y cada vez que transcurre el tiempo definido uno de los fotogramas es utilizado para comenzar el proceso de verificación.

4.2. Preprocesamiento de Imágenes

Cada fotograma evaluado comienza siendo recortado para centrar la imagen y disminuir la cantidad de pixeles innecesarios con los que se van a trabajar, después esta imagen es aumentada debido a la distancia de la cámara al punto focal y recortada de nuevo para mantener la imagen centrada en el objeto. A este fotograma recortado luego se le genera una copia, pero convertida a escala de grises para después ser umbralizada y también para aplicarle a otra copia diferente un filtro Canny.

4.3. Extracción de Características

En este proyecto se definen ciertas funciones cuyo propósito es describir ciertas características del objeto como sus contornos los cuales son determinados gracias al fotograma convertido a escala de grises y umbralizado, al que después se le obtienen todos sus posibles contornos y de ellos se escoge al contorno de mayor área como contorno principal pero también se evalúa si hay algún contorno interior, el cual es determinado solo si existe algún contorno que se encuentre dentro del contorno principal y aparte supere el límite del área deseada, gracias a estos contornos se genera una máscara que luego es aplicada en una copia del fotograma original para quitar el fondo blanco.

Aparte de determinar los contornos también se obtienen la forma del objeto a evaluar, su procedimiento es dividido en dos funciones una que detecta todos los bordes gracias a la imagen en escala de grises evaluada por filtro Canny y otra en la cual esos bordes son simplificados para comparar la forma del objeto con una forma geométrica simple, con ello se determina si el objeto se trata de una tuerca, un tornillo o una arandela.

Otras características que se obtienen son los colores predominantes los cuales son determinados por la cantidad de pixeles que tienen ese color en la imagen con la máscara aplicada, los colores obtenidos de esta función son una lista que va desde el de mayor aparición hasta el de menor aparición, todos los colores se encuentran en dos versiones, su versión en formato BGR y su versión en HSV para su próxima manipulación.

4.4. Clasificación de Objetos

Este programa se encarga de clasificar ciertos objetos metálicos de acuerdo con varios parámetros preestablecidos, con estos mismos se catalogan los objetos a través de la forma, gracias a los bordes y al conteo de los vértices del objeto simplificado logrando diferenciar tuercas, tornillos y arandelas. Otra forma de clasificación es que a través de los colores predominantes se determina si el objeto metálico esta oxidado.

5. Innovación y Creatividad

5.1. Características Adicionales Implementadas

En el proyecto aparte del código principal que esta optimizado para su utilización en la Raspberry pi 4, se agregó una carpeta llamada otros_códigos donde se encuentran el mismo código modificado con 2 versiones disponibles: Una versión con la interfaz gráfica simple y otro empleando una detección por IA.

5.2. Impacto en el Proyecto

Estas 2 modificaciones se crearon para mostrar que el programa puede ser adaptable y escalable dependiendo las necesidades y recursos que se dispongan. Por ejemplo, si se requiere de una buena eficiencia y se tiene de un buen hardware se utiliza visión con IA ya que muestra un menor margen de error, no obstante, si se prefiere un programa más fácil de usar, intuitivo y amigable a la vista, se utiliza el código con interfaz gráfica, que requiere de un consumo solo un poco más grande que la versión original dado a la complejidad del diseño que pese a ser mínima aun así consume almacenamiento. Esto significa una mejora en la eficiencia, velocidad de ejecución y lo intuitivo que es código.

6. Resultados

6.1. Pruebas realizadas

Para comprobar la eficiencia del código se realizaron múltiples pruebas en escenarios controlados, los cuales eran, un fondo blanco, el objeto encima de este y una lampara como fuente de iluminación. Con estas condiciones el código al ejecutarse se obtuvieron los siguientes resultados: El código era capaz de abrir ventanas con la información del objeto abajo de la cámara con fallos casi nulos.

6.2. Desempeño del sistema

Luego de todas las pruebas realizadas se obtuvo una gran eficiencia, está siendo mayor al 95%, el código fue capaz de detectar el nombre del objeto, estado y color de mismo. Con esto se concluye que este mismo fue capaz de cumplir con las expectativas esperadas.

7. Conclusiones

7.1. Éxitos y desafíos

Al finalizar este trabajo se logró obtener un código funcional, con una gran fiabilidad y eficiencia al clasificar los objetos. Con respecto a logros personales, nos sentimos gratificados por toda la experiencia y conocimiento obtenido, ya que logramos desarrollar un código con múltiples funciones que antes no conocíamos. Sin embargo, se presentaron múltiples desafíos que supimos manejar, uno de los principales fue trabajar en un código eficiente para un hardware como la Raspberry pi 4 y el segundo fue aprender a usar las múltiples funciones que se necesitaban para la clasificación de los objetos.

7.2. Aprendizajes

En todo este trabajo en conjunto logramos aprender demasiadas cosas relacionadas con la visión computacional, a trabajar y realizar códigos eficientes por las limitantes de hardware, las múltiples funciones para lograr la mejor imagen para la detección de un objeto. Por otra parte, en los 2 códigos adicionales se aprendió a desarrollar interfaces gráficas, entrenar y realizar clasificaciones de objetos gracias a la inteligencia artificial.

7.3. Posibles mejoras

Algunas mejoras que se pueden implementar a nuestro código son: más objetos a clasificar, pulir mejor la detección de bordes del procedimiento Canny y aumentar la cantidad de condiciones que indican el estado final del objeto.

7.4. Referencias

OpenCV: OpenCV-Python Tutorials. (s. f.). https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html

Ultralytics. (2024, julio). *YOLOV8*. Ultralytics YOLO Documentos. https://docs.ultralytics.com/es/models/yolov8/

Google Colab. (s. f.). https://colab.research.google.com/

Interfaces gráficas de usuario con Tk. (s. f.). Python Documentation. https://docs.python.org/es/3/library/tk.html

8. Anexos

8.1. Manual de Usuario

8.1.1. Instalación

a) Preparación del Entorno

- Se prepara la Raspberry pi 4 con el código y la cámara conectada al mismo.
- La cámara debe tener una buena altura para que logre enfocar al objeto.
- El objeto para ser detectado tiene que estar sobre un fondo blanco.

b) Instalación del Software

- Primeramente, se abre la terminal del sistema.
- En esta terminal se instala opencv en Windows: pip install opencv-python y en linux: sudo apt-get opencv-python.
- Luego de esto, la librería procederá a instalarse, cuando esto se complete saldrá un mensaje que lo confirme.

8.1.2. Configuración

a) Configuración del Sistema

- Para lograr ejecutar el código no es necesaria ninguna configuración especial en la configuración del sistema operativo. Por otra parte, se instaló el Visual Studio Code en la Raspberry para una mayor comodidad a la hora de trabajar con el código, Sin embargo, este puede ser ejecutado desde el propio terminal del sistema operativo.

8.1.3. Uso del Sistema

a) Inicio del Sistema

- Para proceder a iniciar el proyecto solo se tiene que abrir el archivo Python ya sea desde la terminal de Linux o ejecutándolo desde algún programa como Visual Studio Code.

b) Captura y Clasificación de Imágenes

- Al iniciar el programa se comenzarán a recibir los fotogramas si es que hay una cámara conectada.
- En la pantalla se abrirá automáticamente una ventana para previsualizar la cámara y en el momento que se detecte un objeto se abrirán las dos ventanas más, una indicando el estado del objeto y otra indicando la cantidad de colores predominantes.

c) Interpretación de Resultados:

- Si se abren una ventana de color verde con la frase “En buen estado” indica que el objeto que está evaluando está en buen estado. Pero si se abre una ventana en color rojo con la frase “En mal estado” indica que el objeto que está detectando está en mal estado. Aunque si por el contrario no hay ninguna ventana abierta eso indica que no se está detectando ningún objeto.

d) Solución de Problemas Comunes

- Error en la clasificación de los objetos: Este error ocurre por 2 motivos, o está mal centrado el objeto con respecto a la cámara o posee muy poca o por el contrario una excesiva iluminación. La solución es identificar cual de estos genera el problema y corregirlo.
- Error en el código por falta de cámara: Este error ocurre cuando está mal conectada la cámara al dispositivo o la cámara especificada en el código “VideoCapture(0)” no es la que el dispositivo utiliza. La solución sería verificar las conexiones o cambiar el valor de 0 de “VideoCapture(0)” hasta encontrar la cámara que se desea utilizar.