ISSN-E: 2697-3650

Minerva Journal
Vol.4, Issue N°12, (pp. 9-20)

https://doi.org/10.47460/minerva.v4i12.132

Sistema automatizado de conteo de objetos en imágenes digitales

Rivas Maikol
maikol.14rivas@gmail.com
https://orcid.org/0009-0002-8525-8723
Universidad Nacional Experimental Politécnica
Antonio José de Sucre
Departamento de Ingeniería Electrónica
Puerto Ordaz-Venezuela

Lobo Eladio
cruzchiquita123@gmail.com
https://orcid.org/0000-0001-7921-2433
Universidad Nacional Experimental Politécnica
Antonio José de Sucre
Departamento de Ingeniería Electrónica
Puerto Ordaz-Venezuela

Recibido (21/04/2023), Aceptado (12/07/2023)

Resumen: En los inventarios, el contexto del conteo de objetos y la rapidez de respuesta se vuelve esencial. Cuando se trata de pocos elementos resulta más sencillo el proceso, pero su complejidad radica cuando son muchos, están cercanos o superpuestos. Además, la presión por cumplir con plazos ajustados para realizar inventarios agrega otro desafío. El objetivo de este trabajo fue desarrollar un programa que automatice el conteo preciso de objetos similares en imágenes digitales, sin importar su cantidad o disposición. Para ello se ha implementado un algoritmo en Python. Los principales resultados muestran eficiencia para analizar imágenes con objetos similares, un paso significativo hacia la separación y el conteo preciso de objetos adyacentes en diversos campos científicos. Esta solución promete simplificar y agilizar el proceso de conteo en imágenes digitales, con potenciales aplicaciones beneficiosas en múltiples disciplinas científicas.

Palabras clave: Automatizar, conteo, objetos similares, Python.

Automated Object Counting System in Digital Images

Abstract.- In inventories, the context of object counting and speed of response becomes essential. The process is more straightforward when there are only a few items, but its complexity lies when there are many close or overlapping items. In addition, the pressure to meet tight inventory deadlines adds another challenge. This work aimed to develop a program that automates the accurate counting of similar objects in digital images, regardless of their quantity or arrangement. For this purpose, an algorithm has been implemented in Python. The main results show efficiency in analyzing images with similar objects, a significant step towards separating and accurately counting adjacent objects in various scientific fields. This solution promises to simplify and speed up the counting process in digital images, with potential beneficial applications in multiple scientific disciplines.

Keywords: Automate, counting, similar objects, Python.

I. INTRODUCCIÓN

Desde tiempos inmemoriales, los seres humanos hemos sentido la innata necesidad de realizar conteos, abarcando desde objetos como alimentos hasta la misma población, todo con el propósito de obtener datos que puedan influir en la toma de decisiones futuras. En este contexto, hemos persistido en la búsqueda de métodos para cuantificar y analizar variables claves, lo que nos lleva a reflexionar sobre la constante presencia de esta necesidad en nuestras vidas. Con la evolución tecnológica en constante avance, se presenta la emocionante oportunidad de canalizar estos progresos para desarrollar soluciones que simplifiquen y agilicen la tarea de contar objetos. Si bien la aplicación manual puede ser adecuada para recuentos pequeños, la complejidad surge cuando enfrentamos conjuntos considerables o repetitivos. Es en estos escenarios donde la optimización del proceso de conteo a través de la automatización se convierte en una oportunidad de mejora.

Trabajos previos [1] han investigado sobre un sistema automatizado en línea para la clasificación y conteo de la madurez de los tomates. El proceso tradicional de clasificación y conteo de la madurez de los tomates se realiza principalmente de manera manual, lo cual es laborioso y consume mucho tiempo. La precisión de este método depende de la observación precisa del ojo humano. Para abordar este problema, los autores han empleado la inteligencia artificial y visión por computadora para mejorar la eficiencia y precisión del proceso. El proceso implica el uso de una cámara digital para obtener conjuntos de datos de imágenes de tomates, teniendo en cuenta factores como la oclusión e interferencia de la luz externa. Luego, se utiliza un mecanismo de atención MHSA (Multi-Head Self-Attention) junto con el modelo YOLOv8 para mejorar la capacidad de extracción de características del sistema. Los resultados muestran mejoras en la precisión, recall, F1-score y mAP50 del modelo de clasificación de madurez de tomates construido con esta metodología. Además, gracias al rendimiento excepcional del modelo MHSA-YOLOv8, se construyen modelos de conteo con altos niveles de precisión y recall. Estos modelos no solo son adecuados para la detección en línea, sino también para la detección fuera de línea, lo que mejora significativamente la eficiencia en la cosecha y clasificación de los tomates. Las principales innovaciones de este estudio incluyen la construcción de un conjunto de datos de clasificación y conteo de madurez de tomates basado en situaciones de producción reales, la propuesta de un nuevo método de detección de objetos (MHSA-YOLOv8) y la aplicabilidad tanto en entornos en línea como fuera de línea.

Otros desarrollos [2] se enfocaron en encontrar el mecanismo de detección de objetos más eficaz para la identificación automatizada de glomérulos. El experimento implica variaciones en el desarrollo de modelos, incluyendo el uso de Faster R-CNN, considerando tanto glomérulos individuales como parches de imágenes con múltiples glomérulos. Para evaluar el rendimiento de los modelos, se emplean métricas como el Índice de Intersección sobre Unión (IoU) y la Puntuación Media de Precisión (mAP). Los resultados revelan que el modelo Faster R-CNN logra resultados prometedores, con un IoU promedio del 64,2% y un mAP del 65,7% al trabajar con parches de imágenes de biopsias renales. Finalmente, este estudio aporta a la mejora de la detección automatizada de glomérulos, destacando la eficacia del modelo Faster R-CNN en el contexto de biopsias renales. Se reconoce la importancia de estas contribuciones, al tiempo que se identifican posibles limitaciones o áreas de mejora en la metodología empleada.

Además, otros autores [3] han investigado sobre cómo la Inteligencia Artificial (IA) ha revolucionado diversas industrias, centrándose específicamente en el impacto en la industria alimentaria. Destacan una tarea crítica en esta industria, sobre el conteo de productos durante procesos como el procesamiento, el empaquetado y el transporte. Se argumenta que el conteo manual es tedioso, consume tiempo y está sujeto a errores, lo que puede resultar en pérdidas significativas. En respuesta a estos desafíos, se ha desarrollado un sistema de

conteo visual basado en IA para la industria alimentaria con el objetivo de automatizar el proceso de conteo, reducir errores y mejorar la eficiencia. El sistema propuesto utiliza algoritmos de aprendizaje profundo (deep learning) para analizar imágenes digitales de productos alimentarios y proporcionar conteos precisos. Se menciona que la efectividad del sistema se evaluó a través de varios experimentos, y los resultados indican que puede mejorar significativamente la precisión y eficiencia del conteo visual en la industria alimentaria. En resumen, el texto aborda el papel transformador de la IA en la automatización del conteo de productos en la industria alimentaria y destaca el potencial impacto positivo de este sistema en la eficiencia operativa.

Otras investigaciones [4] abordan el tema de la Inteligencia Artificial (IA) y su aplicación en la acuicultura, específicamente en la producción de peces y el conteo de peces durante el proceso de desove. Se destaca que la IA es ampliamente aplicada para aprender problemas y características a partir de datos proporcionados, procesando la información de manera similar al cerebro humano. Cuando un programa de computadora imita una característica del cerebro humano, se considera "innovador". Entre los métodos utilizados se encuentran los métodos estadísticos, los métodos de inteligencia artificial y métodos tradicionales para verificar la validez. La expansión de la IA está vinculada a un almacenamiento prácticamente infinito y a la abundancia de datos, que incluyen intercambios de información, datos geoespaciales, archivos de video, fotos, mensajes de texto y archivos de audio. Los autores mencionan que el aprendizaje automático se divide en aprendizaje profundo, y este último se divide principalmente en numerosas capas de redes neuronales, lo que le otorga la capacidad de aprender gran cantidad de información y replicar la función cerebral. Aumentar la eficiencia mediante la adición de más capas puede ser beneficioso.

Este trabajo pretende aportar a la consecución de conteos automáticos, capitalizando el acceso común a computadoras personales y herramientas de programación, como el lenguaje Python. Se plantea la creación de un programa diseñado para facilitar precisamente la tarea de conteo de objetos presentes en imágenes digitales. En consonancia con la dinámica actual, donde el mundo se transforma continuamente mediante innovaciones tecnológicas, esta investigación aspira a ser un eslabón en esa cadena de avances, maximizando el potencial intrínseco del lenguaje de programación mencionado. El presente documento se estructura de la siguiente manera: se inicia con una contextualización sobre la importancia del conteo de objetos, se detallan el diseño y los pasos de ejecución del programa propuesto, se exponen los resultados obtenidos, se derivan conclusiones relevantes y se comparten recomendaciones fundamentadas en el desarrollo del estudio.

II. ANTECEDENTES

Trabajos previos [5] han desarrollado la automatización del conteo de folículos ováricos en estudios reproductivos multigeneracionales realizados en ratas, de acuerdo con las directrices 443 y 416 de la Organización para la Cooperación y el Desarrollo Económicos (OCDE). La evaluación manual del conteo diferencial de folículos ováricos es una tarea tediosa y que consume mucho tiempo, que requiere personal altamente capacitado. En este contexto, se probaron aplicaciones de redes neuronales profundas (deep neural networks) para facilitar y mejorar el proceso. Los resultados de la aplicación de aprendizaje profundo proporcionan imágenes superpuestas para una documentación más detallada, junto con una mayor reproducibilidad en los recuentos. Para facilitar la validación planificada de buenas prácticas de laboratorio (GLP), se estableció un flujo de trabajo utilizando MLFlow para realizar todas las etapas, desde la generación de escaneos, entrenamiento de la red neuronal, carga de imágenes del estudio a la red neuronal, generación

y almacenamiento de resultados de manera controlada y reproducible. Se utilizó PyTorch como el marco principal para construir la red neuronal convolucional basada en regiones más rápida (Faster R-CNN) para el entrenamiento. El estudio compara el rendimiento de diferentes profundidades de modelos ResNet con un enfoque específico en la sensibilidad, especificidad y precisión de los modelos. El texto describe detalladamente todos los pasos, desde la etiqueta de datos, el entrenamiento de redes hasta las métricas de rendimiento elegidas para evaluar diferentes arquitecturas de red. También se proporcionan recomendaciones sobre los pasos a tener en cuenta cuando se apunta a la validación GLP.

Otros autores [6] han considerado la utilización de la fotografía digital para contar aves, presentando diferentes métodos adecuados tanto para situaciones fuera de línea como en tiempo real en línea. La investigación realiza un análisis del rendimiento de varios métodos utilizados en censos de aves, con el objetivo de superar limitaciones presentes en las técnicas tradicionales de conteo de aves mediante fotografía digital. La investigación resultó necesaria para abordar las limitaciones de las técnicas de conteo de aves que emplean fotografía digital. Se enfoca en estudiar las técnicas existentes para el conteo de objetos en fotografías digitales y propone métodos para superar una o más limitaciones enfrentadas en las técnicas tradicionales. El conteo de objetos es fundamental en diversas áreas de la ciencia y la tecnología, y la eficiencia del conteo manual disminuye a medida que aumenta el número de objetos. La utilización de fotografía digital para contar aves se presenta como un método atractivo, simple y menos costoso en comparación con las técnicas manuales. El conteo manual, aunque es el método básico, se considera ineficiente debido a errores humanos y no es adecuado para grandes bandadas de aves, ya que los individuos pueden omitir algunas aves o volver a contar la misma ave.

En otros trabajos [7] han utilizado las herramientas inteligentes de selección de imágenes para el monitoreo de plagas en campos o experimentos de laboratorio con el objetivo de identificar la variación de los niveles de infección y mejorar el desarrollo de programas integrados de manejo de plagas. Los autores mencionan que la identificación y el conteo manual de los especímenes capturados suelen ser actividades que consumen tiempo, requieren conocimientos taxonómicos y dependen de la experiencia de especialistas. En este contexto, se plantea que la automatización de este proceso podría reducir costos, aumentar la precisión y hacer escalable el análisis. Se destaca que las técnicas actuales de visión por computadora e inteligencia artificial pueden identificar objetos de interés en imágenes digitales de manera oportuna y precisa.

III. METODOLOGÍA

El diseño propuesto en esta investigación se fundamenta en el desarrollo de un algoritmo altamente eficiente para llevar a cabo el recuento preciso de objetos similares en imágenes digitales. Este proceso se articula a través de distintos pasos y procesos meticulosamente diseñados, los cuales se aplican a la imagen de interés. La consecución de una contabilización precisa se logra mediante la implementación de las transformaciones y operaciones necesarias en cada una de las etapas, como se ilustra en la Fig. 1.

Fig. 1. Diagrama de descripción del diseño.

El diseño del algoritmo abarca una secuencia cuidadosamente planeada de pasos, destinados a ampliar los detalles presentes en la imagen. Esto se logra mediante la aplicación de transformaciones selectivas que resalten las características relevantes, al tiempo que optimizan el proceso en cada fase, desde el inicio hasta la finalización. Cada etapa del proceso de diseño está concebida para maximizar la extracción de información clave, permitiendo una representación más nítida de los objetos similares en la imagen digital. La secuencia de pasos asegura que la imagen sea sometida a transformaciones y ajustes específicos, facilitando así la consecución de resultados más precisos en el recuento de objetos similares. En resumen, el diseño del algoritmo propuesto en este trabajo se basa en una estrategia secuencial de procesos y pasos, meticulosamente diseñados para optimizar la identificación y el recuento de objetos similares en imágenes digitales. Cada etapa contribuye de manera sinérgica a la mejora de la representación visual y, por ende, a la exactitud del conteo logrado [1].

A. Descripción de librerías del Programa

El programa se ha construido utilizando el entorno de desarrollo integrado (IDE) IDLE de Python 3.7, lo cual facilita la creación y manipulación de código en el lenguaje de programación Python. Se utilizó la librería OpenCV que es una herramienta de código abierto para Python, fundamental en esta implementación. OpenCV proporciona una rica gama de funciones y algoritmos que son cruciales para el procesamiento de imágenes en el programa. Su integración permite la ejecución de tareas esenciales para analizar y contar objetos en las imágenes seleccionadas [2]. A continuación, se describen los pasos que se siguieron para la ejecución del programa:

Paso 1: selección de una imagen.

Para iniciar el proceso, se permite la selección de imágenes provenientes de diversas fuentes, ya sea de internet o proporcionadas por el usuario. Estas imágenes deben encontrarse almacenadas en una carpeta accesible desde la PC donde se ejecuta el programa. Los formatos de imagen aceptables incluyen JPEG, JPG y PNG, permitiendo una amplia flexibilidad en la elección de las imágenes. Ver ejemplo de una imagen seleccionable en la Fig. 2.

Fig 2. Imagen seleccionable, con distintas figuras.

Paso 2: imagen convertida a escala de grises.

Una vez obtenidas las imágenes a evaluar, se lleva a cabo su conversión a escala de grises. Este proceso implica calcular el promedio de intensidad de los componentes de color (rojo, verde y azul) en cada pixel. La transformación resultante crea una matriz de intensidad donde los valores varían entre 0 y 255, representando niveles de negro y blanco absolutos, respectivamente. La conversión a escala de grises facilita la detección de objetos al resaltar los contrastes entre luz y sombra [4] (Fig. 3).

Fig. 3. Imagen convertida de RGB a escala de grises.

Paso 3: aplicación de doble filtro gaussiano a una imagen.

Con la imagen en escala de grises, se aplica un filtro gaussiano para suavizar la imagen y reducir las desviaciones extremas entre píxeles. Este proceso elimina ruido y uniformiza las diferencias entre grupos de píxeles, permitiendo así una representación más homogénea. Es importante destacar que se aplican dos iteraciones del filtro gaussiano para maximizar la reducción de ruido y garantizar un panorama más nítido (Fig. 4).

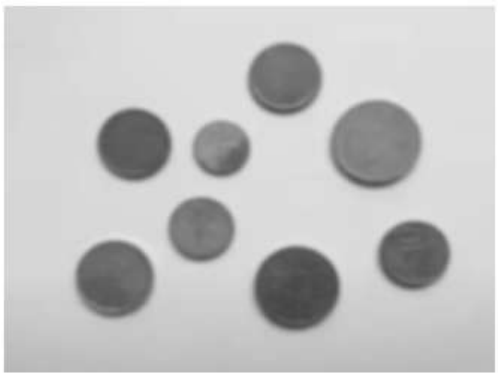

Fig. 4. Imagen aplicando de doble filtro gaussiano

Paso 4: aplicación de algoritmo de Canny a una imagen.

Las imágenes filtradas son sometidas al algoritmo clásico de Canny, reconocido por su capacidad para detectar bordes. Se establecen umbrales alto y bajo, determinando qué bordes serán resaltados en la imagen [5]. La aplicación del algoritmo Canny es crucial para la detección precisa de los objetos y sus contornos en la imagen (Fig. 5).

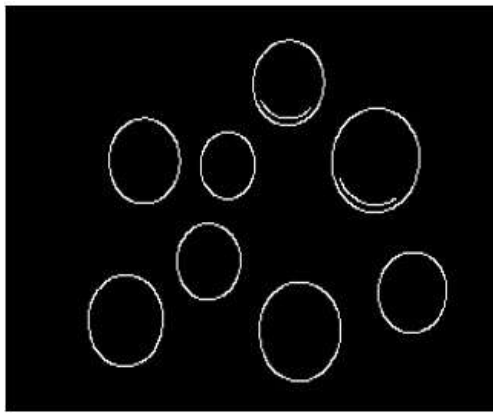

Fig. 5. Imagen aplicando el algoritmo de Canny.

Paso 5: dilatación de la imagen.

Para resaltar y expandir los objetos detectados, se realiza una operación de dilatación. Este proceso involucra el traslado de un elemento estructural a lo largo de la imagen, detectando solapamientos con píxeles de valor 1. La dilatación contribuye a destacar y mejorar la visibilidad de los objetos en la imagen (Fig. 6).

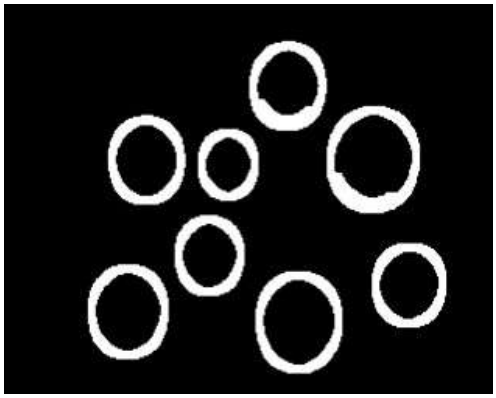

Fig. 6. Imagen aplicando el algoritmo de Canny.

Paso 6: erosión de la imagen.

La operación de erosión se emplea después de la dilatación, con el propósito de refinar la forma de los objetos detectados. Al aplicar un elemento estructural, se verifica si los píxeles de valor 1 están completamente contenidos dentro de la zona de la imagen. La erosión permite una representación más precisa y detallada de los objetos (Fig. 7).

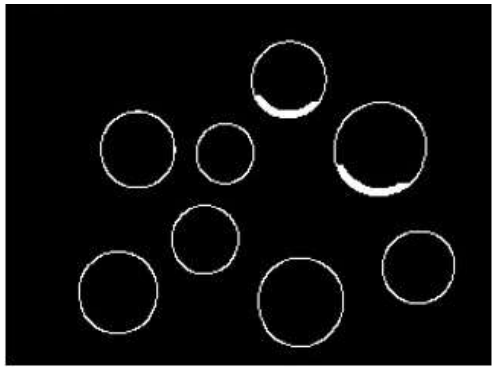

Fig. 7. Imagen cuando es aplicada la erosión.

Paso 7: denotación del borde de la imagen.

En la última etapa, se marcan los bordes de los objetos detectados. Cada objeto es resaltado individualmente, preparando la imagen para el conteo y ofreciendo una visualización clara de los objetos identificados. Esta fase resulta crucial para obtener una cifra verificada de los objetos presentes en la imagen. En síntesis, el diseño del programa abarca una secuencia de pasos estratégicos, desde la selección inicial hasta la denotación de bordes, todos ellos ejecutados de manera coherente para lograr una identificación precisa de objetos en imágenes digitales. Estos pasos se integran de manera sinérgica a través de la biblioteca OpenCV, permitiendo una implementación efectiva del programa diseñado (Fig. 8).

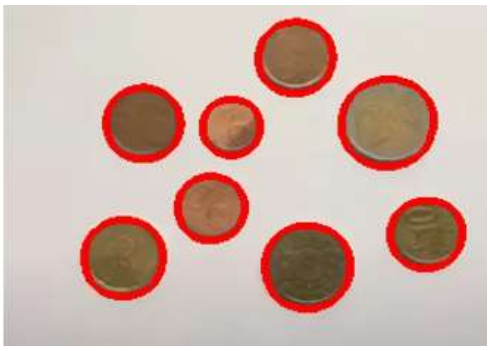

Fig. 8. Imagen con los bordes de los objetos denotados.

IV. RESULTADOS

A. Desarrollo de la aplicación de escritorio

Se ha logrado desarrollar con éxito una aplicación de escritorio utilizando el entorno de desarrollo integrado (IDE) de Python. Esta herramienta permite a los usuarios determinar la cantidad de objetos similares presentes en una imagen seleccionada en su computadora, siempre que la imagen cumpla con los requisitos esenciales para un conteo eficiente de objetos. La programación integral del proyecto se ha realizado en el lenguaje de programación Python, aprovechando su flexibilidad y respaldada por bibliotecas especializadas en el manejo de datos, procesamiento de imágenes e interfaces. La versatilidad de Python ha demostrado ser fundamental para llevar a cabo este tipo de propuestas de manera exitosa.

Interfaz hombre máquina

La interfaz diseñada garantiza una experiencia de usuario intuitiva. La interacción con el algoritmo se ha concebido de manera sencilla, permitiendo a los usuarios buscar, seleccionar y cargar una imagen con facilidad (Fig. 9).

Ventana principal

La ventana principal (Fig. 9) exhibe una interfaz clara y funcional. Los botones "Seleccionar Imagen" facilitan la búsqueda y elección de una imagen para el análisis de conteo de objetos. Asimismo, el botón "Salir" proporciona una manera rápida y eficiente de cerrar el programa.

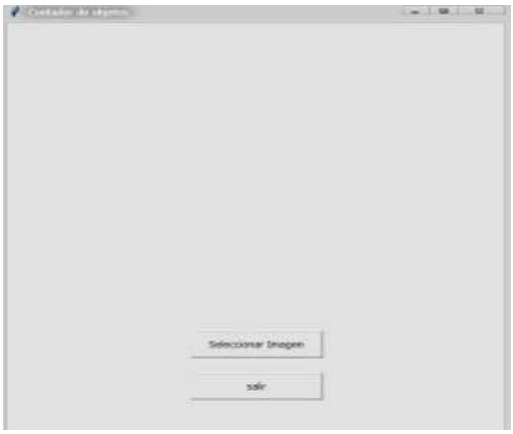

Fig. 9. Ventana al momento de abrir el programa.

La creación de la ventana principal involucra la asignación de un nombre, dimensiones específicas y la restricción de cambios en su tamaño durante el uso. La adición de los botones de selección y salida se muestra en la Fig. 10.

```
ventana=tk.Tk()
ventana.title("Contador de objetos")
ventana.geometry('488x588')
ventana.resizable(False,False)
```

Fig. 10. Ventana principal (Código la creación de la ventana principal).

En la Fig. 10, se muestra parte del código y algunos de sus atributos. La simbología se describe como:

- Tk.tk(): el comando para crear la ventana de la interfaz con nombre de variable “ventana” para identificar y poder asignar sus demás atributos.
- ventana.title: con ese comando le damos el nombre a la ventana.
- ventana.geometry: nos permite darle el tamaño deseado.
- ventana.resizable: es para que no pueda cambiar el tamaña de la ventana.

En la Fig. 11, tenemos el código con el cual creamos los botones que se muestran en la ventana principal.

- boton: contiene los parámetros del botón de seleccionar el archivo de la imagen.
- boton1: contiene los parámetros del botón salir.

```
boton=
tk.Button(ventana,command=seleccio
n_imagen, text = "Seleccionar
Imagen", width ="20", height ="2")
      boton.place(x=180, y = 440)

boton1=
tk.Button(ventana,command=mensaje
, text= "salir",width ="20", height
="2")
```

Fig. 11. Código la creación de los botones.

Ensayo de aplicación:

Una vez que la imagen deseada ha sido seleccionada desde la computadora, esta se presenta de manera visible en la ventana principal, acompañada de la cantidad de objetos identificados (Fig. 12). La imagen seleccionada es procesada internamente, lo que lleva a la ventana principal a mostrar el número de objetos detectados.

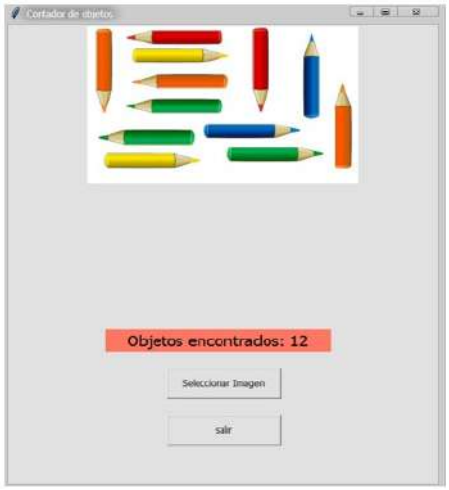

Fig. 12. Ventana al momento de ejecutar el programa.

El proceso de selección y procesamiento de la imagen está respaldado por líneas de código específicas (Fig. 13 y Fig. 14), que aseguran la visualización precisa de la imagen y la presentación del recuento de objetos en la ventana principal de la aplicación.

```
LblInputImage=Label(ventana)
LblInputImage.place(x=50, y=30)
```

Fig. 13. Código para mostrar la imagen seleccionada.

```
def seleccion_imagen ():
   #fomatos permitos para la seleccion de imagenes
    path_image= filedialog.askopenfilename(filetypes=[
        ('Imagen','*.png'),
        ('Imagen','*.jpg'),
        ('Imagen','*.jpeg')])
    if len(path_image) > 0:
        global image
        #leemos la imagen de entrada
        image=cv2.imread(path_image)
        imageToShow=imutils.resize(image,height=256,width=308)
        imageToShow= cv2.cvtColor(imageToShow, cv2.COLOR_BGR2RGB)
        im=Image.fromarray(imageToShow)
        img=ImageTk.PhotoImage(image=im)
        LblInputImage.configure(image=img)
        LblInputImage.image=img
        LblInputImage.pack()
        kernel=np.ones((5,5), np.uint8)
        #pasamos la imagen a escala de grises
        gray=cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
        #aplicamos un doble filtro gaussiano
        gauss2=cv2.GaussianBlur(gray,(5,5),1)
        gauss=cv2.GaussianBlur(gauss2,(5,5),1)
        #aplicacion del algoritmo de canny
        Canny=cv2.Canny(gauss,100,250)
        dilation=cv2.dilate(Canny,kernel, iterations= 1)
        erosion = cv2.erode(dilation,kernel, iterations= 1)
        #contornos
        ctns,_=cv2.findContours(erosion, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
        #denotamos los bordes
        bordes=cv2.drawContours(image, ctns,-1,(0,0,255),3)
        #cantidad de objetos contados
        main_title =Label(text="Objetos encontrados: " +str(len(ctns)),
        font = ("Tahoma",14), bg="#ff7763", fg="black", width ="25")
        main_title.place(x=110, y=390)
```

Fig. 14. Función para seleccionar y tratar la imagen.

En resumen, se ha logrado la creación exitosa de una aplicación de escritorio que permite a los usuarios identificar y contar objetos similares en imágenes seleccionadas. La interfaz amigable y el proceso de conteo eficiente son aspectos destacados de esta implementación, respaldados por una programación en Python y el uso de bibliotecas especializadas.

CONCLUSIONES

En este estudio, se han emprendido investigaciones esenciales para el desarrollo exitoso de una aplicación de escritorio en el Entorno de Desarrollo Integrado IDLE de Python. Se llevó a cabo un análisis de las diversas librerías requeridas, así como de los comandos y funciones específicos que permitirían la consecución de los objetivos planteados. Además, se logró adquirir un dominio fundamental del lenguaje de programación Python, el cual demostró ser una herramienta altamente valiosa, especialmente en el ámbito del procesamiento de información y, en este caso, en el procesamiento de imágenes.

En relación con la ejecución del conteo de objetos, se observó que, si bien la velocidad puede variar en función del número de objetos presentes en la imagen, las diferencias en el tiempo de procesamiento son mínimas y se miden en cuestión de segundos. Es importante considerar también el rendimiento de la PC en la que se ejecuta la aplicación, ya que esto puede influir en los tiempos de respuesta. Uno de los logros sobresalientes de este trabajo es la capacidad de contabilizar objetos que no poseen una coincidencia estricta en términos de color, textura y forma. Este resultado abre la puerta a futuras investigaciones, explorando cómo ampliar aún más la versatilidad del algoritmo para detectar una gama más diversa de objetos.

La elección de emplear Python en lugar de Matlab para esta investigación se fundamentó en su entorno de desarrollo más amigable, así como en su capacidad para optimizar el rendimiento y los recursos del sistema. La decisión de utilizar Python se vio respaldada por su enfoque en la accesibilidad y la facilidad de uso, lo que contribuyó significativamente a la comprensión y el manejo exitoso del lenguaje. Finalmente, este estudio no solo ha culminado con el desarrollo de una aplicación eficiente para el conteo de objetos, sino que también ha enriquecido el conocimiento en programación y ha demostrado el potencial de Python como una herramienta robusta en el procesamiento de imágenes y la solución de problemas complejos [3].

A fin de garantizar una ejecución efectiva del conteo automático de objetos, es esencial que las imágenes sometidas al procesamiento cumplan con requisitos específicos que favorezcan la precisión del conteo. Se recomienda seleccionar imágenes en función de su utilidad prevista, considerando que la eficiencia de la cuenta automática depende en gran medida de la elección adecuada de las imágenes.

El programa desarrollado presenta un amplio espectro de aplicaciones, abarcando campos como la industria, medicina, biología y otros. Sin embargo, es crucial resaltar que su adaptación y modificación resulta imperativa para cada contexto particular. Cada implementación deberá ser ajustada y personalizada en función de los requisitos específicos de cada área.

Para mejorar la efectividad del programa, se plantean las siguientes recomendaciones:

1. Integrar Aplicaciones: se sugiere la integración de funcionalidades adicionales en la aplicación, con el propósito de ampliar su utilidad y brindar a los usuarios una experiencia más versátil y completa.
2. Mantenerse Actualizado en Python: dado que el lenguaje de programación Python sigue evolucionando con la incorporación de herramientas, librerías y mejoras en el entorno IDLE, es esencial mantenerse al tanto de las actualizaciones y adquirir conocimientos en las nuevas funcionalidades para maximizar el potencial del programa.

3. Contabilización de Objetos Variados: se recomienda avanzar hacia la capacidad de contar objetos con formas, colores y texturas distintas, agrupándolos en base a aproximaciones significativas. Esta mejora permitirá una identificación más precisa y versátil de objetos en la imagen.

4. Desarrollo de Aplicación Móvil: explorar la posibilidad de desarrollar una aplicación móvil (APP) que permita contar objetos similares a través de imágenes capturadas por la cámara del dispositivo. Esta expansión facilitaría la utilización del programa en diversas situaciones y ubicaciones.

REFERENCIAS

[1] P. Li, J. Zheng, P. Li, H. Long, M. Li, and L. Gao, "Tomato Maturity Detection and Counting Model Based on MHSA-YOLOv8," Sensors, vol. 23, no. 15, 2023, doi: 10.3390/s23156701.

[2] P. Jasitha and P. N. Pournami, "Glomeruli Detection Using Faster R-CNN and CenterNet," in 2023 3rd Asian Conference on Innovation in Technology, ASIANCON 2023, 2023. doi: 10.1109/ASIANCON58793.2023.10270511.

[3] A. Aharari, K. Kuwaduru, and F. Mehdipour, "Development of an Artificial Intelligence (AI) Based Visual Counting System for the Food Industry," in 13th IEEE Symposium on Computer Applications and Industrial Electronics, ISCAIE 2023, 2023, pp. 136–139. doi: 10.1109/ISCAIE57739.2023.10165399.

[4] S. K. Aruna, N. Deepa, and T. Devi, "Underwater Fish Identification in Real-Time using Convolutional Neural Network," in Proceedings of the 7th International Conference on Intelligent Computing and Control Systems, ICICCS 2023, 2023, pp. 586–591. doi: 10.1109/ICICCS56967.2023.10142531.

[5] E. Carboni et al., "A Workflow for the Performance of the Differential Ovarian Follicle Count Using Deep Neuronal Networks," Toxicol Pathol, vol. 49, no. 4, pp. 843–850, 2021, doi: 10.1177/0192623320969130.

[6] K. M. Spoorthy, S. G. Hegde, N. Vijetha, M. S. Rudramurthy, T. G. Keerthan Kumar, and S. A. Sushma, "Performance analysis of bird counting techniques using digital photograph," in Proceedings - 5th International Conference on Intelligent Computing and Control Systems, ICICCS 2021, 2021, pp. 1482–1491. doi: 10.1109/ICICCS51141.2021.9432265.

[7] T. De Cesaro Júnior and R. Rieder, "Automatic identification of insects from digital images: A survey," Comput Electron Agric, vol. 178, 2020, doi: 10.1016/j.compag.2020.105784.