CARRERA DE ESPECIALIZACIÓN EN INTELIGENCIA ARTIFICIAL

MEMORIA DEL TRABAJO FINAL

Reconocimiento de intrusos en video de vigilancia aérea

Autor:
Ing. Juan Pablo Nieto Uribe

Director:
Esp. Ing. Hernán Contigiani (FIUBA)

Jurados:
Nombre del jurado 1 (pertenencia)
Nombre del jurado 2 (pertenencia)
Nombre del jurado 3 (pertenencia)

Este trabajo fue realizado en la ciudad de Bogotá, Colombia, entre octubre de 2021 y junio de 2020.

Resumen

La presente memoria detalla el proceso de creación de un módulo de inteligencia artificial diseñado para detectar personas y animales en vídeo de vigilancia capturado con cámaras montadas en drones. En particular, el objetivo de este trabajo es detectar intrusos en streaming de vídeo de seguridad. El módulo se desarrolló en colaboración con Manta Beach, empresa que hace parte del programa de vinculación de la universidad y que propuso el proyecto como parte de su plan de investigación y desarrollo.

El desarrollo de este trabajo hace un uso extensivo de los conocimientos de la rama de visión por computadora a través de redes neuronales convolucionales. De igual manera, se ha requerido una base sólida en Python, teniendo en cuenta que se partió de un algoritmo establecido con anterioridad.

Agradecimientos

Índice general

Índice de figuras

Índice de tablas

A Iván. Por tu intuición.

Capítulo 1

Introducción general

1.1. Introducción

El uso de cámaras es una práctica extremadamente común en el negocio de la seguridad. Su uso puede mejorar el área de cobertura de vigilancia, mientras que al mismo tiempo reduce la cantidad de personal empleado en hacer rondas de seguridad. Sin embargo, tener cámaras estáticas implica que será necesario contar con más de ellas, que a la vez tendrán que ser revisadas por varias personas.

La propuesta abordada en este trabajo busca solucionar estos dos problemas a través de la creación de un módulo de inteligencia artificial que permita el análisis automático de varios streamings de vídeo provenientes de cámaras montadas en drones, y sea capaz de responder a estos estímulos digitales, al pasarle la información del intruso encontrado al software del que hará parte.

Esto permitirá que haya un análisis mucho más preciso y a menor costo de lo que era posible anteriormente, teniendo en cuenta que los drones podrán abarcar un área mucho mayor, y que no se dependerá de la concentración de una persona para encontrar intrusos. Esto implica que el sistema deberá estar en capacidad de analizar más de un streaming de video al tiempo. Igualmente, el uso de estas tecnologías permite la utilización de cámaras infrarrojas, lo que hace posible la identificación de intrusos en la noche, cuando el ojo humano es incapaz de distinguir.

Dado que este sistema hará parte de un módulo más grande, se decidió que, dentro de los requisitos, el sistema no deberá tomar decisiones basadas en sus hallazgos, sino simplemente comunicarlos tanto a otro sistema (del que hace parte), como al personal de seguridad encargado.

1.2. Motivación

Este trabajo responde a cinco necesidades concretas del ámbito de la vigilancia. Estas necesidades son:

- Razones económicas: Teniendo en cuenta que, al reducir al personal encargado de hacer seguimiento a las cámaras, o de hacer rondas de seguridad en horarios determinados, es posible reducir costos. De igual manera, el uso de drones permite una reducción sustancial en el número de cámaras que deberán ser instaladas, cosa que resulta particularmente útil en grandes áreas y permitirá una reducción muy importante en los costos de mantenimiento asociados.

- Calidad de la vigilancia: Dado que el trabajo de gran parte del personal de vigilancia tiene la particularidad de que puede volverse muy monótono. Esto teniendo en cuenta que es un trabajo en el que se espera que no suceda nada. Sin embargo, es de vital importancia cuando esto deja de cumplirse, dado que hay algún evento que pueda afectar la seguridad.

  Al ser tan monótono, es muy probable que el personal encargado se vea expuesto a distracciones, que pueden tener consecuencias severas en el momento en el que se presente algún evento que comprometa la seguridad. La situación se puede agravar en caso de tener una gran cantidad de cámaras, dado que el estar constantemente obligados a supervisarlas puede generar niveles no razonables de carga en el personal.

- Velocidad de reacción: Una vez identificada una potencial amenaza, a través del sistema encargado de controlar los drones se podrá enviar ordenes a los equipos a fin de tomar mejores decisiones para manejar la situación. Es por esto que es de vital importancia que el módulo de inteligencia artificial encargado de la detección de los intrusos también sea capaz de hacerlo en tiempos razonables, que en este caso, han sido definidos por el cliente y se encuentran por debajo de los 4 segundos. Este valor se ha calculado a partir de la velocidad esperada de vuelo de los drones de hasta 30 km/h, lo que otorga un radio máximo de 33 metros entre detecciones. Cabe anotar, sin embargo, que el tiempo ideal es menor o igual a un segundo, lo que genera un radio de 8 metros aproximadamente.

- Aporte de pruebas: Los drones (a través de la necesidad de velocidad de reacción) son un método ideal para aportar pruebas confiables de los hechos. Es así como un modelo de inteligencia artificial permite generar una mayor claridad en la interpretación de las pruebas aportadas, lo que puede generar una mejor comprensión y análisis de los hechos por parte de todos los implicados.

- Identificación de intrusos en horas de la noche: Teniendo en cuenta que los modelos de inteligencia artficial pueden ser capaces de identificar las clases requeridas con diferentes condiciones de iluminación, puede ayudar a la identificación de intrusos cuando las condiciones de iluminación no son ideales. Por otro lado, es muy probable que esto sea mejorado sustancialmente con el uso de cámaras infrarrojas.

Este tipo de propuestas, sin embargo, no se han hecho posibles hasta hace relativamente poco, cuando los modelos de inteligencia artificial alcanzaron una madurez que permite que no requieran grandes capacidades computacionales para poder ser ejecutados, de forma que implementarlos no requiera una inversión importante en equipos de cómputo.

1.3. Estado del arte

Existen varias técnicas y varios modelos que se puede utilizar para implementar visión por computadora. Uno de los métodos más comunes a la fecha es el algoritmo YOLO, que se caracteriza por ser el primer modelo que no requiere hacer un doble análisis de la imagen (de aquí proviene su nombre: You only look once), y en consecuencia, analiza mucho más rápido.

Modelo	Fecha de Lanzamiento
YoloV3	Abril de 2018
YoloV3 Lite	Abril de 2020
YoloV6	Junio de 2022

A pesar de que, a la fecha de redacción de este documento, las versiones más avanzadas de los modelos de la familia YOLO son las versiones V6 y V7, que ofrecen grandes mejoras en términos de precisión y rendimiento en comparación con sus antecesores. Sin embargo, como se puede ver en la tabla 1, estos modelos no estuvieron disponibles hasta una fecha posterior al inicio del trabajo correspondiente al proyecto.

Por otro lado, las versiones 4 y 5 fueron descartadas para implementación de este proyecto, visto que son dos versiones que fueron desarrolladas por personas diferentes a los desarrolladores originales, quienes se retiraron del proyecto por conflictos éticos con las capacidades de la técnica y sus potenciales usos. En consecuencia, gran parte de la industria se hace uso de la versión 3 del modelo.

Ahora bien, es importante tener en cuenta que existen diferentes versiones del modelo, cada una adaptada a algún caso de uso particular. Destacan dentro de estas versiones la arquitectura YOLOV3 y la arquitectura YOLOV3 Lite:

- YOLOV3: Es la versión "base" de la arquitectura YOLO. Representa un avance incremental con respecto a las arquitecturas YOLOV1 y YOLOV2. Su principal ventaja por encima de los modelos anteriores (en particular comparado con RetinaNet-50 y RetinaNet-101) es una reducción significativa en los tiempos de detección. El dataset sobre el que fue entrenado es el dataset COCO.
- YOLOV3 Lite: Es una versión reducida de la versión YOLOV3 diseñada específicamente para ser ejecutada en dispositivos con menor potencia, o que no cuentan con tarjeta gráfica. Si bien es un modelo que es capaz de reducir sustancialmente la complejidad del equipo requerido, también cuenta con una menor precisión en el momento de hacer la detección correspondiente.

Por otro lado, además de la arquitectura, es de vital importancia tener en cuenta el dataset que fue utilizado para el entrenamiento del modelo. En este caso, se trata del dataset coco, que cuenta con más de 200.000 imágenes etiquetadas (a la fecha de redacción de este documento), con más de 1.5 millones de objetos detectados en ellas.

1.4. Objetivos y alcance

A continuación, se presentan los requisitos definidos durante la etapa de definición de los objetivos del proyecto, así como el estado de cada uno de ellos a la fecha de finalización del trabajo. Es importante tener en cuenta que, dado que el proyecto sufrió cambios por temas operativos, de forma que algunos de los requisitos no pudieron ser cubiertos.

A pesar de que algunos de los requerimientos funcionales fueron descartados dado el estado del proyecto en el extremo del cliente, también hubo otros que fueron agregados:

Tipo de Requerimiento	Número de Requerimiento	Requerimiento
Requerimientos Funcionales	1.1	El módulo debe detectar personas
	1.2	El módulo debe ser capaz de detec
	1.3	El módulo debe correr en un comp
	1.4	El sistema debe ser capaz de recon
	1.5	El sistema debe correr en un siste
Requerimientos de documentación	2.1	El sistema deberá contar con un m
	2.2	Se deberá contar con un manual d
	2.3	El código fuente deberá estar debi
Requerimientos de testing	3.1	El módulo deberá ser testeado for
	3.2	El cliente debe poder acceder en c
Requerimientos de la interfaz	4.1	El usuario debe poder iniciar fácil
Requerimientos de interoperabilidad	5.1 A	El módulo debe estar en capacidad
	5.1 B	El módulo debe ser capaz de recib
	5.2	Se requiere que el sistema pueda c

Tipo de Requerimiento	Número de Requerimiento	Fecha de Lanzamiento
Requerimientos Nuevos	6.1	El módulo deberá retransmitir los frames proce

Es importante tener en cuenta que el trabajo no contempló:

- Reconocimiento del intruso. Es decir, en caso de que se detecte a una persona, no se deberá dar con su identidad.
- Reacción del dron: es decir, una vez se haya detectado satisfactoriamente a un intruso, el sistema no deberá dar instrucciones específicas al dron, por ejemplo, de seguirlo.
- No existen requisitos específicos de cómo realizar la retransmisión de los frames ya procesados.

Capítulo 2

Introducción específica

Todos los capítulos deben comenzar con un breve párrafo introductorio que indique cuál es el contenido que se encontrará al leerlo. La redacción sobre el contenido de la memoria debe hacerse en presente y todo lo referido al proyecto en pasado, siempre de modo impersonal.

2.1. Definición de Intruso

Dado que el trabajo tiene un objetivo específico en mente, la definición de intruso dada por el cliente incluye únicamente personas o animales de gran porte. El dataset de COCO permite el reconocimiento de una serie de animales de gran porte que no hacen parte de la fauna argentina, por lo que se decidió en conjunto con el cliente que algunas etiquetas se filtrarían a fin de evitar que el modelo trate de reconocer estas clases. Algunos ejemplos de etiquetas filtradas incluyen:

- Elefantes
- Jirafas
- Cebras

Por otro lado, y a fin de mejorar el reconocimiento de intrusos, se decidió incorporar algunas clases que corresponden a objetos que suelen cargar las personas y vehículos. De esta manera se hace posible una suerte de reconocimiento indirecto, dándole al modelo una "segunda oportunidad" en casos en los que por algún motivo generó un falso negativo en el reconocimiento de una persona. Es importante tener en cuenta que estos objetos propuestos ya hacían parte del dataset COCO, por lo que no se requirió reentrenar el modelo para lograr su detección. Dentro de estos objetos relacionados encontramos:

- Vehículos
- Objetos de porte personal, como maletines o sombrillas

Finalmente, es importante tener en cuenta que cualquier detección de las clases indicadas dentro del streaming de vídeo será marcada como intruso. Es decir, no será necesario considerar que la presencia de cualquiera de estas clases en un lote vecino no deberá ser marcada como intruso. Dicho esto, las clases que generarán un reporte positivo para intruso al ser detectadas son:

Hay que recordar que la sola detección de una de estas clases generará un reporte positivo de intruso. El modelo no está en capacidad de identificar personas específicas en el vídeo, por lo que le resultará imposible diferenciar entre personal

Clases detectadas como Intruso	
Persona	Vaca
Camión	Bus
Automóvil	Sombrilla
Motocicleta	Mochila
Bicicleta	Maletín
Perro	Maleta
Caballo	Gato

autorizado en un área (el mismo personal de seguridad, o la persona al mando del drone), y a intrusos en la zona.

2.2. Modelo Utilizado

El modelo utilizado para el desarrollo de la totalidad del trabajo es YoloV3. La principal ventaja que tiene este modelo es que es capaz de ejecutarse con una velocidad considerablemente mayor a la de modelos anteriores como RetinaNet-50 y RetinaNet-101. Este modelo funciona a través del uso de redes neuronales convolucionales y calcula tres parámetros:

- Intersection over Union (IoU), que puede ser interpretado como la precisión del modelo al rodear un objeto con una caja. Es decir, qué tan acertada es la posición de la caja propuesta con el modelo, en relación con la posición predicha del objeto anteriormente.
- Offset de la caja detectada en comparación con la caja ancla.
- Tensor con la probabilidad de ocurrencia de las 80 clases que forman parte del dataset COCO.

Es importante tener en cuenta que estos no son los parámetros finales arrojados por el modelo, sino que más adelante, la red calcula las coordenadas de la caja con la que se rodeará al objeto. Es decir, para cada objeto detectado, el modelo va a devolver (en ese orden):

- pc: Probabilidad de que haya un objeto.
- bx: Coordenada x del centro de la caja propuesta
- by: Coordenada y del centro de la caja propuesta
- bh: Altura de la caja propuesta
- bw: Ancho de la caja propuesta
- c: Tensor con la probabilidad de ocurrencia de las 80 clases que forman parte del dataset COCO.

Esto se logra a través del uso de Anchor Boxes: cajas de predicción predefinidas que el modelo va refinando a medida que analiza la imagen y calcula las métricas de cada uno de los objetos encontrados. A través de este método es posible eliminar la necesidad de una "ventana deslizante", lo que a su vez permite analizar la totalidad de la imagen en un tiempo mucho menor que los métodos basados en aquel método, y, en consecuencia, se hace posible llevar a cabo un análisis en tiempo real.

Dadas las bondades explicadas anteriormente, en conjunto con su facilidad de implementación en Python, se escogió partir del modelo YoloV3 y modificar su código fuente para lograr generar y extraer la información pertinente para completar el propósito del proyecto. Cabe aclarar que, la decisión de utilizar este modelo se vio reforzada una vez se ejecutó la primera versión en tanto que el modelo está optimizado para funcionar con tarjeta gráfica (Nvidia Titan X), el código se pudo ejecutar satisfactoriamente, sin dejar a un lado los requisitos del proyecto en CPU (AMD Ryzen 5).

2.3. Protocolos de vídeo y transmisión

El módulo del proyecto encargado de extraer y permitir el cálculo sobre cada uno de los frames es OpenCV, por lo que los protocolos de vídeo aceptados corresponderán a los aceptados por OpenCV. En este sentido, el código está en capacidad de detectar intrusos en:

- Vídeo guardado localmente
- Secuencia de imágenes
- URL de streaming de vídeo
- Pipeline Gstreamer
- Datos obtenidos desde Webcam

Sin embargo, es importante mencionar que, durante el desarrollo, se hizo pruebas únicamente con los métodos de vídeo guardado localmente y URL de streaming de vídeo. Los otros métodos no han sido probados, y, por lo tanto, no se podrá garantizar su correcto funcionamiento al solicitar al módulo ejecutarse utilizándolos como origen de datos.

Por otro lado, y recordando que se trata de la librería que hace posible extraer la información del vídeo para hacer posible su análisis, es de vital importancia tener en mente cuáles son los formatos de vídeo admitidos por OpenCV. Dentro de ellos se encuentran:

- H264
- MPEG4
- AV1

Ahora bien, para acceder a ellos, bastará con pasar el link apropiado al método . Dentro de los protocolos probados durante el desarrollo del proyecto encontraremos:

- MQTT: Es un protocolo que permite la comunicación entre dispositivos desarrollado por IBM con gran énfasis en IoT. Es un protocolo muy seguro teniendo en cuenta con un componente que filtra las comunicaciones entre el servidor y el cliente, por lo que ninguno de los componentes conocerá los datos del otro componente. En cambio, el cliente solicita una suscripción a un tema que le interesa, y el agente se encarga de distribuir los mensajes enviados por el servidor. Cabe mencionar que MQTT está pensado especialmente para dispositivos IoT con recursos limitados y poco ancho de banda.

			Software Requerido
Paquete	Tipo	Licencia	Función
PIP	Gestor de Librerías	MIT	Gestor de librerías
Numpy	Librería de Python	BSD	Cálculos Matricial
OpenCV	Librería de Python	Apache 2	Manipulación de i
Matplotlib	Librería de Python	BSD	Creación de visua
Tensorflow	Librería de Python	Apache 2	Framework de Ma
Pycharm	IDE	Apache 2	Ambiente de desa
OBS Studio	Software completo	GPLv2	Streaming de víde
Python	Lenguaje de programación	Python Software Foundation License V2	Lenguaje en el que
Ubuntu	Sistema operativo	Creative Commons CC-BY-SA v 3.0 UK	Sistema operativo

- • RTMP: Es un protocolo diseñado específicamente para la transmisión de audio y vídeo con gran rendimiento. Esto lo logra dividiendo la información en pequeños fragmentos, cuyo tamaño puede ser negociado de manera dinámica entre el cliente y el servidor de forma que la transmisión de vídeo se adapta mejor a las condiciones de la red de la que depende la transmisión. Sin embargo, dado que se trata de un protocolo diseñado para la transmisión de vídeo y audio con el menor delay posible, este protocolo puede no ser el más adecuado cuando las condiciones de la red son limitadas.

2.4. Software y Hardware utilizados

Para su funcionamiento, este trabajo requiere la presencia de algunos paquetes de Software instalados en la máquina, los cuales son llamados directamente por el usuario cuando requiera la funcionalidad (Por ejemplo, OBS Studio), o bien, que ya vienen incorporados en el código y por lo tanto, son un requisito indispensable para el proyecto y deberán encontrarse instalados. Dicho esto, podemos clasificar el software utilizado en tres categorías:

1. Paquetes de Python: Son paquetes importados en el código a través del comando import. Cabe mencionar que todos los paquetes son de código abierto. Ejemplos de estos paquetes son:

- Numpy
- Matplotlib
- OpenCV

2. Código en Python: En especial para la implementación de la metodología YOLOV3. En internet está disponible el código fuente en formato Python (py).

3. Software completo: En particular para la retransmisión del vídeo procesado, se utilizó OBS estudio. Esta decisión responde a que, dentro del alcance del proyecto, no se definió ninguna actividad que no se encuentre directamente relacionada con la implementación de la detección de intrusos. Sin embargo, dado que se trata de un proyecto en el que se debe devolver la imagen procesada, es importante asegurarse que la retransmisión de los datos funcione de la manera correcta.

Más específicamente, las diferentes librerías y paquetes de software utilizados para el desarrollo de este trabajo, y sus versiones correspondientes son:

Es importante tener en cuenta que la totalidad de programas y librerías utilizadas, tanto para el desarrollo, como para la ejecución del proyecto son Software libre, por lo que no se requerirá adquirir ningún tipo de licencia para ponerlo en funcionamiento.

De igual manera, cabe resaltar que OBS Studio es el programa de retransmisión elegido teniendo en cuenta que esta es una característica que se encuentra fuera del alcance del trabajo y por lo tanto, cuya implementación es necesaria a través de este método mientras que hacerlo a través de software integrado en el código fuente (como FFMpeg) será parte del alcance de trabajos posteriores.

El paquete de Python ya viene instalado por defecto en la instalación de Ubuntu, por lo que no se requerirá su instalación. El resto de los paquetes, sin embargo, deberán ser instalados de acuerdo con el manual de instalación que se entrega al cliente, en el que se detalla de manera minuciosa cuál es el procedimiento que se debe seguir a fin de instalar correctamente todos los paquetes.

De igual manera, se debe tener en cuenta que dadas las restricciones de la máquina en la que se desarrolló el proyecto, la totalidad del desarrollo fue llevado a cabo en máquinas virtuales a través del uso de programas de virtualización como VMWare Workstation. Dentro de los requerimientos mínimos de hardware del proyecto, se tienen los siguientes:

- Procesador: AMD Ryzen 5
- Tarjeta Gráfica: No requerida
- Memoria RAM: 6 GB.

Es importante tener en cuenta que los listados anteriormente son los requisitos mínimos del sistema, que van a garantizar que el proyecto se ejecute cumpliendo con los requerimientos del cliente. Sin embargo, se recomiendan las siguientes características del sistema para un rendimiento óptimo a la hora de ejecutar el módulo.

- Procesador: Intel Core i7
- Tarjeta Gráfica: NVidia Tesla T4 o NVidia Titan X.
- Memoria RAM: 8 GB.

El modelo YoloV3, sobre el que está basado el desarrollo del proyecto utiliza de manera extensiva la API Cuda de Nvidia. Por este motivo, tratar de ejecutar el proyecto en tarjetas gráficas de AMD resultará en el modelo siendo llevado al procesador por Tensorflow. En consecuencia, se afirmará que el trabajo resulta incompatible con estas tarjetas gráficas. Esto fue comprobado con el uso de una tarjeta AMD Radeon Vega 8.

Capítulo 3

Diseño e implementación

3.1. Consideraciones generales

La idea de esta sección es resaltar los problemas encontrados, los criterios utilizados y la justificación de las decisiones que se hayan tomado.

Se puede agregar código o pseudocódigo dentro de un entorno lstlisting con el siguiente código:

```
\begin{lstlisting}[caption= "un epígrafe descriptivo"]
las líneas de código irían aquí...
\end{lstlisting}
```

A modo de ejemplo:

```
#define MAX_SENSOR_NUMBER 3
#define MAX_ALARM_NUMBER  6
#define MAX_ACTUATOR_NUMBER 6

uint32_t sensorValue[MAX_SENSOR_NUMBER];
FunctionalState alarmControl[MAX_ALARM_NUMBER]; //ENABLE or DISABLE
state_t alarmState[MAX_ALARM_NUMBER];           //ON or OFF
state_t actuatorState[MAX_ACTUATOR_NUMBER];     //ON or OFF

void vControl() {

  initGlobalVariables();

  period = 500 ms;

  while(1) {

    ticks = xTaskGetTickCount();

    updateSensors();

    updateAlarms();

    controlActuators();

    vTaskDelayUntil(&ticks, period);
  }
}
```

CÓDIGO 3.1. Pseudocódigo del lazo principal de control.

3.2. Arquitectura del proyecto

3.3. Esquema del módulo

3.4. Esquema de modificaciones

3.5. Ajustes para cumplir con el rendimiento

3.6. Entrega de resultados al resto del software

Capítulo 4

Ensayos y resultados

4.1. Descripción del proceso de pruebas

La idea de esta sección es explicar cómo se hicieron los ensayos, qué resultados se obtuvieron y analizarlos.

4.2. Pruebas en Raspberry Pi

4.3. Caso de Uso

Capítulo 5

Conclusiones

5.1. Resultados Obtenidos

La idea de esta sección es resaltar cuáles son los principales aportes del trabajo realizado y cómo se podría continuar. Debe ser especialmente breve y concisa. Es buena idea usar un listado para enumerar los logros obtenidos.

Algunas preguntas que pueden servir para completar este capítulo:

- ¿Cuál es el grado de cumplimiento de los requerimientos?
- ¿Cuán fielmente se puedo seguir la planificación original (cronograma incluido)?
- ¿Se manifestó algunos de los riesgos identificados en la planificación? ¿Fue efectivo el plan de mitigación? ¿Se debió aplicar alguna otra acción no contemplada previamente?
- Si se debieron hacer modificaciones a lo planificado ¿Cuáles fueron las causas y los efectos?
- ¿Qué técnicas resultaron útiles para el desarrollo del proyecto y cuáles no tanto?

5.2. Tiempos de ejecución

Acá se indica cómo se podría continuar el trabajo más adelante.

5.3. Monitoreo de resultados