Revista Científica EMINENTE
Mayo 2023 Volumen 7, Número 1 85-93

EVALUAR LA CAPACIDAD DE DETECCIÓN DE OBJETOS EMPLEANDO INTELIGENCIA ARTIFICIAL PROBANDO LA TEORÍA DE PIOLA Y JAMES CON EL ENTRENAMIENTO A BASE DE REDES NEURONALES PARA MEJORAR EL ENTRENAMIENTO DE TIRO DE LOS COMBATIENTES

EVALUATE THE OBJECT DETECTION CAPACITY USED BY ARTIFICIAL INTELLIGENCE TESTING THE PIOLA AND JAMES THEORY WITH TRAINING BASED ON NEURAL NETWORKS TO IMPROVE FIGHTERS' SHOOTING TRAINING

Ing. Bryan José Atahuchi Pinto [1] * [$]

Recibido: Abril 5, 2022; Aceptado: Marzo 22, 2023

RESUMEN

En la actualidad el entrenamiento de tiro ya sea de escuela o puntería avanzada se lo realiza en las unidades militares y polígonos autorizados utilizando armamento y munición física.

Se tiene un polígono de tiro virtual en la EMI U.A.S.C. implementado el 2021 por ingenieros investigadores, el cual contiene un polígono de tiro virtual que detecta el color de una silueta negra para realizar la detección del blanco y marcado de puntaje.

El objetivo del presente trabajo de investigación es implementar una técnica de reconocimiento de blancos en movimientos, a través de visión artificial y entrenamiento de objetivos por épocas en una red neuronal.

Para la fase de entrenamiento se tiene una red neural echo en Python, el cual se enmarca cada imagen y generación de puntos de coordenadas X y Y de nuestra muestra.

En la fase de diseño del ambiente se utiliza herramienta de modela 3D y Unity3D, creando el ambiente tropical virtual empleado materiales, patrones de diseño y animaciones del entorno.

Para la comunicación entre la visión artificial y el ambiente 3D se utilizó el servicio de comunicación a través de IP denominado SOKET.IO, creando un servidor de C Sharp scripting y un cliente en Python.

Palabras claves: Soket, C Sharp scripting. , Modelado 3D.

ABSTRACT

Currently, shooting training, whether school or advanced marksmanship, is carried out in military units and authorized ranges using weapons and physical ammunition.

There is a virtual shooting range at the EMI U.A.S.C. implemented in 2021 by research engineers, which contains a virtual shooting range that detects the color of a black silhouette to perform target detection and scoring.

The objective of this research work is to implement a moving target recognition technique, through artificial vision and epoch-based target training in a neural network.

For the training phase, we have a red neural echo in Python, which frames each image and generates X and Y coordinate points of our sample.

In the environment design phase, 3D modeling tools and Unity3D are used, creating the virtual tropical environment using materials, design patterns and environment animations.

For the communication between the artificial vision and the 3D environment, the communication service through IP called SOKET.IO was obtained, creating a C Sharp scripting server and a Python client.

Keywords: Soket, C Sharp scripting. , Models 3D.

Citación: Bryan José Atahuichi Pinto, **EVALUAR LA CAPACIDAD DE DETECCIÓN DE OBJETOS EMPLEANDO INTELIGENCIA ARTIFICIAL PROBANDO LA TEORÍA DE PIOLA Y JAMES CON EL ENTRENAMIENTO A BASE DE REDES NEURONALES PARA MEJORAR EL ENTRENAMIENTO DE TIRO DE LOS COMBATIENTES.** Revista Científica EMINENTE 2023, 7-1: 85-93.

[1] Ingeniero de Sistemas – Unidad de Investigación Ciencia y Tecnología - Unidad Académica Santa Cruz - Escuela Militar de Ingeniería.
* Corresponde al Autor (correo electrónico: atahuichi46@hotmail.com)
[$] Dirección de contacto Investigador primer autor: Zona Sur Barrio Tierras Nuevas Av. /Moscú - Telf.: (+591) 77040993 Santa Cruz – Bolivia.

Revista Científica EMINENTE
Mayo 2023 Volumen 7, Número 1 85-93

INTRODUCCIÓN

La simulación de eventos y sucesos reales, cada vez se está haciendo más realistas, con la evolución de la ciencia y tecnología se tienen nuevos equipos, herramientas para mejorar las habilidades humanas a través de simulaciones de eventos discretos y continuos.

La visión artificial ha ido evolucionando con el tiempo, se tiene mejores cámaras y sensores ópticos para poder visualizar la realidad percibida, dentro de estos la comunicaron con los ordenadores la trasmisión de imagen se los hace a través de paquete de bytes que el ordenador procesa y reconstruye.

Las herramientas para poder detectar objetos identificando un objeto base y los clasificadores pudiendo diferenciar de una misma familia las diferencias, fueron aumentando la robustez y técnicas de interpretación de datos a trabas de un muestreo aleatorio de imágenes.

Esto conlleva que el desarrollo de realidad aumentada en los últimos años sea más accesible en los dispositivos móviles, computadoras personales y tabletas, pudiendo crear aplicaciones con los modelos 3D, detección de objetos, clasificadores, realidad aumentada entre otros.

Este trabajo se divide en dos fases: la obtención, de un matriz entrenada con redes neuronales en cien épocas y muestreo aleatorio para la detección de objetos., la segunda fase que será diseño de la arquitectura de aplicación con patrones de diseño y simulación de un ambiente virtual a través de realidad semi-inmersiva.

OBJETIVO GENERAL

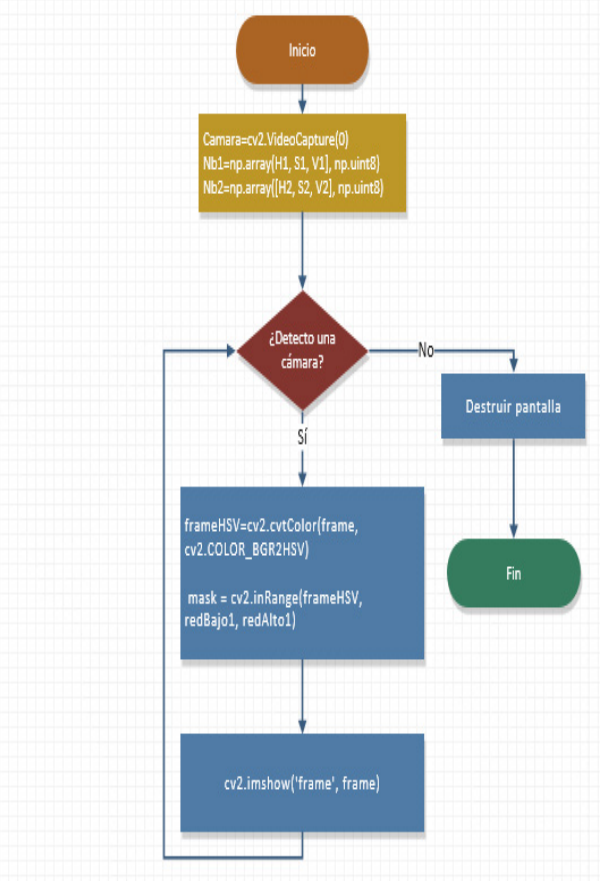
Evaluar la capacidad de detección de objetos empleado inteligencia artificial probando la teoría de piola y james con el entrenamiento a base de redes neuronales para mejorar el entrenamiento de tiro de los combatientes.

METODOLOGÍA

Fase de la obtención, de un matriz entrenada con redes neuronales en cien épocas y muestreo aleatorio para la detección de objetos.

Primero analizamos el identificador de colores integrados en OpenCV el cual maneja un filtro HSV considerado como Tono, Saturación y Brillo de cada color, incorporamos en Python esta herramienta para trabajar con visión artificial, el cual nos indica delimitar filtros altos y bajos del color a detectar la función ocupada es np.array (color, saturación, brillo) y almacenaremos en dos variables diferentes con el fin de detectar el rango mínimo y máximo del color en este caso tenemos que detectar el color Negro de la silueta como se muestra en la siguiente figura.

Figura 1. Detección de flujo de detección de color.



Fuente: Elaboración propia.

Para esto declaramos una cámara y hacemos un bucle, mientras se detecte una cámara, convertimos la detección HSV a formato BGR, establecem0os una máscara con los rangos mínimos y máximos y enviamos el frame detectado, enviamos los frame filtrados a la pantalla y si no se detecta ninguna cámara se destruye la interfaz y finaliza la aplicación el resultado se lo ve en la siguiente figura.

Figura 2. Detección por máscara de escala de grises.

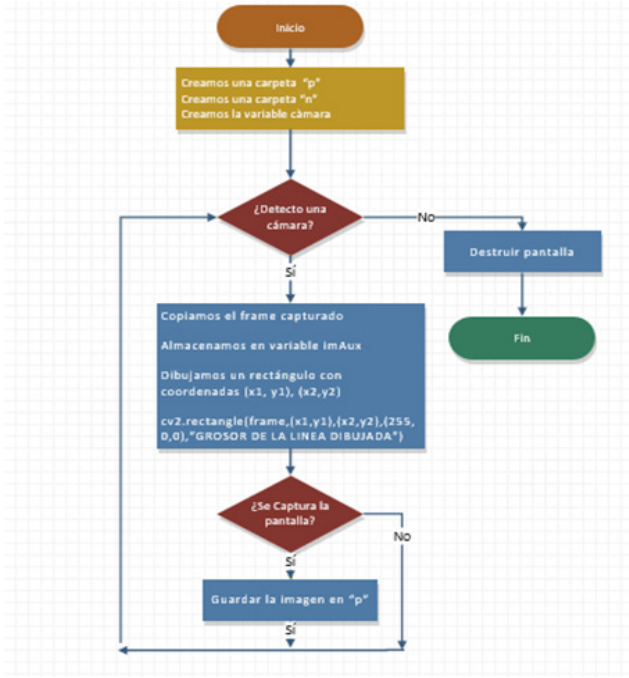
Fuente: Elaboración propia.

Como se puede apreciar en la imagen tenemos identificado en las pruebas 2 problemas uno con el ruido se puede aplicando un filtro de ruido, pero solo aumenta el caso de falsos positivos en la detección la segunda que una variación de luz o enfoque afecta considerablemente a la precisión por lo que se requiere de un algoritmo más robusto y capaz de detectar y clasificar objetos.

Probando la teoría de Piola y James sobre Hard Cascade, es un algoritmo para procesamiento de imágenes y detección de objetos, que se basa en escala de grises elabora como salida una matriz de datos el cual genera una serie de coordenadas de detección, se tomó una muestra de seiscientas fotos de la silueta de entrenamiento de polígono de tiro de tipo D, en el cual se creó un módulo para recolectar muestras diseñado como se ve en la figura 3.

Para este caso tenemos un muestreo de dos tipos imágenes: las positivas, donde se identifican fotografías donde está la silueta en distintos ambientes y negativas donde se toma un muestreo de imágenes donde no aparece el objeto a detectar, se debe tener una gran diferencia entre estos dos tipos, debido a que puede afectar el resultado de detección.

Figura 3. Diagrama de flujo para toma de muestras.



Fuente: Elaboración propia.

Una vez establecido empezamos a correr en el trainner las muestras clasificadas el cual nos da como salida iteraciones de la matriz de resultados de los pesos como se puede apreciar en la figura 4.

Figura 4. Salida de entrenamiento.

```
Parent node: 2

*** 1 cluster ***
POS: 9625 9768 0.985360
NEG: 17785 0.1666
BACKGROUND PROCESSING TIME: 0.63
Precalculation time: 28.50
+----+----+-+---------+---------+---------+---------+
|  N |%SMP|F|  ST.THR |    HR   |    FA   | EXP. ERR|
+----+----+-+---------+---------+---------+---------+
|   1|100%|-|-0.827884| 1.000000| 1.000000| 0.470814|
+----+----+-+---------+---------+---------+---------+
|   2|100%|+|-0.740429| 1.000000| 0.999438| 0.355272|
+----+----+-+---------+---------+---------+---------+
|   3| 99%|-|-0.895278| 1.000000| 0.999438| 0.355272|
+----+----+-+---------+---------+---------+---------+
|   4| 99%|+|-0.979484| 1.000000| 0.999438| 0.355418|
+----+----+-+---------+---------+---------+---------+
|   5| 99%|-|-1.123960| 1.000000| 0.999438| 0.210325|
+----+----+-+---------+---------+---------+---------+
|   6| 99%|+|-1.196007| 1.000000| 0.999438| 0.198030|
+----+----+-+---------+---------+---------+---------+
|   7| 99%|-|-1.238015| 0.995117| 0.770762| 0.192776|
+----+----+-+---------+---------+---------+---------+
|   8| 84%|+|-1.176625| 0.999481| 0.963621| 0.180518|
+----+----+-+---------+---------+---------+---------+
|   9| 94%|-|-1.242356| 0.999481| 0.963452| 0.180482|
+----+----+-+---------+---------+---------+---------+
|  10| 97%|+|-1.134056| 0.996675| 0.853360| 0.167019|
+----+----+-+---------+---------+---------+---------+
|  11| 87%|-|-1.138138| 0.996675| 0.853360| 0.167019|
+----+----+-+---------+---------+---------+---------+
|  12|100%|+|-1.138138| 0.996675| 0.853360| 0.167019|
+----+----+-+---------+---------+---------+---------+
|  13|100%|-|-1.138138| 0.996675| 0.853360| 0.167019|
+----+----+-+---------+---------+---------+---------+
|  14|100%|+|-1.138138| 0.996675| 0.853360| 0.167019|
+----+----+-+---------+---------+---------+---------+
|  15|100%|-|-1.138138| 0.996675| 0.853360| 0.167019|
+----+----+-+---------+---------+---------+---------+
```

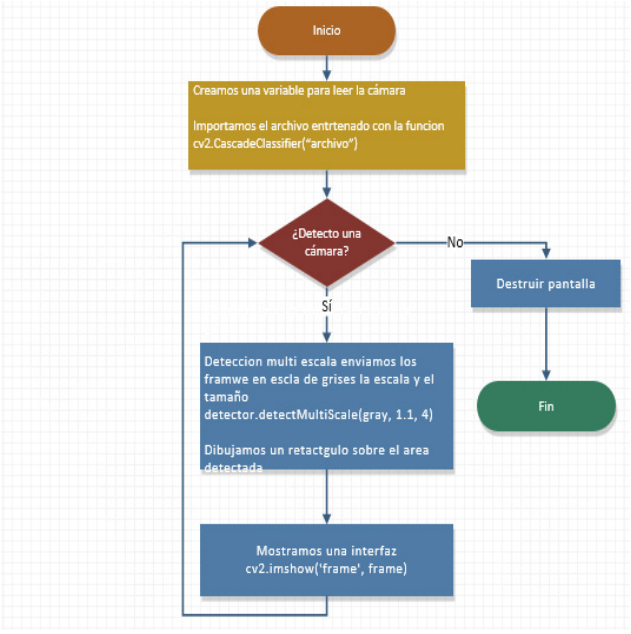
Fuente: Elaboración propia.

ISSN 2409 - 2371

Revista Científica EMINENTE
Mayo 2023 Volumen 7, Número 1 85-93

Una vez llegada a terminar, nos genera un XMl con los pesos procesados que se cargará a nuestro script de detección en Python y realizamos la detección como se ve en la siguiente figura 5.

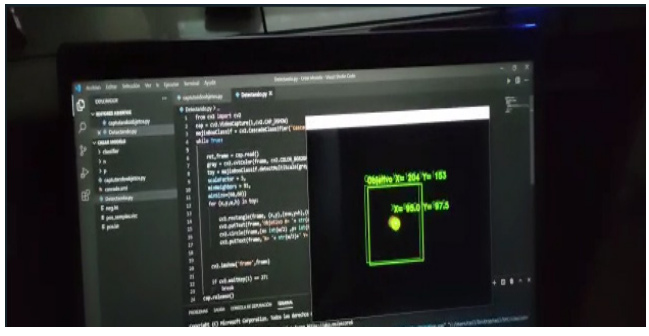
Figura 5. Algoritmo de detección Multi-escala.



Fuente: Elaboración propia.

Como se puede apreciar en la gráfica importamos los pesos, una vez importado pasamos a capturar el entorno a través de la cámara, posterior a esto convertimos todos los frame capturados en escala de grises, luego cargamos los pesos con la función detectMultiScale() y enviamos tres parámetros, los frame en grises, escala y tamaño; con estos datos nuestro entrenador puede empezar a detectar objetos, como se ve en la figura 6 la detección de láser óptico.

Figura 6. Detección de láser óptico.



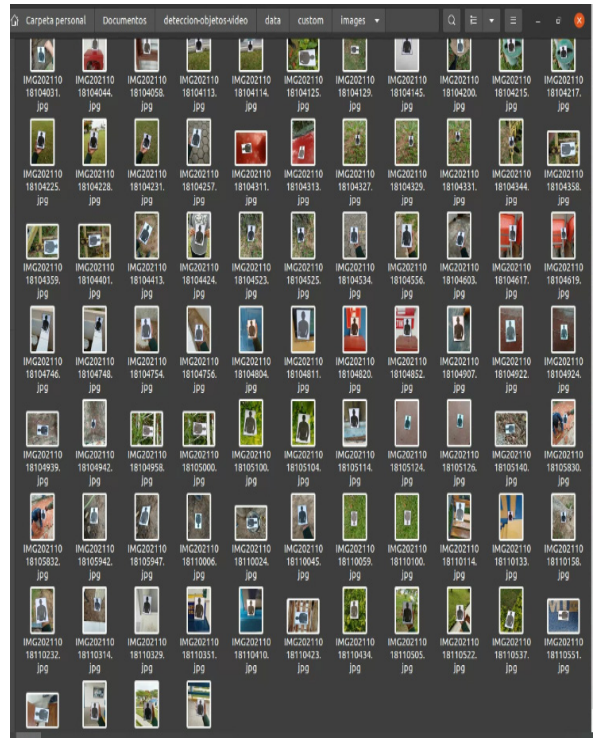
Fuente: Elaboración propia.

El resultado de este entrenamiento se puedo calibrar para detectar siluetas en estado sin movimiento, al realizar movimientos tenía mucho ruido y provocaba falsos positivos, si podía detectar siluetas estáticas es decir nos sirve para detectar objetos estáticos como son en un polígono de tiro, pero no para tiro de combate o puntería avanzada no diferenciaba un tipo de objeto con otro.

Para mejorar el sistema de visión artificial se requiere un entrenamiento más predictivo, correctivo en tiempo real que sea capaz de detectar distintos objetos, ahora probaremos el entrenamiento en base a la aplicación de redes neuronales.

Para esto vamos a crear un proyecto en sistema operativo de Ubuntu, primero debemos tomar una muestra en fotografías del objeto, tenemos una muestra de setecientas imágenes como se puede apreciar en la siguiente figura.

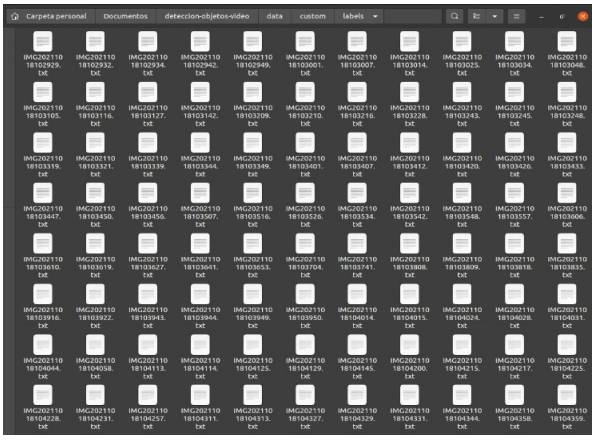
Figura 7. Muestra de imágenes para entrenamiento.



Fuente: Elaboración propia.

Estas imágenes las enmarcamos con un rectángulo en la zona que queremos detectar generando un archivo .txt con las coordenadas del rectángulo como salida, como se ve en la siguiente figura.

Figura 8. Resultado del etiquetado de imágenes.

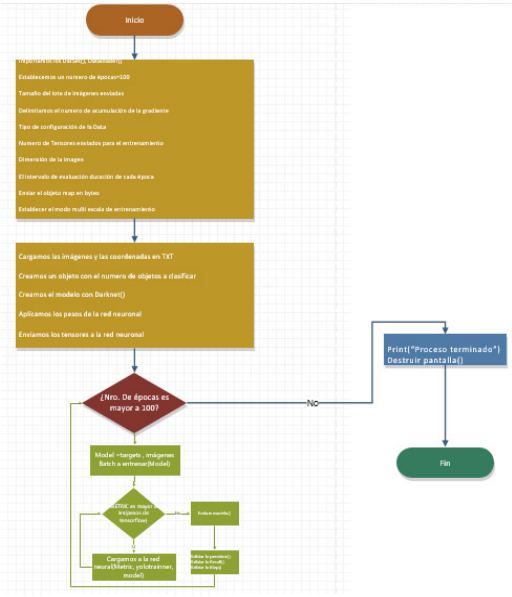


Fuente: Elaboración propia.

Nos generara un archivo clases.names, donde tendremos los distintos objetos que seleccionamos para entrenar.

Una vez tengamos la muestra, necesitamos los pesos de la red neuronal, Tensorflow nos da una estructura pre diseñada para determinar el contorno de los objetos a detectar, el modelo empleado es yoloWeight.

Figura 9. Algoritmo de entrenamiento por Épocas.



Fuente: Elaboración propia.

Una vez establecida la estructura, creamos el modelo en relación al número de objetos a detectar, posteriormente, realizamos el algoritmo de entrenamiento donde establecemos los batchs, las épocas, el gradiente entre imágenes, la cantidad de imágenes a procesar, carga de las imágenes y TX previamente preparados como se puede ver en la figura 9, para cien épocas.

Este proceso con setecientas imágenes tardo alrededor de cuarenta y ocho horas en terminar de entrenar como salidas para poder validar el avance tenemos la siguiente matriz generada por condigo.

Figura 10. Salida de entrenamiento de la Red Neuronal.

```
---- [Epoch 66/100, Batch 55/105] ----
+-------------+---------------+---------------+---------------+
| Metrics     | YOLO Layer 0  | YOLO Layer 1  | YOLO Layer 2  |
+-------------+---------------+---------------+---------------+
| grid_size   | 12            | 24            | 48            |
| loss        | 0.396808      | 0.222365      | 0.425488      |
| x           | 0.002107      | 0.046994      | 0.147448      |
| y           | 0.005864      | 0.002285      | 0.027560      |
| w           | 0.006041      | 0.002605      | 0.031622      |
| h           | 0.034126      | 0.051917      | 0.041586      |
| conf        | 0.347245      | 0.118505      | 0.176400      |
| cls         | 0.001425      | 0.000059      | 0.000871      |
| cls_acc     | 100.00%       | 100.00%       | 100.00%       |
| recall50    | 1.000000      | 1.000000      | 1.000000      |
| recall75    | 1.000000      | 0.500000      | 0.500000      |
| precision   | 0.666667      | 0.500000      | 0.222222      |
| conf_obj    | 0.972618      | 0.964956      | 0.931950      |
| conf_noobj  | 0.001957      | 0.000589      | 0.000594      |
+-------------+---------------+---------------+---------------+
Total loss 1.044661045074463
```

Fuente: Elaboración propia.

Una vez entrenado probamos el modulo, para esto incorporamos el peso entrenado en Tensorflow lo validamos con la cámara y compilamos, el resultado se ve en la figura 11.

Figura 11. Validación del entrenamiento.

Fuente: Elaboración propia.

El tirador pudo disparar y se detectó tanto la silueta como el láser óptico; en la siguiente grafica se puede evidenciar al sujeto que realizo su tiro de práctica.

ISSN 2409 - 2371

Revista Científica EMINENTE
Mayo 2023 Volumen 7, Número 1 85-93

Figura 12. Pruebas de funcionalidad.

Fuente: Elaboración propia.

Como resultado de la experimentación de la red neuronal y desarrollo de un clasificador de objetos se pudo detectar siluetas en movimiento y diferentes objetos a la vez.

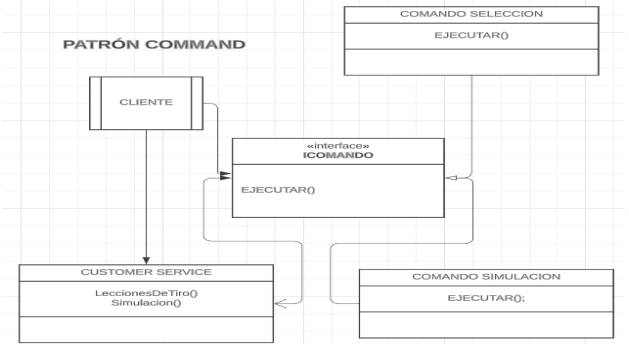
Fase de diseño de la arquitectura de aplicación con patrones de diseño y simulación de un ambiente virtual a través de realidad semi-inmersiva.

Arquitectura de patrones de diseño

Durante el proceso de análisis de los requerimientos del simulador se seleccionaron los siguientes patrones estructurales, creacionales y de comportamiento con el fin de tener un rendimiento óptimo y escalable.

Patron comand.- Para este vamos a desarrollar el manejo óptimo de la selección de tiro y el ambiente de simulación a través de una interfaz y manejador de los servicios, como se puede evaluar en la siguiente figura.

Figura 13. Comportamiento de la selección de tiro.

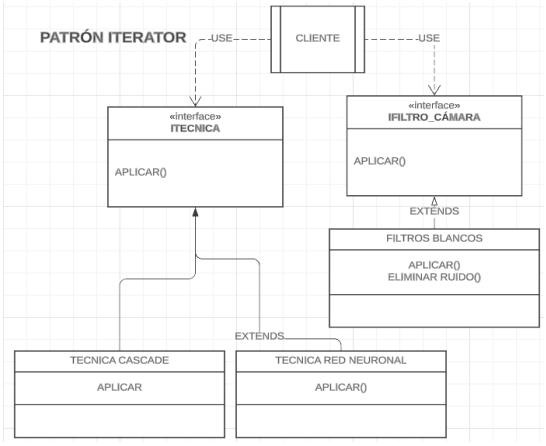


Fuente: Elaboración propia.

Ahora necesitamos una herramienta para las propiedades de la cámara y la técnica de detección a emplear en este caso aplicamos un iterador.

Patrón Iterator: En este caso manejamos dos interfaces que el cliente llama para aplicar el filtro de la red neuronal, técnica de cascade y los filtros adecuados para la detección a través de la cámara, estructurados como se ve en la figura 14.

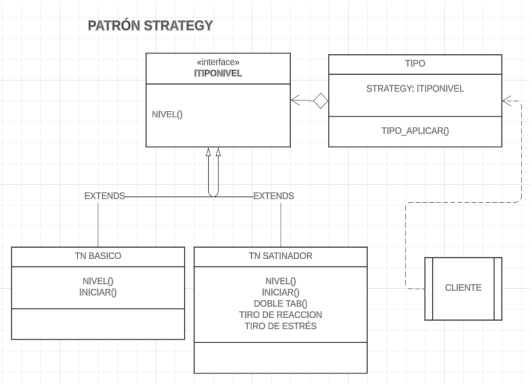
Figura 14. Comportamiento de configuración de la cámara



Fuente: Elaboración propia.

Dentro del simulador de tiro de escuela y tiro de combate tenemos una serie de lecciones de tiro, para lo cual necesitamos un manejador de comportamiento de objetos, el aplicado en este simulador es el strategy para intercambio de niveles óptimo, como se puede apreciar en la siguiente gráfica.

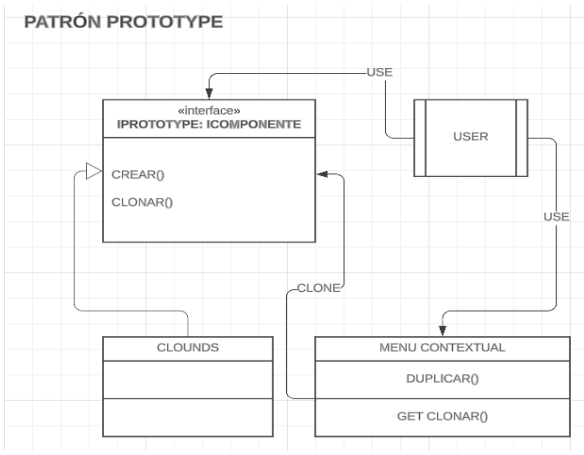
Figura 15. Comportamiento de lección de tiro.



Fuente: Elaboración propia.

En la simulación del ambiente tenemos árboles, nubes representadas por objetos en el código fuente, para evitar la dependencia de estos en cada clase el patrón adecuado a aplicar es el creacional prototype y está estructurado de la siguiente manera, figura 16.

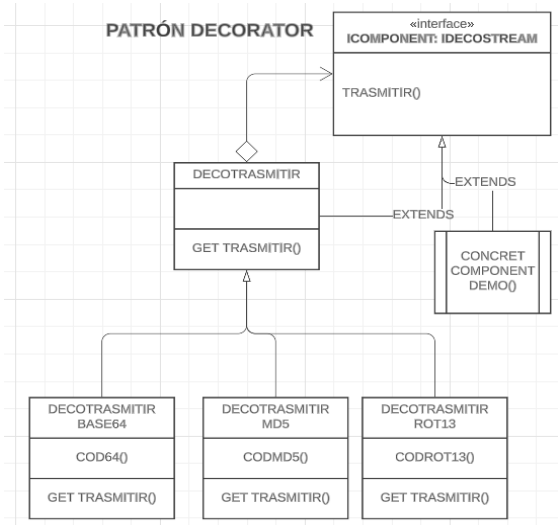
Figura 16. Comportamiento den entorno 3D.



Fuente: Elaboración propia.

Para la simulación de los datos obtenidos codificación y decodificación necesitamos encriptar la información ya sea en 2 o tres capaz según lo requerido, para esto los objetos comparte algunas propiedades que establecimos en la clase abstracta y la estructura se lo puede ver en la figura 17.

Figura 17. Comportamiento de codificación de datos.



Fuente: Elaboración propia.

Estos patrones son los que conformaran el backend del software de entrenamiento, con el fin de fortalecer la respuesta y tiro de precisión de los combatientes.

RESULTADOS

Para la Fase de la obtención, de una matriz entrenada con redes neuronales en cien épocas y muestreo aleatorio para la detección de objetos se aplicaron tres técnicas.

La primera era a base de detección de objetos con filtro HSV, donde se pudo realizar la detección el cual tenía mucho ruido de ambiente y no era óptimo para la detección de siluetas.

La segunda técnica era aplicar la teoría de Piola y James, donde se tuvo muestras positivas y negativas, pasadas por el entrenador a base de escala de grises de la imagen donde como resultado se obtuvo la detección de siluetas fijas, pero al hacer movimientos no funcionaba con la teoría de matriz de escala de grises.

La tercera técnica fue a base de tensores, en el cual teníamos una estructura de red neuronal pre diseñada en Tensorflow, para estos, seleccionamos nuestra muestra y marcamos las coordenadas de los objetos a detectar, clasificándolos con silueta y láser óptico, establecimos los parámetros para validar el entrenamiento en épocas y batch el cual como resultado nos dio la detección de múltiples objetos en movimiento que era óptimo para el sistema de entrenamiento de tiro de estrés.

Para la fase de diseño de la arquitectura de aplicación con patrones de diseño y simulación de un ambiente virtual a través de realidad semi-inmersiva se identificaron el comportamiento de los objetos.

Como se pudo apreciar se hizo el diseño y comportamiento de los objetos, creacionales, de comportamiento aplicando patrones de diseño de software, para la reutilización de código, mejorar en el rendimiento de la aplicación en consumo de recursos y escalabilidad del sistema.

Recomendaciones:

Se recomienda seguir con el proceso de investigación en el campo de la visión artificial, explorando nuevos métodos y técnicas de detección y clasificación de objetos.

Así también explorar los algoritmos genéticos para mejorar la latencia de tiro prediciendo los cambios de luz del entorno.

ISSN 2409 - 2371

Se recomienda seguir con la línea base de la investigación en seguimiento de objetos, y manuales de instrucción para puntería avanzada permitiendo el entrenamiento de tiro de estrés.

CONFLICTO DE INTERÉS

El autor declara que no tiene conflictos de interés con la presente investigación.

AGRADECIMIENTOS

Mi agradecimiento a la Dirección Nacional de Investigación Ciencia y Tecnología de la Escuela Militar de Ingeniería, a la Carrera de Ingeniería de Sistemas y al Laboratorio de Sistemas y Redes de la Unidad de Investigación Ciencia y Tecnología de la Unidad Académica Santa Cruz de la Escuela Militar de Ingeniería por apoyar esta investigación.

REFERENCIAS BIBLIOGRÁFICAS

[1] "TensorFlow Core | Machine Learning for Beginners and Experts". TensorFlow. https://www.tensorflow.org/overview (accedido el 19 de abril de 2022).

[2] E. A. Jeremías Ambrogio. "Reconocimiento de objetos a traves de Haard Cascade". CONFEDI – Consejo Federal de Decanos de Ingeniería. https://confedi.org.ar/wp-content/uploads/2020/12/Articulo1-RADI16.pdf (accedido el 19 de abril de 2022).

[3] G. Solano. "Detección de colores en OpenCV - Python (En 4 pasos) » omes-va.com". OMES. https://omes-va.com/deteccion-de-colores/ (accedido el 19 de abril de 2022).

[4] F. Morota. "Simulaciones con Realidad Semi-inmersiva, inmersiva y no inmersiva". UCEMA. https://ucema.edu.ar/publicaciones/download/documentos/740.pdf (accedido el 19 de abril de 2022).

[5] J. Solawetz. "How to Train A custom object detection Model". Medium. https://towardsdatascience.com/how-to-train-a-custom-object-detection-model-with-yolo-v5-917e9ce13208 (accedido el 18 de abril de 2022).

Bryan José Atahuichi Pinto.

Nació en Yacuiba, Tarija - Bolivia, es Ingeniero de Sistemas. Realizó Diplomado en Planificación y Desarrollo de Competencias Profesionales, Diplomado en Ethical Hacking, Diplomado en Didáctica de Aula. Es investigador en Desarrollo y Simulación de Sistemas del Centro de Investigación, Desarrollo Tecnológico e Innovación para las FF.AA. de la Escuela Militar de Ingeniería.