Juego basado en visión artificial aplicando la detección de movimiento

CatchBall

Jiménez Gonzaga Freddy Fernando
Electrónica y Telecomunicaciones
Universidad Técnica Particular de Loja
Loja, Ecuador
ffjimenez1@utpl.edu.ec

Cueva Jiménez Felix Enrique
Sistemas Informáticos y Computación
Universidad Técnica Particular de Loja
Loja, Ecuador
fecueva3@utpl.edu.ec

***Abstract* — La creación de aplicaciones basadas en la visión artificial es un campo interesante y actualmente la sociedad exige una gran demanda de dichas aplicaciones que permitan resolver problemas en tiempo real. A través de las librerías de procesamiento de imágenes (OpenCV) se pueden crear varias aplicaciones las cuales permiten que el usuario interactúe con la máquina. En el presente documento se detalla el desarrollo de un juego, para niños de 5 a 7 años de edad, que consiste en permitir al usuario interactuar con el computador e identificar objetos por medio de la detección de colores. Para realizar este programa se utilizará el lenguaje de programación Python versión 2.7 y las librerías OpenCV versión 3.1 y Pygame versión 1.9.3.**

Palabras claves — Visión arificial, Tracking, Detección de movimiento, Python, OpenCV, Pygame.

I. INTRODUCCIÓN

El campo científico de la visión artificial abarca varias disciplinas como son la matemática, el cálculo integral y deferencial, la programación, etc. La visión artificial o visión por computador incluye varios métodos que permiten analizar imágenes que se presentan dentro del mundo real con el objetivo de enviar información que pueda ser procesada por un computador. La visión por computador trata de producir un efecto similar al del ojo humano, es decir visualizar y comprender nuestro alrededor en tiempo real. Para que el computador reciba dicha información se hace el uso de dispositivos externos, los cuales cumplan con la tarea de capturar imágenes en tiempo real; en este caso se utiliza una videocámara que permite captar imágenes y enviar los datos al computador.

El objetivo del presente juego es permitir que los niños desarrollen sus capacidades motrices ya que muchas de las veces nacen niños con deficiencia motriz, y el objetivo es ayudar mediante el proceso de interacción con el computador de una manera dinámica y divertida para ellos. Para realizar el código del juego se utilizará el lenguaje de programación Python el cual aporta varias librerías diseñadas para el desarrollo de juegos; es el caso de Pygame que es una librería que permite cargar imágenes y sonidos, además de otras funciones. Y para realizar la parte de visión artificial se usará la librería OpenCV la cual está diseñada con un enfoque en aplicaciones que trabajen en tiempo real.

II. MARCO TEÓRICO

En el siguiente apartado se detalla una breve información de los métodos utilizados para desarrollar el juego, además se describe cómo actúan las librerías y métodos usados en el desarrollo del juego de manera matemática.

A. OpenCV

OpenCV es una librería de software utilizada en el campo de la visión artificial y el aprendizaje automático. Esta librería fue construida con el objetivo de proporcionar una infraestructura común para aplicaciones de visión por computador. La librería cuenta con más de 2500 algoritmos que son utilizados para detectar y reconocer caras, reconocimiento de objetos, entre otros. OpenCV pretende proporcionar un entorno de desarrollo fácil de utilizar y altamente eficiente [1].

En el presente juego, la librería OpenCV cumple varias funciones desde la captura y procesamiento de imágenes hasta la detección de movimiento y de color, funciones que serán captadas en tiempo real.

B. Pygame

Pygame es una librería de lenguaje de programación de Python libre y de código abierto para hacer aplicaciones multimedia como juegos construidos sobre la excelente biblioteca SDL. Al igual que SDL, Pygame es altamente portátil y funciona en casi todas las plataformas y sistemas operativos [2].

La librería de Pygame permite realizar varias funciones que son indispensables para la creación del siguiente juego tales como la implementación de imágenes en la ventana de salida de las imágenes captadas en tiempo real.

C. Python

Python es un lenguaje de programación multiparadigma. Esto significa que más que forzar a los programadores a adoptar un estilo particular de programación, permite varios estilos: programación orientada a objetos, programación imperativa y

programación funcional. Otros paradigmas están soportados mediante el uso de extensiones. Python usa tipado dinámico y conteo de referencias para la administración de memoria.

Una característica importante de Python es la resolución dinámica de nombres; es decir, lo que enlaza un método y un nombre de variable durante la ejecución del programa. Otro objetivo del diseño del lenguaje es la facilidad de extensión, Python puede incluirse en aplicaciones que necesitan una interfaz programable [3].

D. Detección de Objetos

La detección de objetos es la tarea para encontrar e identificar objetos en una imagen o secuencia de video. Generalmente se pueden distinguir dos partes en el proceso de detección: la extracción de características del contenido de una imagen y la búsqueda de objetos basada en dichas características. La extracción de características consiste en la obtención de modelos matemáticos compactos que "resuman" el contenido de la imagen con el fin de simplificar el proceso de aprendizaje de los objetos a reconocer [4].

Para llevar a cabo la detección de color se utilizará los espacios de colores RGB Y HSV y posteriormente usaremos una técnica para convertir la imagen RGB a HSV

1) Espacio de color RGB

El modelo de color llamado RGB es el que se utiliza en todos los sistemas que forman imágenes a través de rayos luminosos, ya sea emitiéndolos o recibiéndolos. Está formado por los tres componentes de colores primarios aditivos y como mínimo un componente de sincronismo. Los componentes de color son las señales rojo, verde y azul; siendo transmitidos cada uno independiente y aislado del resto.

El espacio RGB se representa como un cubo donde un color viene definido por la mezcla de valores de intensidad de tres colores primarios, rojo, verde y azul (figura 1). Para indicar con qué proporción es mezclado cada color, se asigna un valor a cada uno de los colores primarios, de manera que el valor "0" significa que no interviene en la mezcla y, a medida que ese valor aumenta, se entiende que aporta más intensidad a la mezcla [5].

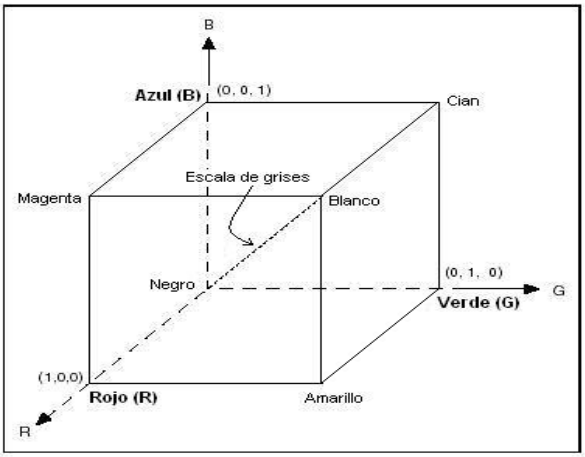

Fig. 1. Esquema del cubo de espacio de color RGB

2) Espacio de color HSV

Se representa como un grado de ángulo cuyos valores posibles van de 0 a 360°. Cada valor corresponde a un color. Disponemos de 360 grados dónde se dividen los 3 colores RGB, eso da un total de 120º por color, sabiendo esto podemos recordar que el 0 es rojo RGB (1, 0, 0), 120 es verde RGB (0, 1, 0) y 240 es azul RGB (0, 0, 1). Para colores mixtos se utilizan los grados intermedios, el amarillo, RGB (1, 1, 0) está entre rojo y verde, por lo tanto 60º. Se puede observar cómo se sigue la secuencia de sumar 60 grados y añadir un 1 o quitar el anterior.

El espacio de color HSV se representa por una región circular; una región triangular separada, puede ser usada para representar la saturación y el valor del color [6]. Normalmente, el eje horizontal del triángulo denota la saturación, mientras que el eje vertical corresponde al valor del color (figura 2).

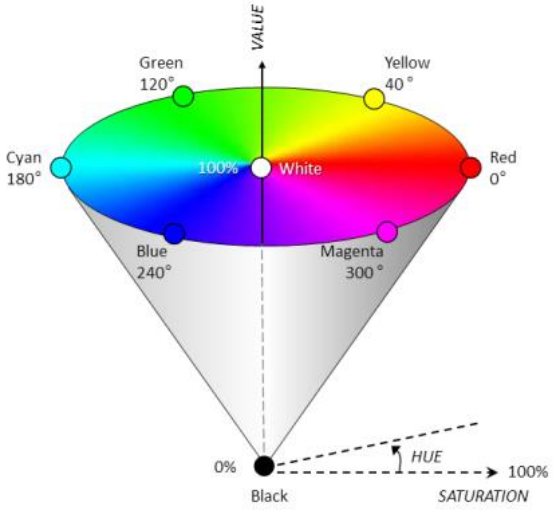

Fig. 2. Representación del espacio de color HSV

3) Transformación de RGB a HSV

Dentro de nuestro juego realizamos la operación de conversión del espacio de color RGB a HSV mediante la operación de OpenCV (cv2.cvtColor), donde intervienen las siguientes operaciones matemáticas (ecuaciones 1, 2 y 3):

Ecuaciones para conversión RGB a HSV [6]

$$H = \begin{cases} no\ definido, & si\ MAX = MIN \\ 60° \ x\ \dfrac{G - B}{MAX - MIN} + 0°, & si\ MAX = R\ y\ G \geq B \\ 60° \ x\ \dfrac{G - B}{MAX - MIN} + 360°, & si\ MAX = R\ y\ G < \\ 60° \ x\ \dfrac{B - R}{MAX - MIN} + 120°, & si\ MAX = G \\ 60° \ x\ \dfrac{R - G}{MAX - MIN} + 240°, & si\ , MAX = B \end{cases} \quad \textbf{(1)}$$

$$S = \begin{cases} 0, & si\ , MAX = 0 \\ 1 - \frac{MIN}{MAX}, & en\ otro\ caso \end{cases} \quad \textbf{(2)}$$

$$V = MAX \quad \textbf{(3)}$$

Sea MAX el valor máximo de los componentes (R, G, B), y MIN el valor mínimo de esos mismos valores. G será la intensidad del verde, B será la intensidad del azul y R será la intemsidad del rojo.

E. Detección de Discontinuidades

El método utilizado para buscar discontinuidades es la correlación de la imagen con una máscara. En este procedimiento se realiza el producto de los elementos de la máscara por el valor de gris correspondiente a los pixels de la imagen encerrados por la máscara. La respuesta a la máscara de cualquier pixel de la imagen viene dada por

$$R = \sum_{i=1}^{9} w_i z_i \quad \textbf{(4)}$$

donde Zi es el nivel de gris asociado al pixel de la imagen con coeficiente de la máscara Wi (ecuación 4) [7]. Como suele ser habitual, la respuesta de la máscara viene referida a su posición central. Cuando la máscara esté centrada en un pixel de borde de la imagen, la respuesta se determina empleando el vecindario parcial apropiado [7].

F. Detección de Bordes

En el desarrollo del juego se hace el uso de imágenes externas que no son captadas por la videocámara. Para realizar la detección de la imagen mostramos los bordes de dicha imagen por medio del método del gradiente [7]. En este caso la imagen que se desea detectar mostrando sus bordes en una pelota de futbol (figura 3).

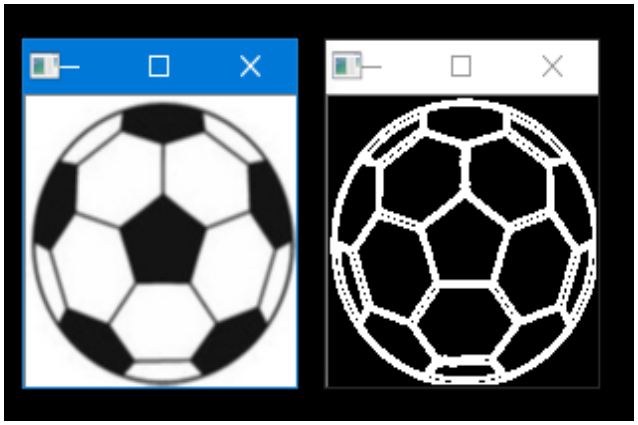

Fig. 3. Detección de bordes

Para realizar esta operación hacemos uso del operador de Sobel [7]. Tiene la ventaja de que proporcionan un suavizado además del efecto de derivación. Ya que la derivación acentúa el ruido, el efecto de suavizado es particularmente interesante, puesto que elimina parte del ruido. El requisito básico de un operador de derivación es que la suma de los coeficientes de la máscara sea nula, para que la derivada de una zona uniforme de la imagen sea cero (ecuaciones 5 y 6) [7].

Las derivadas según el operador de Sobel vienen dadas por

$$R_x = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7) \quad \textbf{(5)}$$

$$R_y = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3) \quad \textbf{(6)}$$

cuyo resultado se evidencia en la figura 3.

III. DESARROLLO

En la siguiente sección se detalla de una manera más clara como se llevó a cabo el desarrollo del juego. Se especifica el reconocimiento del objeto a través del color, el uso de técnicas para insertar las imágenes de modo aleatorio y la culminación del juego.

A. Reconocimiento del Objeto

Para realizar el reconocimiento del objeto debemos realizar primeramente el cambio de espacio de color, es decir convertir el espacio RGB a HSV usando la función (cv2.COLOR_BGR2HSV). Una vez realizado el cambio de espacio de color podemos extraer un objeto de color definido por el usuario, en este caso verde. Luego debemos aplicar filtros que permitan "limpiar" la imagen, para ello usamos los filtros de erosión y dilatación (figura 4). En OpenCV la erosión se representa como cv2.erode, y la dilatación como cv2.dilate [8].

Aplicadas las técnicas del cambio de espacio de color y la técnica de filtros, podemos localizar la posición del objeto mediante el color definido por el usuario (figura 5). En OpenCV usamos la función (cv2.moments), los "moments" de la imagen nos permiten calcular algunas características como el centro de masa del objeto, el área del objeto, etc. Matemáticamente para extraer datos como el área y el centroide de la imagen sería con las siguientes ecuaciones (ecuaciones 7 y 8):

Ecuaciones para extraer are y centroide de la imagen [9]

$$C_z = \frac{M_{10}}{M_{00}} \quad \textbf{(7)}$$

$$C_y = \frac{M_{01}}{M_{00}} \quad \textbf{(8)}$$

Fig. 4. Uso del filtro de erosión

Fig. 5. Localización de la posición del objeto

Finalmente mostramos un circulo en la posición donde se encuentra el objeto usando la función de OpenCV (cv2.circle) (figura 6) [9].

Fig. 6. Imagen detectada por la función cv2.circle

B. Insertar Imágenes en Orden Aleatorio

Una clave del desarrollo del juego es hacer que las imágenes de la pelota aparezcan en orden aleatorio para poner dificultad al niño (figura 7). Para realizar esta parte usamos una de las librerías de Python que en este caso es random. En el lenguaje de programación Python random se representa mediante la siguiente función (random.choice) [10].

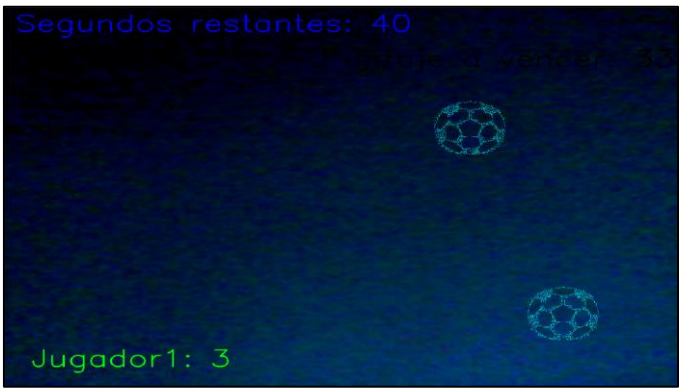

Fig. 7. Posicionamiento aleatorio de pelotas

C. Ejecución del juego

Como ya se ha mencionado anteriormente para que el usuario pueda jugar necesitará utilizar un objeto del mismo color, el cual se definió en el código. En este caso un objeto de color verde como se muestra en la figura 5.

Una vez iniciado el juego se detectará el objeto de color verde automáticamente y empezará a seguir su movimiento (figura 8).

Fig. 8. Ejecución del juego

Cuando el niño mueva el objeto de color verde hacia las pelotas se detectará y el contador empezara a subir. Este puntaje será presentado en una imagen final e indicará el puntaje total. La posición donde se muestre el puntaje será visualizada en la parte inferior izquierda de la pantalla y estará de color verde (figura 9).

Fig. 9. Visualización del puntaje

El juego debe constar con un límite de tiempo que permita dar a conocer al usuario el tiempo que tiene para jugar. El tiempo límite será definido por la librería de Python (time.time()) y donde el usuario ingresará el tiempo límite de duración del juego, en este caso 60 segundos. El tiempo será visualizado en la parte superior izquierda de la pantalla y será de color azul (figura 10).

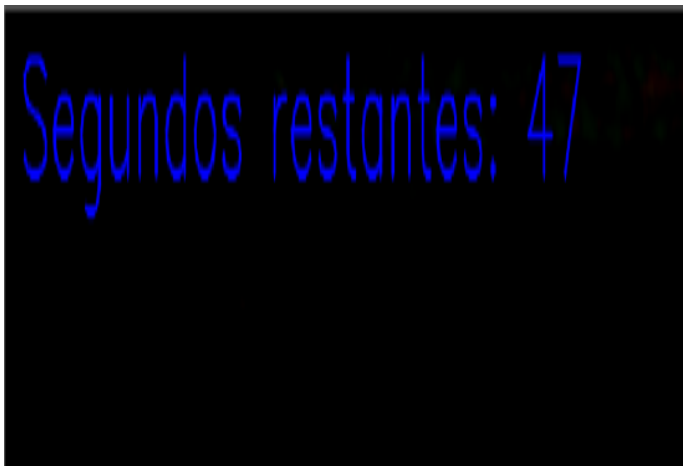

Fig. 10. Visualización de los segundos restantes.

Para hacer que el juego sea más desafiante se insertó un puntaje aleatorio que el jugador tiene que superar. Para insertar el puntaje aleatorio se utiliza la librería de Python (random.randint()) donde el usuario debe insertar el rango de puntajes en esta caso su utiliza un rango de 25 a 60. El puntaje a vencer se visualizará en la parte superior derecha de la pantalla y será de color negro (figura 11).

Fig. 11. Visualización del puntaje a vencer

Finalmente, una vez el tiempo llegado a 0 mostrará una imagen final. En el código se utiliza un condicional (if) en donde se visualizará una imagen en caso de que el puntaje del jugador sea menor al puntaje a vencer (figura 12), o se mostrará otra imagen en caso de que el puntaje del jugador supere el puntaje a vencer (figura 13).

Fig. 12 Imagen final si puntaje del jugador es menor al puntaje a vencer

Fig. 13. Imagen final si puntaje del jugador es mayor al puntaje a vencer

Mediante nuestras especificaciones indicamos que al presionar la tecla ”q” se cerrarán las ventanas del juego.

IV. RESULTADOS

Para probar la eficiencia del juego se realizó pruebas para saber en que distancia no se puede detectar el color (tabla 1).

Distancia (m)	Verde
0,5 metros	Detectado
1 metro	Detectado
2 metros	Detectado
2,5 metros	No Detectado
3 metros	No Detectado

Tabla 1. Resultados de la detección del color aplicando distancias

V. CONCLUSIONES

A partir de los 2,5 metros podemos evidenciar que la detección del color empieza a fallar por lo que el usuario deberá situarse a una distancia con un rango de1 a 2 metros para hacer uso del juego.

En algunos casos la detección del color falla por la variación de la luz en el ambiente, por lo que es recomendable trabajar en un ambiente controlado.

El juego debería utilizarse con una videocámara externa ya que la videocámara de la portátil difiere con la cantidad de luz que se presenta en el ambiente lo que conlleva a varios problemas en la detección de color.

El uso de las librerías OpenCV y Pygame son muy eficaces para el desarrollo de juegos basados en la visión artificial.

VI. REFERENCIAS

[1] OpenCV Team, «OpenCv» 2017. [En línea]. Dsiponible: http://opencv.org/about.html. [Último acceso: 11 Julio 2017].

[2] Pygame, «Pygame» [En línea]. Disponible: http://www.pygame.org/wiki/about. [Último acceso: 11 Julio 2017].

[3] A. Kuchling, «Python» 3 Julio 2010. [En línea]. Dsiponible: https://docs.python.org/2/whatsnew/2.7.html. [Último acceso: 11 Julio 2017].

[4] I. Kokkinos, «Introduction to Classification» [En línea]. Disponible: http://cvn.ecp.fr/personnel/iasonas/course/Lecture_1.pdf. [Último acceso: 11 Julio 2017].

[5] Fundación Wikimedia, Inc., «Wikipedia» 23 Junio 2017. [En línea]. Disponible: https://es.wikipedia.org/wiki/RGB. [Último acceso: 11 Julio 2017].

[6] Fundación Wikimedia, Inc., «Wikipedia» 9 Junio 2017. [En línea]. Dsiponible: https://es.wikipedia.org/wiki/Modelo_de_color_HSV. [Último acceso: 11 Julio 2017].

[7] M. Marcos, «Técnicas Clásicas de Segmentación de Imagen» 4 Mayo 2004. [En línea]. Dsiponible: http://lmi.bwh.harvard.edu/papers/pdfs/2003/martin-fernandezCOURSE03b.pdf. [Último acceso: 11 Julio 2017].

[8] OpenCV Team, «OpenCV» 10 Noviembre 2014. [En línea]. Dsiponible: http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_morphological_ops/py_morphological_ops.html. [Último acceso: 11 Julio 2017].

[9] OpenCV Team, «OpenCV» 11 Julio 2017. [En línea]. Disponible: http://docs.opencv.org/trunk/dd/d49/tutorial_py_contour_features.html. [Último acceso: 11 Julio 2017].

[10] Python Software Foundation, «Python,» [En línea]. Disponible: https://docs.python.org/2/library/random.html. [Último acceso: 11 Julio 2017].