Katherine Michelle Ortiz Burgos
Código: 506221041
Juan Esteban Acosta Rodriguez
Código: 506221068
Profesor: Oscar Alexander Mendez Aguirre
Fundación Universitaria Konrad Lorenz

Prototipo Face : Detección de Personas y Objetos

I. Resumen

Este prototipo consiste en una interfaz web para la detección de objetos en tiempo real a través de una cámara web utilizando el modelo COCO-SSD (Common Objects in Context - Single Shot MultiBox Detector) proporcionado por ml5.js, una biblioteca de machine learning en JavaScript basada en TensorFlow.js.s. Al permitir la interacción inmediata con el entorno visual a través de una cámara web, ofrece aplicaciones potenciales en una amplia gama de campos,como la seguridad y vigilancia. Su implementación basada en web facilita su uso y adaptación por parte de desarrolladores, promoviendo la innovación y la escalabilidad en el ámbito de la visión por computadora y el aprendizaje automático.

II. Introducción

El proyecto de detección de objetos en tiempo real que hemos desarrollado presenta una integración entre el frontend y el backend para ofrecer una experiencia completa de detección de objetos a través de una interfaz web.

El frontend de la aplicación se encarga de la parte visible para el usuario en su navegador web. Esto incluye la presentación de la transmisión de video en tiempo real desde la cámara web, así como la visualización de los resultados de la detección de objetos. Además, ofrece controles interactivos, como botones para activar y desactivar la cámara, así como para iniciar y detener el proceso de detección de objetos.

Por otro lado, el backend de la aplicación maneja la lógica detrás de la detección de objetos. Utiliza el modelo COCO-SSD proporcionado por la biblioteca ml5.js para detectar objetos en el flujo de video en tiempo real. El backend también se encarga de procesar los datos recibidos del modelo de detección y enviar los resultados al frontend para su visualización.

Esta colaboración entre el frontend y el backend permite que la aplicación funcione de manera fluida y eficiente. Mientras que el frontend proporciona una interfaz intuitiva para que los usuarios interactúen con la detección de objetos, el backend se encarga de la parte pesada del procesamiento de datos y la detección de objetos en sí misma.

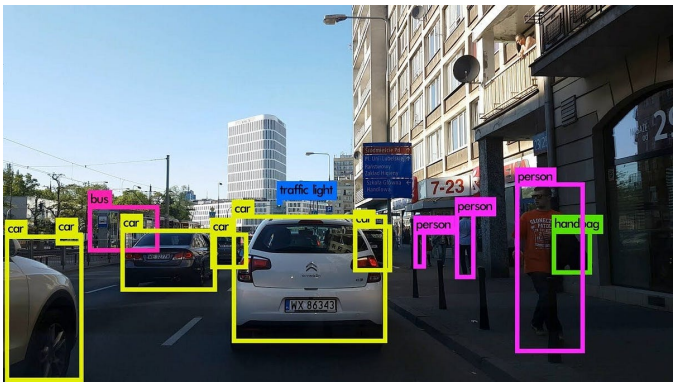

Figura 1. Detección De Objetos.

III. Diseño y desarrollo de la API backend

Para comprender la funcionalidad subyacente que impulsa la detección de objetos en tiempo real en nuestra aplicación web, es crucial explorar el papel del backend, representado por el archivo app.js. Este componente actúa como el motor detrás de la escena, gestionando la lógica de detección de objetos utilizando el modelo COCO-SSD y facilitando la comunicación entre la interfaz de usuario y el modelo de machine learning. A través de un conjunto de funciones y eventos, app.js se encarga de procesar continuamente el flujo de video capturado por la cámara web, identificar objetos en cada fotograma y transmitir los resultados pertinentes de vuelta al frontend para su presentación en tiempo real. En esta sección, exploraremos en detalle cómo app.js desempeña un papel crucial en la creación de una experiencia fluida y dinámica para los usuarios al brindar la capacidad de detección de objetos en tiempo real a través de una interfaz web accesible y fácil de usar.Por lo tanto, explicaremos el app.js:

1. El archivo app.js despliega una serie de variables y funciones cruciales para la funcionalidad del backend de nuestra aplicación de detección de objetos en tiempo real. En primer lugar, se inicializan las variables, incluyendo video para capturar el video de la cámara, detector para almacenar el modelo COCO-SSD, detections para almacenar los resultados de la detección de objetos, y otras variables para controlar la visibilidad del video y el estado de la detección. Además, se establecen referencias a los elementos de la interfaz de usuario para controlar la activación y desactivación de la cámara y la detección de objetos.
2. La función preload() se encarga de cargar el modelo COCO-SSD proporcionado por la biblioteca ml5.js,

mientras que setup() inicializa el lienzo de dibujo y configura la captura de video desde la cámara web.

3. La función draw() se ejecuta continuamente y se encarga de dibujar el video capturado en el lienzo, así como de procesar los resultados de la detección de objetos y contar el número de personas detectadas en cada fotograma.
4. Las funciones boundingBox() y drawLabel() se utilizan para dibujar cuadros delimitadores y etiquetas sobre los objetos detectados en el video.
5. La función onDetected() se activa cuando se detecta un objeto y actualiza la lista de detecciones, filtrando solo los objetos de tipo ”persona”. La función detect() se encarga de iniciar el proceso de detección de objetos utilizando el modelo COCO-SSD.
6. Por último, las funciones toggleVideo() y toggleDetecting() controlan la activación y desactivación de la cámara y la detección de objetos, respectivamente, actualizando la interfaz de usuario en consecuencia.

IV. Creación de la interfaz gráfica y conexión con el backend

El archivo index.html es la página principal de nuestra aplicación de detección de objetos en tiempo real. Aquí se define la estructura HTML de la interfaz de usuario y se establecen los enlaces con los archivos JavaScript y CSS necesarios para su funcionamiento. Vamos a analizar sus componentes y su conexión con la parte de backend y frontend:

Estructura HTML

1. En el ”head” del documento se enlazan las bibliotecas p5.js y ml5.js, necesarias para la manipulación del lienzo y la detección de objetos, respectivamente.
2. Se establece un enlace al archivo styles.css que contiene estilos para la apariencia visual de la página.
3. Se define un contenedor ”div” con id=”canvasContainer” para albergar los elementos canvas donde se mostrará el video y los resultados de detección de objetos.
4. Se crea otro contenedor ”div” con id=”buttonsContainer” para contener los botones de control de la cámara y la detección de objetos.
5. Se incluyen dos botones con id=”videoAction” y id=”detectionAction” para activar/desactivar la cámara y para iniciar/detener la detección de objetos, respectivamente.
6. Al final del ”body”, se enlaza el archivo app.js, que contiene la lógica de la aplicación.

Estilos CSS

1. Se definen estilos para el cuerpo (body) de la página, incluyendo una imagen de fondo y configuraciones de diseño flexbox para centrar los elementos verticalmente.
2. Se establecen estilos para el contenedor de los canvas (canvasContainer), configurando su disposición y espaciado.
3. Los canvas dentro de canvasContainer tienen bordes negros y un ancho fijo, con un pequeño margen entre ellos para separación visual.
4. Los botones (videoAction y detectionAction) tienen un ancho y alto fijo, un estilo de botón personalizado y una posición absoluta para fijarlos en la parte superior de la página. La posición izquierda se ajusta para centrar los botones horizontalmente en la página.

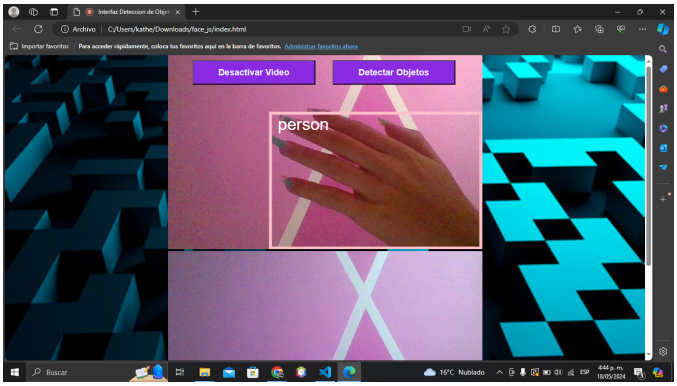

Figura 2. Prototipo Face.

V. Conclusiones

1. La capacidad de interactuar con la detección de objetos a través de un navegador web y una cámara web facilita su uso en diversos contextos y entornos, promoviendo así su usabilidad y accesibilidad.
2. Este prototipo ilustra la capacidad de las tecnologías web y de aprendizaje automático para crear soluciones innovadoras y avanzadas. La combinación de bibliotecas como ml5.js y p5.js permite aprovechar el potencial de la visión por computadora y el machine learning para desarrollar aplicaciones sofisticadas.
3. La aplicación de detección de objetos en tiempo real presenta una amplia variedad de aplicaciones prácticas en diversos campos. Su versatilidad y flexibilidad la convierten en una herramienta valiosa para mejorar procesos y experiencias en diferentes sectores.

VI. Referencias

1. Google Colab. (s/f-b). Google.com. Recuperado el 18 de mayo de 2024, de https://colab.research.google.com/drive/1MIoprqtyD8XjFyaa_rjHZZ8xW6RRN4yp?hl=es