

PAVIMENTA2 MANUAL DE USO

Actualización octubre 2024

ÍNDICE

INTRODUCCIÓN.....	3
DEFINICIONES	4
¿Qué es Computer vision?	4
Yolov8.....	5
¿Qué es el Sistema Pavimenta2?	6
MODELOS DE PAVIMENTA2.....	7
Modelo de pavimento.....	7
USO DEL SISTEMA	9
Recolección de datos	9
Alistamiento del vehículo	9
Proceso de recolección de datos	9
Detalles de datos de entrada al sistema	12
Descarga de librerías	13
Enlaces	13
Librería en Python	13
Quick Start	13
Flujos de trabajo.....	15
Uso de procesadores	16
crear una aplicación	17
Resultados.....	17
Análisis de resultados – Casos de uso	19
INFORMACIÓN DE CONTACTO	23
LICENCIA DE SOFTWARE.....	23
Licencia.....	24
Definiciones	24
Derechos del BID.....	24
Derechos del Usuario sobre Software Derivado	24
Restricciones.....	25
Reconocimiento	25
Nombre y Logo del BID	25

INTRODUCCIÓN

La infraestructura vial de un país tiene impactos directos e indirectos en su economía, ya que las carreteras no solo facilitan la conectividad, sino que son esenciales para el acceso a actividades económicas, servicios y el desarrollo social. Una red vial en condiciones adecuadas permite una circulación eficiente de bienes y personas, lo que favorece el comercio, la integración de comunidades y la provisión de servicios básicos.

En este contexto, una gestión eficaz del sistema vial es fundamental. Para asegurar que la red de carreteras se mantenga en condiciones óptimas y prevenir daños que puedan afectar la movilidad y la economía, es necesario un monitoreo continuo y confiable del estado de las vías. La inspección visual, aún empleada por muchos países, resulta costosa y laboriosa. Por ello, es esencial adoptar tecnologías avanzadas que permitan una evaluación más eficiente del estado de la infraestructura vial, facilitando la planificación e implementación de inversiones en mantenimiento, rehabilitación y mejora, lo que repercute positivamente en la economía y en la calidad de vida de los ciudadanos.

Pavimenta2 es una herramienta desarrollada por el Departamento de Infraestructura del Banco Interamericano de Desarrollo (BID) que permite la identificación de defectos en pavimentos localizados en autopistas y carreteras. La herramienta automatiza el análisis del estado del pavimento con el uso de inteligencia artificial (IA) a partir de vídeos grabados por una cámara en un vehículo convencional.

La herramienta reduce el proceso de lo que pueden ser varios años de medir y registrar manualmente cada falla del pavimento, a solo algunas semanas de recopilación de videos y a minutos u horas de procesamiento de las imágenes. De manera similar, cada informe produce un registro electrónico, lo que reduce la posibilidad de perder archivos y de cometer errores humanos al registrar notas, en la misma medida en que reduce la duplicidad del trabajo en el futuro.

DEFINICIONES

¿QUÉ ES COMPUTER VISION?



Ilustración 1 - El sistema visual humano no tiene ningún problema en interpretar las sutiles variaciones de translucidez y sombreado en esta fotografía y segmentar correctamente el objeto de su fondo.

Como humanos, percibimos la estructura tridimensional del mundo que nos rodea con aparente facilidad. Tenemos una vívida percepción tridimensional cuando miramos un balón de fútbol en el momento de un tiro penal. En este caso, distinguimos la forma y la translucidez del balón de fútbol y del césped, a través de los sutiles patrones de luz y sombreado que se reproducen sobre su superficie (*Ilustración 1*).

Ahora bien, las máquinas ya ven y semejan el ojo humano, y es lo que llamamos **Computer Vision**. Computer Vision es un área de la Inteligencia Artificial que estudia el procesamiento de imágenes por parte de un computador, es decir, es el área que estudia formas de dar a las máquinas la capacidad de interpretar visualmente la información de lo que vemos.

Los modelos avanzados que utilizamos en visión artificial se desarrollan generalmente en física (radiometría, óptica y diseño de sensores) y gráficos por computadora. Ambos campos modelan cómo se mueven y animan los objetos, a medida que la luz se refleja en sus superficies, se extiende a través de la atmósfera, se refracta a través de las lentes de la cámara (o los ojos humanos), y finalmente se proyecta en un plano de imagen plano (o curvo). Aunque los gráficos por computadora aún no son perfectos (ninguna película completamente computarizada con personajes humanos ha logrado cruzar el extraño valle que separa a los humanos reales de los robots androides y los humanos animados por computadora), en dominios limitados como la representación de una escena fija compuesta de objetos cotidianos o la animación de criaturas extintas como los dinosaurios, la ilusión de la realidad es perfecta.¹

Computer Vision intenta describir el mundo que vemos en una o más imágenes y reconstruir sus propiedades, como la forma, la iluminación y las distribuciones de color.

¹ <https://www.ibm.com/topics/computer-vision#:~:text=Computer%20vision%20is%20a%20field,recommendations%20based%20on%20that%20information.>

YOLOv8

YOLOv8 (You Only Look Once, Versión 8) es un modelo de detección de objetos en tiempo real que identifica objetos específicos en videos, transmisiones en vivo o imágenes. YOLO utiliza recursos aprendidos por una red neuronal convolucional profunda para detectar un objeto². YOLO fue creado por Joseph Redmon y Ali Farhadi en la Universidad de Washington y fue lanzado por primera vez en 2015.

YOLOv8, desarrollado por Ultralytics³, es una actualización de las versiones anteriores que mejora el rendimiento, la flexibilidad y la eficiencia del modelo. Esta versión optimiza la precisión en la detección y ofrece mejor adaptabilidad a diferentes dispositivos y aplicaciones, manteniendo su alto desempeño en tiempo real⁴.

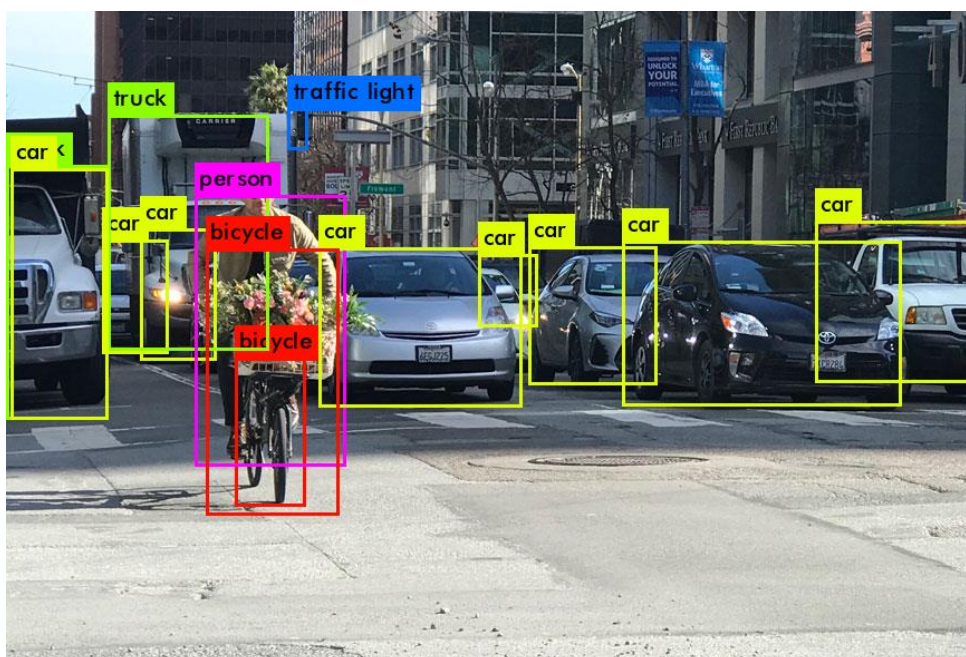


Ilustración 2 - Ejemplo de Computer Vision.

Los sistemas de clasificación de objetos son utilizados por los programas de Inteligencia Artificial (IA) para percibir objetos específicos en una clase como sujetos de interés. Los sistemas clasifican los objetos en imágenes en grupos donde los objetos con características

² Definición: Una Red Neuronal Convolucional (ConvNet/Red Neuronal Convolucional/CNN) es un algoritmo de Deep Learning que puede capturar una imagen de entrada, asignar importancia (pesos y viseras que se pueden aprender) a diversos aspectos/objetos de la imagen y ser capaz de diferenciar uno del otro. El preprocesamiento requerido en un ConvNet es mucho menor en comparación con otros algoritmos de clasificación. Mientras que en los métodos primitivos los filtros son hechos a mano, con suficiente entrenamiento, convNets tienen la capacidad de aprender estos filtros / características.

³ Los modelos entrenados usando Ultralytics YOLO están bajo la licencia AGPL-3.0. Esta licencia requiere que todo el software y modelos AI desarrollados sean de código abierto y bajo la misma licencia. Para más información consultar <https://www.ultralytics.com/license>.

⁴<https://docs.ultralytics.com/#where-to-start>

similares se colocan juntos, mientras que otros se descuidan a menos que se programe para hacer lo contrario.⁵

¿QUÉ ES EL SISTEMA PAVIMENTA2?

El mantenimiento eficiente de las carreteras requiere un sistema de monitoreo confiable, y un método directo es la inspección visual humana. Sin embargo, para las entidades públicas de América Latina y el Caribe, este método es difícil de aplicar ya que es caro, laborioso y poco efectivo. Por lo tanto, se han desarrollado varias soluciones para la inspección automática de daños en carretera, incluidos los métodos basados en vibraciones, láser, e imágenes. Mientras que la detección por métodos de vibración se limita a las partes de la carretera en contacto, los métodos de escaneo láser proporcionan información precisa sobre el estado de las carreteras; sin embargo, estos métodos son costosos y requieren el cierre de carreteras. Mientras tanto, los métodos de procesamiento de imágenes son baratos y mucho más rápidos.

Cambiar el paradigma de cómo orientar las inversiones para que sean más eficaces ha permitido que la División de Transporte del Banco Interamericano de Desarrollo (BID) desarrolle una nueva herramienta revolucionaria para evaluar las condiciones de las carreteras. El Banco ha creado una herramienta para identificar, medir y cuantificar fallas en el pavimento utilizando inteligencia artificial (IA).

La herramienta Pavimenta2 permite a los usuarios acceder a los modelos y el código necesario para procesar sus videos e imágenes, y obtener reportes de salida que permiten identificar las ubicaciones geográficas de cada falla, y las cantidades totales para cada tipo de falla. Pavimenta2 puede medir cantidades y ubicaciones de líneas borrosas, grietas lineales, grietas transversales, grietas de cocodrilo y otras fallas. Esta herramienta automatiza el proceso de cargar, documentar, medir y registrar cada falla, que ha sido obtenida de las imágenes al conducir a través de la red vial con un teléfono celular montado o GoPro®. Pavimenta2 reduce el proceso, de lo que pueden ser varios años de medir y registrar manualmente cada falla del pavimento, a varias semanas de recopilación de videos y varios minutos de procesamiento de cada video. De manera similar, cada informe finalizado produce un registro electrónico, lo que reduce la posibilidad de que se pierdan archivos, de cometer errores humanos al registrar notas, y la duplicidad del trabajo en el futuro.

⁵<https://viso.ai/deep-learning/yolov3-overview/#:~:text=YOLOv3%20AI%20models-,What%20is%20YOLOv3%3F,Joseph%20Redmon%20and%20Ali%20Farhadi.>

MODELOS DE PAVIMENTA2

MODELO DE PAVIMENTO

Este modelo se basa en la versión 8 de Yolo, que permite identificar objetos o elementos dentro de una imagen. Se ha vuelto a entrenar para detectar seis tipos de fallas:


1. Grietas e intervalos lineales longitudinales: son grietas y juntas de secciones en el pavimento que se ubican longitudinalmente.
2. Grietas e intervalos lineales transversales: Son grietas y juntas de secciones en el pavimento que se encuentran cruzadas transversalmente.
3. Piel de cocodrilo: Son grupos de fisuras que generalmente se encuentran en bloques de tal manera que se parecen a la piel escamosa de un cocodrilo.
4. Protuberancias o agujeros: Estos son agujeros o superficies sobresalientes que se encuentran en el pavimento.
5. Desenfoque del paso peatonal: Ocurre cuando la demarcación en el piso está fuera de foco o borrosa.
6. Desenfoque de línea blanca: Ocurre cuando la demarcación en el piso está fuera de foco o borrosa.

El modelo devuelve la siguiente información:

1. Clase del objeto identificado.
2. Ubicación dentro de la imagen.
3. Probabilidad de la detección.

El catálogo de imágenes presentado a continuación (*Tabla 1*) muestra el tipo de fallas encontrados normalmente en los pavimentos y con los cuales se entrenan los algoritmos de Computer Vision para la detección de problemas en la malla vial.

Tabla 1 – Catálogo de fallas de pavimentos

CLASE	IMAGEN
Grieta e Intervalo Lineal Longitudinal (D00)	

CLASE	IMAGEN
Grieta e Intervalo Lineal Transversal (D10)	
Piel de cocodrilo (D20)	
Protuberancia, baches (D40)	
Desenfoque de paso peatonal (D43)	
Desenfoque de línea blanca (D44)	

USO DEL SISTEMA

RECOLECCIÓN DE DATOS

ALISTAMIENTO DEL VEHÍCULO

Los vídeos utilizados se pueden generar desde un teléfono móvil o cámaras como GoPro, como muestra en la *Ilustración 3*. Es fundamental que el dispositivo que se utilice tenga función GPS ya que se requiere tener archivos georreferenciados para localizar las fallas. Algunos modelos de GoPro que tienen función GPS son Hero9 Black y GoPro Hero11 Black.



Ilustración 3 – Ejemplo colocación de cámara en vehículo

Al ajustar la cámara para grabar, se debe tener cuidado de que la cámara filme toda la plataforma de la carretera. También se debe asegurar que la cámara esté bien instalada y se mantenga fija en todo el recorrido.

PROCESO DE RECOLECCIÓN DE DATOS

El sistema necesita recibir archivos con la información visual de la carretera a procesar. Esto se refiere a un video o un conjunto de imágenes. Se deben tener en cuenta una serie de condiciones a la hora de tomar las imágenes para realizar el procesamiento y la evaluación de la malla vial.

- Es altamente sugerido tomar fotos cada 2 metros, de esta forma aseguramos no perder información relevante del pavimento durante el recorrido. Si no es posible, se recomienda que el intervalo no sea superior a 20 metros.
- La posición de la cámara debe ser tal que la imagen capturada se vea como se muestra en la *Ilustración 4* y *Ilustración 5*:
 - No se observa el capo del vehículo
 - Se está enfocando el canal que se está estudiando
 - Las señales de tránsito se observan con claridad, en específico en el canal que se está estudiando.



Pavimento ocupa
aproximadamente 75%
de la imagen

Ilustración 4 – Ejemplo de imagen capturada.



Ilustración 5 – Ejemplo de imagen capturada.

- La imagen debe enfocar el carril que se está estudiando.
- La imagen debe tomarse de día, con suficiente luz solar, y evitando:
 - Lluvia

- baja visibilidad
- oscuridad de la imagen por el clima
- La cámara debe estar fuera del vehículo para evitar las deformaciones que produce el parabrisas sobre la imagen. El lugar ideal está en la orilla del capo del vehículo.
- La cámara no debe tener efectos especiales como ojo de pez.
- La inclinación de la cámara debe ser tal que se observe detalles del pavimento sin perder los detalles de las señales. Se sugiere colocar la cámara aproximadamente a 1 metro del suelo con una inclinación aproximada de 70°.
- La velocidad al tomar videos debe ser aproximadamente 60 km/h, pero se recomienda esta opción solo si no es posible tomar fotografías como se explicaba anteriormente.
- Se debe hacer un recorrido por sentido observado. Si la vía cuenta con dos sentidos, se debe hacer un recorrido por cada uno enfocando ese segmento.
- Es necesario que el vehículo no permanezca estático durante el corrido.
- El archivo GPS que procesa el sistema es en formato NMEA. En particular el sistema usa el estándar \$GPGGA o \$GPRMC, que normalmente van en un archivo .LOG
- No se puede procesar información sin el registro GPS de ubicación.
- Las imágenes deben tener buena resolución, 1920×1080 por ejemplo. Pero no debe haber sido procesada por otro software dado que degradan la calidad de la imagen al momento de ingresarla en el sistema. Como se trata de detectar líneas o grietas delgadas, al modificar la calidad original de la imagen esas grietas tienden a difuminarse durante el proceso.
- En general, los segmentos donde la cámara no apunte al pavimento (como cuando hay un vehículo enfrente), donde el vehículo esté detenido o cuando está sobrepasando otro vehículo, no tendrán resultados válidos.

Ejemplos de situaciones que evitar: A continuación, se presentan algunos ejemplos (*Ilustración 6, Ilustración 7, Ilustración 8*) de situaciones que se deben evitar, para el buen funcionamiento del algoritmo y la correcta evaluación de los problemas en la malla vial.



Ilustración 6 – Ejemplo de imagen no apta para procesar



Ilustración 7 – Ejemplo de imagen no apta para procesar



Ilustración 8 – Ejemplo de imagen no apta para procesar

DETALLES DE DATOS DE ENTRADA AL SISTEMA

Actualmente el sistema es capaz de procesar dos formatos de archivos geo-referenciados:

1. Posiciones almacenadas directamente en las imágenes: cuando las imágenes ya tienen embebidas las posiciones. Este formato solo está permitido en el caso que la entrada visual sean imágenes.

2. Archivo geo-referenciado en formato NMEA⁶, cuyo formato normalmente se visualiza de la siguiente forma:

\$GPGGA,181908.00,3404.7041778,N,07044.3966270,W,4,13,1.00,495.144,M,29.200,M,0.10,0000*40.

DESCARGA DE LIBRERÍAS

ENLACES

Para más información y descarga de la librería de Pavimenta2 visite el sitio:

- <https://github.com/EL-BID/pavimentados/>
- <https://code.iadb.org/en/tools/pavimentados>

LIBRERÍA EN PYTHON

Las versiones de python soportadas son python 3.10 y python 3.11.

Para instalar, se puede utilizar el siguiente comando:

```
pip install pavimentados
```

El siguiente paso es descargar el artefacto del modelo con el siguiente enlace:
https://github.com/EL-BID/pavimentados/raw/feature/v1.0.0/models/model_20240818.tar.gz?download=

Requisitos de sistema

No hay requisitos específicos de sistema para poder ejecutar el algoritmo, sin embargo, para que funcione de manera más eficiente se recomienda:

- Tarjeta de procesamiento grafico NVIDIA con compute capability superior a 5.3.
- Mas de 6GB de memoria RAM disponibles para ejecutar el sistema.
- Procesador de 8 núcleos.

QUICK START

En la carpeta de *notebooks* encontrará un ejemplo completo de cómo procesar tanto las imágenes como los vídeos presentes en *notebooks/road_videos* o *notebooks/road_images* y guardar los resultados en *notebooks/outputs*.

El primer paso es importar los componentes que crean un flujo de trabajo con imágenes:

```
from pavimentados.processing.processors import MultiImage_Processor
from pavimentados.processing.workflows import Workflow_Processor
```

En este ejemplo, tenemos el objeto procesador de imágenes *MultiImage_Processor*, que se encarga de tomar las imágenes y analizarlas individualmente utilizando los modelos. Además, tenemos el objeto *Workflow_Processor* que se encarga del flujo de trabajo de procesamiento de imágenes.

⁶ Para más información <https://www.gpsworld.com/what-exactly-is-gps-nmea-data/>

Manual de usuario de Pavimenta2

Internamente, el Workflow_Processor tiene objetos que pueden interpretar diferentes fuentes de imágenes o información GPS.

Entre las fuentes de imágenes permitidas están:

1. image_routes: lista de rutas de acceso a las imágenes
2. image_folder: carpeta con todas las imágenes
3. vídeo: ruta de acceso a un archivo de vídeo

Entre las fuentes de datos gps permitidas están:

1. image_routes: lista de las rutas de acceso de las imágenes que tienen datos GPS incrustados en ellas.
2. image_folder: carpeta con todas las imágenes que tienen datos gps incrustados en ellas.
3. loc: archivo en formato NMEA

Una vez que se importan estos elementos, el procesador se instancia de la siguiente manera:

```
from pathlib import Path
models_path = Path("./artifacts") # Path to downloaded model
ml_processor = MultiImage_Processor(artifacts_path=str(models_path))
```

Alternativamente, se puede especificar un archivo JSON adicional para la configuración o para sobrescribir algunos parámetros determinados de configuración en los modelos.

```
ml_processor = MultiImage_Processor(artifacts_path=str(models_path),
config_file="./models_config.json")
```

Los parámetros permiten especificar la confianza, el iou y la cantidad máxima de detecciones por fotograma.

Ejemplo de archivo de configuración:

```
{
  "signal_model": {
    "yolo_threshold": 0.20,
    "yolo_iou": 0.45,
    "yolo_max_detections": 100
  },
  "paviment_model": {
    "yolo_threshold": 0.20,
    "yolo_iou": 0.45,
    "yolo_max_detections": 100
  }
}
```

El objeto de flujo de trabajo es capaz de recibir el procesador instanciado, sin él no es capaz de ejecutar el flujo de trabajo.

Manual de usuario de Pavimenta2

```
input_video_file = "sample.mp4"
input_gps_file = "sample.log"

# Create a workflow for videos
workflow = Workflow_Processor(
    input_video_file, image_source_type="video", gps_source_type="loc",
    gps_input=input_gps_file, adjust_gps=True
)
```

El último paso consiste en ejecutar el flujo de trabajo y guardar los resultados en formato csv.

```
results = workflow.execute(ml_processor, video_output_file="processed_video.mp4",
video_from_results=True, batch_size=16)
```

```
# Save results to outputs directory
import pandas as pd
for result_name in results.keys():
    pd.DataFrame(results[result_name]).to_csv(f"{result_name}.csv")
```

En los resultados se encontrará lo siguiente:

1. table_summary_sections: DataFrame con tabla de resumen por secciones.
2. data_resulting: DataFrame con resultados por fotograma.
3. data_resulting_fails: DataFrame con resultados por fallas únicas encontradas.

FLUJOS DE TRABAJO

Los flujos de trabajo (workflows) dentro de la librería de Pavimenta2 son clases que permiten realizar trabajos enteros de procesamiento con una definición mínima de parámetros.

Para lograr realizar un procesamiento es necesario disponer principalmente de 3 elementos:

1. Modelo de Inteligencia Artificial, que se carga en el sistema a través de los objetos procesadores.
2. La data de visualización, que puede ser un video o conjunto de imágenes.
3. La data de geolocalización, la cual puede ser un archivo gps en csv o algún otro formato como nmea.

El flujo de trabajo simplemente provee un framework alrededor de estos elementos para facilitar la ejecución del procesamiento. Toma los datos, los transforma en objetos procesables y permite obtener los resultados.

El flujo de trabajo se utiliza usando los métodos provistos dentro de la librería:

```
from pavimentados.processing.workflows import Workflow_Processor
```


Manual de usuario de Pavimenta2

Al importarlo se tiene acceso a sus métodos y propiedades. Para generar uno se instancia como cualquier otro método de python y se definen los parámetros de entrada, que son:

- `Images_input` (obligatorio): Es el objeto de visualización que puede ser video, imágenes en formato array, una lista de las rutas de las imágenes o la carpeta donde están alojadas todas las imágenes.
- `Image_source_type` (default `image_folder`): dependiendo del tipo de entrada puede tomar cuatro valores:
 - `Image_routes`
 - `Images_folders`
 - `Video`
- `gps_source_type` (default `image_folder`): Como en el caso anterior puede tomar varios valores dependiendo del tipo de datos de geolocalización:
 - `loc`: si el archivo de localización es un archivo georeferenciado en formato `nmea`.
 - `images routes`: si la localización esta embebida en las imágenes.
 - `images folder`: si la localización esta embebida en las imágenes y todas están en la misma carpeta.
- `gps_input`: El objeto `gps` propiamente dicho, o la ruta de dicho objeto.
- `adjust_gps` (default `False`): permite realizar un ajuste a las ubicaciones en los casos que el archivo venga en formato `gps` o `loc`. EL parámetro debe ser boolean.
- `gps_section_distance` (default `100`): indica la longitud en metros de la longitud de los tramos que se considerara en los análisis.

En el caso de que el archivo `gps` sea `csv` se necesita añadir, además:

- `latitud_column`
- `longitud_columns`
- `time_columns`
- `date_column`
- `decimal_character`

Ejemplo de instancia de este objeto:

```
workflow = Workflow_Processor(route, image_source_type='image_folder',
gps_source_type = 'image_folder')
```

USO DE PROCESADORES

El siguiente paso de uso de este objeto es la ejecución. Para esto es necesario instanciar un procesador:

```
from pavimentados.processing.processors import MultiImage_Processor
from pathlib import Path
models_path = Path("./artifacts") # Path to downloaded model
ml_processor = MultiImage_Processor(artifacts_path=str(models_path))
```

Con el procesador instanciado se puede ejecutar el flujo de trabajo

```
results = workflow.execute(ml_processor,
                           batch_size=16,
```

```
video_output_file="processed_video.mp4"
)
```

De esta forma se pueden obtener los resultados los cuales se pueden acceder posteriormente usando la propiedad `get_results()`.

CREAR UNA APLICACIÓN

Para construir una aplicación utilizando la librería de Pavimenta2 simplemente se necesita instanciar todos los elementos mencionados anteriormente. Los flujos de trabajo permitirán realizar las ejecuciones de los datos obtenidos utilizando los procesadores.

El modelo cliente servidor es recomendado para las aplicaciones que utilizan esta librería, donde se construye un REST Api utilizando cualquier metodología conocida y se incorporan los elementos mencionados en este manual. Al hacer este proceso modular se evita que otros elementos interfieran con las ejecuciones de los flujos de trabajo o viceversa.

El API debe recibir toda la información necesaria para llevar a cabo una ejecución.

RESULTADOS

Los informes que se pueden generar a partir de los fotogramas y de los vídeos se dividen en tres tipos de informes.

Descripción de archivos:

- `table_summary_sections`: dataframe con resumen de datos por secciones.

Tabla 2: Output `table_summary_sections`

Nombre de columna	Tipo de dato	Descripción	Ejemplo
section	int	Identificador de la sección	1
D00	int	Distancia lineal en metros de este tipo de falla.	14
D10	int	Distancia lineal en metros de este tipo de falla.	42
D20	int	Distancia lineal en metros de este tipo de falla.	25
D40	int	Distancia lineal en metros de este tipo de falla.	8
D43	int	Distancia lineal en metros de este tipo de falla.	84
D44	int	Distancia lineal en metros de este tipo de falla.	2
latitude	float	Latitud inicial del tramo	-11.78369
longitude	float	Longitud inicial del tramo	-77.15555
end_longitude	float	Latitud final del tramo	-77.15626
end_latitude	float	Longitud final del tramo	-11.7830

section_distance	float	Distancia en metros de la sección	106.68306
-------------------------	-------	-----------------------------------	-----------

- *data_resulting*: dataframe con resultados por fotograma.

Tabla 3: Output data_resulting

Nombre de columna	Tipo de dato	Descripción	Ejemplo
latitude	float	Latitud de la detección (este dato sale del archivo georeferenciado).	-11.78366
longitud	float	Longitud de la detección (este dato sale del archivo georeferenciado).	-77.15560
distances	float	Distancia en metros del fotograma donde está la detección ⁷ .	7.653
ind	int	Índice auxiliar para trabajar la tabla.	0
fotograma	int	Número de fotograma dentro del video trabajado o número de imagen.	27
section	int	Número de tramo al cual pertenece la imagen sobre la que se hizo la detección ⁸ .	0
classes	string	Clase de la detección (este dato es un output del modelo).	D00
ind2	int	Índice/identificador secundario	0
scores	float	Probabilidad de la detección (este dato es un output del modelo)	0.22
boxes	List[float]	Coordenadas de la caja de la detección dentro de la imagen (este dato es un output del modelo).	[0.432063, 0.8396928, 0.65154, 0.99937]
class_id	string	Clase de la detección (este dato es un output del modelo).	D00
area	float	Área en pixeles de la caja de la detección (height x width).	72654.2
center	Tuple[float]	Centro en pixeles de la caja de la detección.	(1765.5051, 585.14807)
height	float	Altura en pixeles de la caja de la detección.	237.03959
width	float	Base en pixeles de la caja de la detección.	306.58985
total_area	float	Área total de la imagen en pixeles.	2073600
fail_id_section	int	Id de la falla a la que pertenece la detección ⁹ .	0

⁷ Es la distancia en metros entre fotogramas (fotograma n y n-1). La distancia se obtiene en función de las ubicaciones de los fotogramas (longitud, latitud). Como en los datos de entrada por segundo se tiene más de un dato de latitud y longitud, la ubicación del fotograma se establece aplicando una función de interpolación, de manera de obtener una única latitud y longitud por fotograma.

⁸ Se establece que las secciones serán de aproximadamente 100m. Basado en esto se generan agrupando fotogramas.

⁹ Una falla puede estar compuesta por más de una detección. Cuando se detecta la misma falla a lo largo de varios fotogramas consecutivos se considera una única falla.

- *data_resulting_fails*: dataframe con resultados por fallas únicas encontrados.

Tabla 4: Output data_resulting_fails

Nombre de columna	Tipo de dato	Descripción	Ejemplo
class_id	int	Identificador de la clase asignado a la detección.	0
classes	string	Clase de la detección (este dato es un output del modelo).	D00
fail_id_section	int	Id de la falla a la que pertenece la detección ¹⁰ .	43
distances	float	Distancia lineal en metros de la falla ¹¹ .	65.8743
start_coordinate	string	Coordenada inicial de la falla, con formato (latitud, longitud).	"(-11.7836697, -77.1556084)"
start_latitude	float	Latitud del punto de inicio de la falla.	-11.7836697
start_longitude	float	Longitud del punto de inicio de la falla.	-77.1556084
end_coordinate	string	Coordenada inicial de la falla, con formato (latitud, longitud).	"(-11.7830935, -77.15626067)"
end_latitude	float	Latitud del punto final de la falla.	-11.7830935,
end_longitude	float	Longitud del punto final de la falla.	-77.15626067
width	float	Base promedio en pixeles de la falla ¹² .	411
boxes	int	Cantidad de cajas usadas para construir la falla	19

ANÁLISIS DE RESULTADOS – CASOS DE USO

Pavimenta2 arroja el análisis por fotograma, segmento y por tipo de falla. En base a esto, es posible tomar los resultados y posteriormente diseñar reportes o dashboards de acuerdo con las necesidades de cada usuario¹³.

Aquí se presentan algunos ejemplos de uso:

- *Convertir los resultados al formato shapefile para crear visualizaciones en un sistema GIS o utilizando herramientas de código abierto.*

Los resultados obtenidos de Pavimenta2 permiten realizar un mapeo de las ocurrencias de defectos en las carreteras, facilitando la visualización y, en consecuencia, la planificación y priorización de trabajos de mantenimiento y mejoras. Por ejemplo,

¹⁰ Una falla puede estar compuesta por más de una detección. Cuando se detecta la misma falla a lo largo de varios fotogramas consecutivos se considera una única falla.

¹¹ La distancia se calcula como la suma de la distancia en metros de las detecciones que comprenden una falla (fail_id_section).

¹² Promedio en pixeles del ancho de las fallas agrupado según fail_section_id de data_resulting.

¹³ Estos análisis no son automáticamente generados por Pavimenta2. Deben ser desarrollados por los usuarios.

desde el BID, se aplicó Pavimenta2 exitosamente en la red de carreteras de Centroamérica, el Corredor del Pacífico, y con los resultados se desarrolló una herramienta de visualización.

Como resultado de dicho desarrollo, a continuación (Ilustración 9, Ilustración 10), se presentan tramos del Corredor coloreados según su estado. El color indica el porcentaje de fallas en cada tramo: los tramos con un bajo porcentaje de fallas aparecen en verde, mientras que aquellos con un alto porcentaje se muestran en rojo. Al hacer zoom en cada tramo, se puede observar con detalle la extensión del mismo y el porcentaje de fallas (Ilustración 11). En este caso, se ilustra la falla "Piel de Cocodrilo". Las imágenes fueron tomadas del Hub de Integración y Transporte del BID¹⁴.

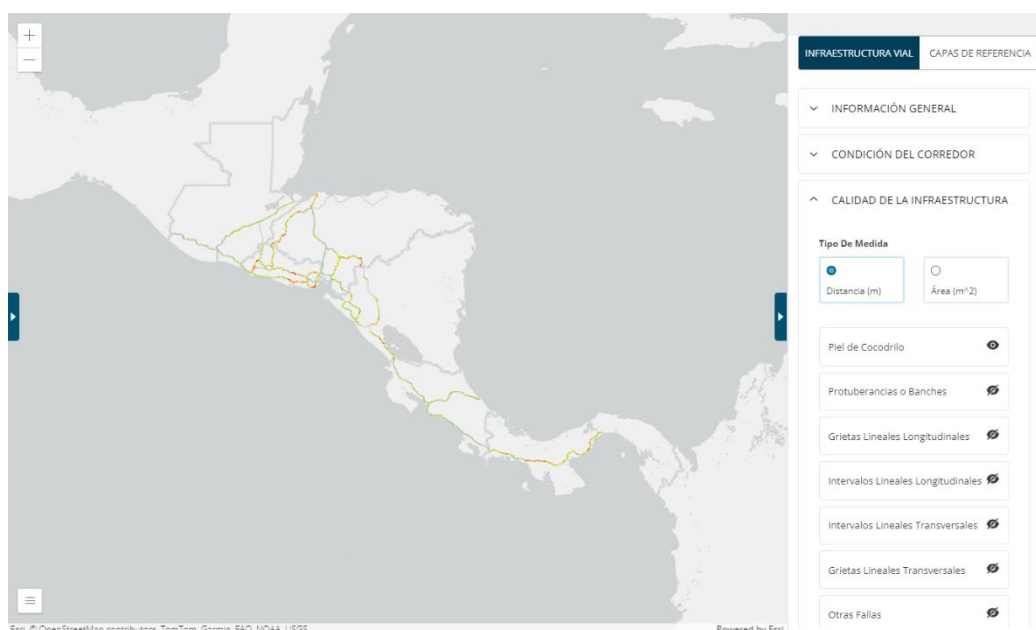


Ilustración 9: Ejemplo de visualización de resultados.

¹⁴ [IDB | Integration and Transport HUB](#)

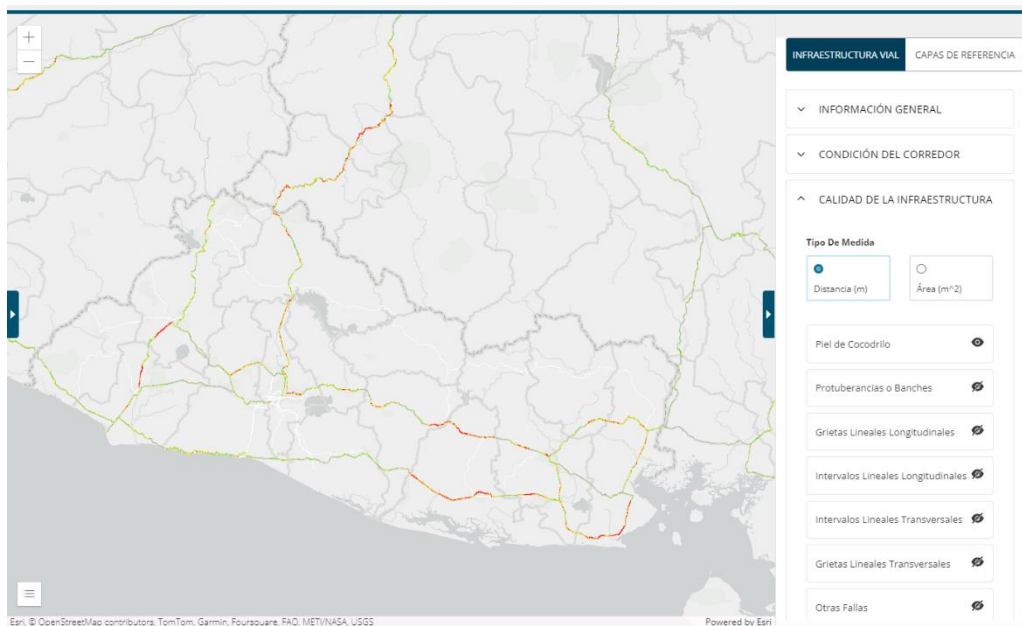


Ilustración 10: Ejemplo de visualización de resultados.



Ilustración 11: Ejemplo de visualización de resultados.

En la Ilustración 12: Mapa de calor, clasificación de defectos. Ilustración 12, se presenta otro ejemplo que muestra la distribución geográfica del volumen de fallas por zonas. El color verde, más cercano a 0, indica los segmentos con menor volumen de fallas, mientras que el rojo, más cercano a 1, representa las áreas con mayor volumen. El número que se muestra refleja, en términos porcentuales, la proporción de fallas en relación con el total de metros del segmento analizado.

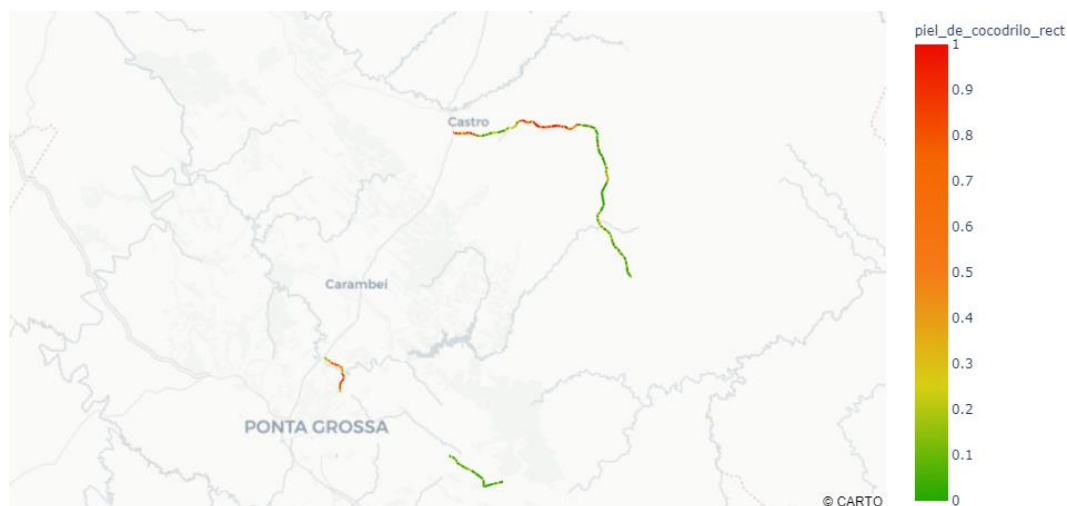


Ilustración 12: Mapa de calor, clasificación de defectos.

- Realizar equivalencias para calificar los segmentos de vía en bueno, regular y malo y elaborar matrices de soluciones.

Por ejemplo, se pueden considerar las siguientes directrices para la calificación del pavimento.

1. Pavimento en buen estado: sin defectos aparentes (menos de 20% del tramo tiene fallas).
2. Pavimento en estado regular: con baja incidencia de grietas (menos de 50% del tramo tiene fallas).
3. Pavimento en mal estado: con incidencia media a alta de grietas (50% del tramo o más tiene fallas).

En función de esto, se puede determinar la extensión de carretera en bueno, regular y mal estado.

Tabla 5: Ejemplo calificación de segmentos

Estado General	Extensión	
	km	%
Bueno	28	18.7
Regular	57	38
Malo	65	43.3
Total	150	100

Dependiendo del tipo de defecto se pueden predecir soluciones específicas para la mejora de la carretera a través de los datos de salida de Pavimenta2. En este sentido se puede elaborar una matriz de solución que debe definirse para segmentos de comportamiento homogéneo del pavimento.

Un segmento de comportamiento homogéneo se define como cada fracción de la extensión de estiramiento que tiene la misma constitución de estructura y parámetros razonablemente constantes de defectos superficiales e irregularidades.

La Matriz de Soluciones incluye tipologías del defecto y la solución sugerida, como ejemplo a continuación:

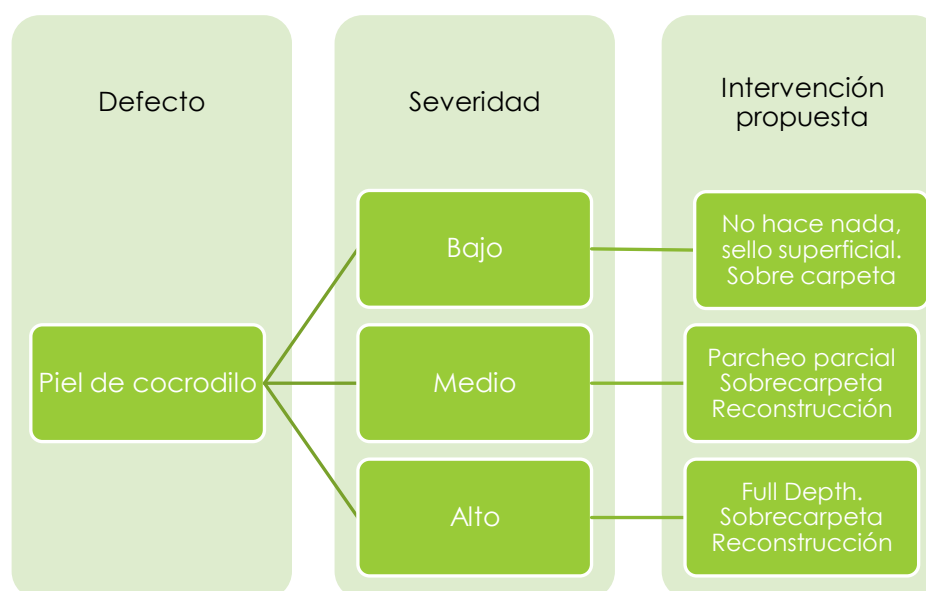


Ilustración 13: ejemplo matriz de soluciones

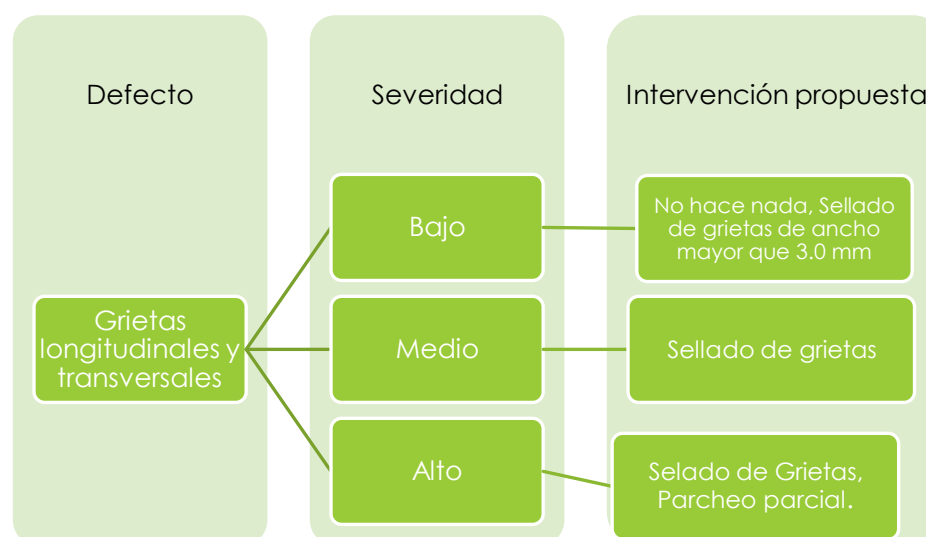


Ilustración 14: ejemplo matriz de soluciones

INFORMACIÓN DE CONTACTO

Para obtener más información sobre el uso de la herramienta, puede enviar un correo a infradigital@iadb.org.

LICENCIA DE SOFTWARE

Manual de usuario de Pavimenta2

El Software, con excepción de los Documentos de Soporte y Uso, deberá estar sujetos a los siguientes términos y condiciones, los cuales se empaquetarán como parte de cada Software bajo un archivo que llevará el nombre de "LICENCIA": "Copyright © [año]. Banco Interamericano de Desarrollo ("BID"). Uso autorizado."

LICENCIA

Por medio de la presente licencia ("Licencia") no exclusiva, revocable, global y libre de regalías el BID otorga permiso al usuario ("Usuario") para reproducir, distribuir, ejecutar públicamente, prestar y poner a disposición del público, y modificar el software y sus modificaciones, por sí o como parte de una colección, siempre que sea para fines no comerciales, sujeto a los términos y condiciones aquí señalados. Salvo indicación en contrario, la Documentación de Soporte y Uso del software se licenciará bajo Creative Commons IGO 3.0 Atribución-NoComercial-SinObraDerivada (CC-IGO 3.0 BY-NC-ND).

DEFINICIONES

Software: conjunto de programas, instrucciones y reglas informáticas que permiten ejecutar tareas en una computadora a fin de gestionar un determinado proceso u obtener un determinado resultado. El Software incluye (i) código fuente, (ii) código objeto y (iii) Documentación de Soporte y Uso.

El Software puede ser del tipo, pero sin estar limitado a (i) programa ejecutable desde el computador, (ii) aplicación móvil y de escritorio, (iii) algoritmo y (iv) hoja de cálculo que contienen macros, así como otras instrucciones para simular, proyectar o realizar otros cálculos.

Código Fuente: conjunto de líneas de texto con los pasos que debe seguir la computadora para ejecutar el Software. Puede estar compuesto por un número indeterminado de documentos, con distintas extensiones y organizados en carpetas.

Código Objeto: código que resulta de la compilación del Código Fuente. La compilación es un proceso informático que traduce el Código Fuente en lenguaje máquina o bytecode.

Documentación de Soporte y Uso (la "Documentación"): se refiere a la información de acompañamiento al código fuente y objeto que permite a un humano comprender los objetivos, arquitectura y lenguaje de programación de Software.

Software Derivado: obras derivadas del Software, tales como modificaciones, actualizaciones y mejoras de esta.

Usuario: cualquier persona, natural o jurídica, que obtenga una copia del Software.

DERECHOS DEL BID

El BID se reserva todos los derechos de propiedad intelectual, incluyendo sin limitación derechos de autor, relacionados con o asociados al Software.

El BID se reserva expresamente los derechos de ceder, transferir y/o sub-licenciar el Software y/o cualquiera de sus componentes a terceros sin previo aviso.

DERECHOS DEL USUARIO SOBRE SOFTWARE DERIVADO

El Usuario que desarrolle Software Derivado retendrá todos los derechos de propiedad intelectual, incluyendo sin limitación derechos de autor, relacionados con o asociados al Software Derivado, en el entendido que dicho Software Derivado y cualquier otra edición futura estén sujetas a los mismos términos y condiciones de esta Licencia.

RESTRICCIONES

- El Usuario sólo está autorizado a hacer uso del Software conforme se describe en esta Licencia.
- Con excepción de esta Licencia, el BID no otorga ninguna otra licencia al Usuario con respecto a cualquier otra información u obra propiedad del BID o de cualquier derecho de autor, marcas o cualquier otro derecho de propiedad intelectual del BID. Cualquier licencia adicional deberá ser por escrito y firmada por un representante del BID debidamente autorizado para tales fines.
- La Licencia no constituye una venta del Software ni de cualquier parte o copia de esta.
- El Usuario no podrá usar ni permitir que otros usen el Software para fines comerciales.
- El Usuario sólo podrá autorizar el uso del Software a terceros de conformidad con lo establecido en esta Licencia, sin que en ningún caso los terceros puedan adquirir más derechos sobre el Software que los expresamente otorgados por el BID mediante esta Licencia.

RECONOCIMIENTO

El Usuario deberá mantener los siguientes enunciados y exenciones en el archivo principal de la Documentación del Software: "Copyright © [año]. Banco Interamericano de Desarrollo ("BID"). Uso autorizado.

Los procedimientos y resultados obtenidos en base a la ejecución de este software son los programados por los desarrolladores y no necesariamente reflejan el punto de vista del BID, de su Directorio Ejecutivo ni de los países que representa."

En el caso de Software del Fondo Multilateral de Inversiones ("FOMIN"), la exención de responsabilidad deberá ser: "Las opiniones expresadas en esta obra son de los autores y no necesariamente reflejan el punto de vista del BID, de su Directorio Ejecutivo ni de los países que representa, así como tampoco del Comité de Donantes del FOMIN ni de los países que representa."

El Usuario podrá incluir en el Software Derivado una referencia al Software como obra original, siempre y cuando dicho aviso no genere confusión alguna respecto a la titularidad de los derechos del BID sobre el Software.

NOMBRE Y LOGO DEL BID

El Usuario no podrá hacer uso del nombre y/o logo del BID para fines diferentes a los aquí estipulados, ni podrá hacer promesas, ni adquirir compromisos u obligaciones, ni otorgar garantías de ninguna clase en nombre del BID.