# COMPARATIVA DE TÉCNICAS DE PROCESAMIENTO DE IMÁGENES PARA LA DETECCIÓN DE PEATONES CON SVM

Álvaro Ruiz Gutiérrez, Lidia Jiménez Soriano, Víctor Flores González, Óscar Menéndez Márquez

8 de marzo de 2025

**Resumen**

En este trabajo se analiza y compara la eficiencia de cuatro métodos de extracción de características en imágenes: Harris Corner Detection, SURF, SIFT y HOG, aplicados a la detección de peatones mediante un modelo SVM preentrenado. Se estudia la teoría detrás de cada método, extrayendo y explicando sus principios fundamentales de manera didáctica. Posteriormente, se implementan o adaptan implementaciones existentes, detallando los parámetros utilizados y los efectos que tienen en el rendimiento del modelo. Para evaluar la eficiencia de cada método, se definen métricas clave como la precisión y tiempo de procesamiento. Se presentan los resultados obtenidos mediante experimentación con un conjunto de datos de imágenes, analizando el impacto de cada técnica en la identificación de peatones. Finalmente, se expone de manera clara la comparativa de los métodos estudiados, destacando sus ventajas y desventajas en función de los resultados obtenidos.

**Palabras clave:** Harris Corner Detection,SURF , SIFT, HOG, procesamiento de imágenes, detección de peatones, SVM.

## 1. Introducción

El procesamiento de imágenes digitales es un campo clave dentro de la visión por computador, con aplicaciones en reconocimiento de patrones, inteligencia artificial y análisis de imágenes. Dentro de este ámbito, la detección de objetos y, en particular, la identificación de peatones, representa un desafío fundamental con implicaciones en seguridad, movilidad y automatización.

Este trabajo se centra en el estudio y comparación de cuatro técnicas de extracción de características utilizadas en la identificación de peatones: **Harris Corner Detection**, **SURF (Speeded Up Robust Features)**, **SIFT (Scale-Invariant Feature Transform)** y **HOG (Histogram of Oriented Gradients)**. Estas técnicas permiten identificar estructuras relevantes en una imagen, facilitando su uso en tareas de reconocimiento y clasificación. Para evaluar su rendimiento, se utilizará un **modelo SVM (Support Vector Machine) preentrenado**, el cual procesará los descriptores generados por cada método para realizar la identificación de peatones en imágenes.

El **objetivo** principal de este trabajo es ofrecer una explicación didáctica de los métodos estudiados, tanto desde un punto de vista teórico como práctico. Para ello, el desarrollo del proyecto se estructura en tres fases clave:

- **Análisis teórico:** Se investigarán los principios fundamentales de cada método, extrayendo las ideas principales de fuentes científicas.
- **Implementación práctica:** Se desarrollará o adaptará la implementación existente de cada técnica, detallando los pasos intermedios, el significado de los parámetros y la experimentación con distintas configuraciones.
- **Presentación didáctica:** Se explicarán de forma estructurada todos los pasos necesarios para la detección de peatones utilizando cada método, complementando con ejemplos gráficos y análisis de resultados.

Dado que el desarrollo del trabajo está orientado a la investigación y aplicación de técnicas avanzadas no vistas en clase, se apoyará en artículos científicos recientes. La **planificación** y ejecución del proyecto se realizará en un equipo de trabajo compuesto por 4 alumnos ,distribuyendo **70 horas por persona**. Además, se utilizarán herramientas de gestión de proyectos para organizar y documentar las tareas realizadas, asegurando un seguimiento adecuado de los avances. Concretamente, se utilizará GitHub como repositorio del código fuente y documentación, Clockify para el registro y seguimiento de horas y tareas, y Microsoft Project para realizar una planificación previa precisa de las fases, hitos y actividades previstas.

# 2. Planteamiento teórico

El problema abordado se centra en la detección automatizada de peatones a partir de imágenes digitales mediante técnicas de procesamiento de imágenes como **Harris**, **SURF**, **SIFT** y **HOG**. Esta tarea es fundamental en múltiples aplicaciones, como la seguridad vial, la monitorización del tráfico y la visión por computadora en sistemas autónomos. La detección precisa y eficiente de peatones permite mejorar la toma de decisiones en entornos urbanos, reducir accidentes o incluso optimizar sistemas de asistencia a la conducción.

Para abordar este problema, se han implementado y evaluado los algoritmos mencionados, los cuales permiten extraer características distintivas de las imágenes con el objetivo de identificar patrones asociados a la presencia de peatones. En esta sección, se describirá en profundidad el funcionamiento de cada uno de estos algoritmos, analizando sus fundamentos teóricos, principios matemáticos y ventajas frente a otros enfoques. Esta base teórica servirá como punto de partida para su posterior implementación y evaluación experimental.

## 2.1. Harris Corner Detection

## 2.2. SURF (Speeded-Up Robust Features)

## 2.3. SIFT (Scale-Invariant Feature Transform)

El algoritmo **SIFT** (Scale-Invariant Feature Transform) es una técnica ampliamente utilizada en visión por computadora para la detección y descripción de características en imágenes. Fue desarrollado por *David Lowe* en 1999 y destaca por su capacidad de identificar puntos clave en una imagen de manera robusta frente a transformaciones como cambios de escala, rotación e iluminación. Gracias a estas propiedades, SIFT se ha convertido en una herramienta fundamental para aplicaciones como el reconocimiento de objetos, la detección de patrones y el emparejamiento de imágenes.

#### Fundamentos teóricos

El algoritmo SIFT se basa en la extracción de puntos característicos (keypoints) de una imagen y la generación de descriptores asociados a ellos. Este proceso se realiza en varias etapas, las cuales garantizan la invariancia a escala y a rotación.

- **Construcción de la Pirámide Gaussiana:** Se aplican sucesivas convoluciones con un filtro gaussiano para generar versiones suavizadas de la imagen en diferentes escalas. Este proceso permite analizar la imagen a distintos niveles de detalle, lo que hace que los puntos característicos sean robustos a cambios de tamaño y desenfoque.
- **Construcción de la Pirámide de Diferencia de Gaussianos (DoG):** Se calculan diferencias entre imágenes suavizadas consecutivas de la pirámide gaussiana. Esta operación permite resaltar bordes y estructuras significativas en la imagen, facilitando la detección de puntos clave.

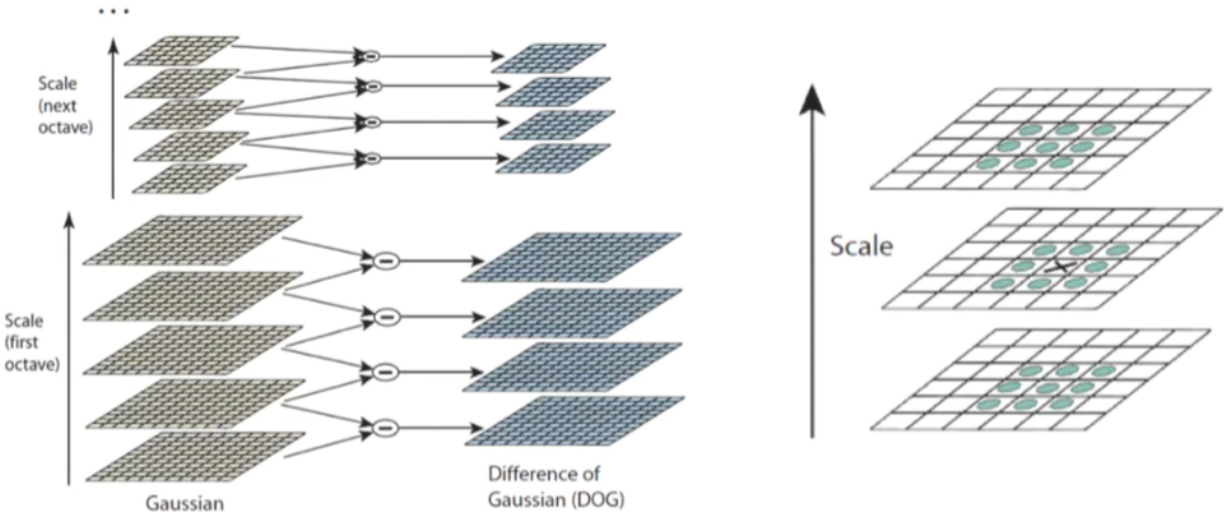


Figura 1: Proceso de la piramide de diferencia gaussiana.

- **Detección de keypoints:** Se identifican los puntos extremos en la pirámide DoG, comparando cada píxel con sus vecinos en los niveles superior e inferior de la pirámide, así como en su entorno local. Los puntos seleccionados se someten a un proceso de refinamiento para descartar aquellos con bajo contraste o situados en bordes poco definidos.
- **Asignación de orientación:** A cada keypoint se le asigna una orientación dominante basada en la distribución de gradientes en su vecindad. Para ello, se calcula el gradiente horizontal y vertical, y se calcula el histograma de orientaciones del gradiente en una ventana centrada en el punto en la escala en la que se ha detectado. La orientación se discretiza en 36 intervalos y se van acumulando las amplitudes del gradiente en cada ángulo detectado. Esto permite que el algoritmo sea invariante a rotaciones.

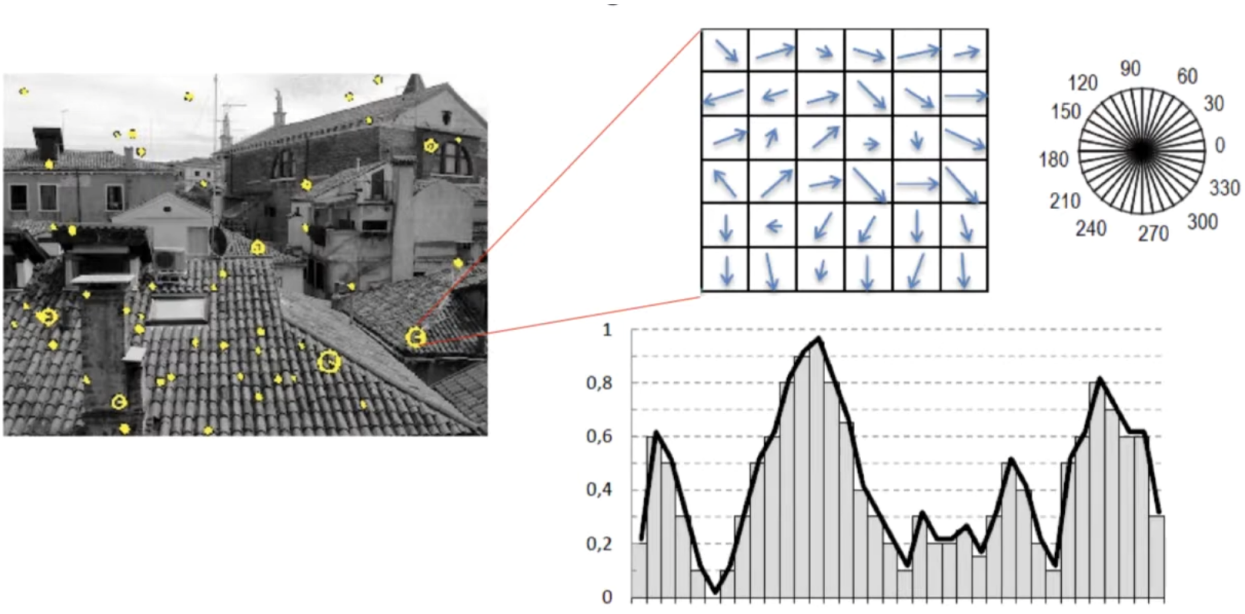


Figura 2: Proceso de orientación de un keypoint.

- **Generación del descriptor SIFT:** Para representar de manera robusta un keypoint, se genera un descriptor basado en la distribución de los gradientes dentro de una ventana centrada en dicho punto. Esta región se divide en una cuadrícula de $4 \times 4$ celdas, formando

un total de 16 subregiones. En cada subregión se calcula un histograma de orientaciones de gradientes con 8 direcciones posibles. Finalmente, los 16 histogramas se concatenan en un único vector de características de 128 dimensiones, dado que proviene de la combinación de las 16 subregiones ($4 \times 4$) y las 8 direcciones ($16 \times 8 = 128$). Este descriptor permite identificar y comparar keypoints de manera robusta frente a variaciones en escala, rotación e iluminación.

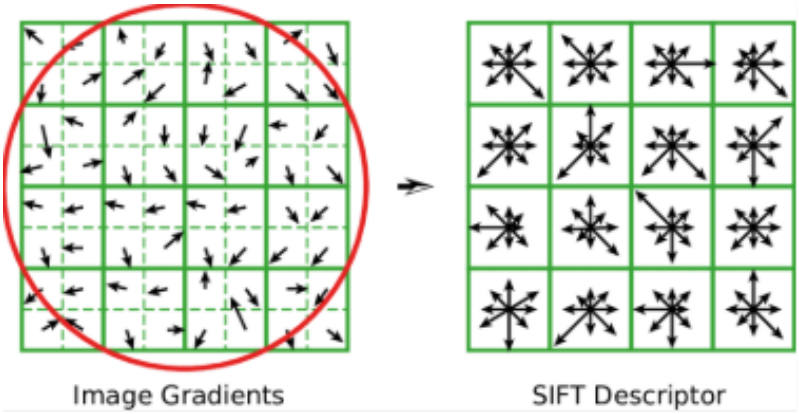


Figura 3: Keypoint Descriptor.

**Fundamentos matemáticos**

El algoritmo SIFT se basa en varios principios matemáticos que garantizan su eficacia en la detección y descripción de características. A continuación, se presentan algunos de los conceptos clave:

- **Filtro Gaussiano:** La función gaussiana en dos dimensiones está definida por:

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$

  donde $\sigma$ controla el nivel de suavizado aplicado a la imagen.

- **Diferencia de Gaussianos (DoG):** Se define como la resta entre dos imágenes suavizadas con diferentes valores de $\sigma$:

$$DoG(x, y, \sigma) = G(x, y, \sigma_2) - G(x, y, \sigma_1)$$

  Esta operación resalta los cambios de intensidad y permite la detección de keypoints.

- **Cálculo del gradiente:** Para determinar la orientación de un keypoint, se calcula el gradiente de la imagen en cada píxel utilizando:

$$m(x, y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2}$$

$$\theta(x, y) = \tan^{-1}\left(\frac{L(x, y+1) - L(x, y-1)}{L(x+1, y) - L(x-1, y)}\right)$$

  donde $m(x, y)$ representa la magnitud del gradiente y $\theta(x, y)$ su dirección.

- **Generación del descriptor SIFT:** El descriptor de cada keypoint como se explicó anteriormente, se forma a partir de histogramas de gradientes calculados en una ventana de $16 \times 16$ píxeles alrededor del punto de interés. La ventana se subdivide en $4 \times 4$ celdas y en cada celda se calcula un histograma de 8 orientaciones, generando un vector de:

$$4 \times 4 \times 8 = 128$$

  dimensiones, que representa la información local de la imagen.

#### Ventajas de SIFT

El algoritmo SIFT presenta varias ventajas en comparación con otros métodos de detección de características:

- **Invariancia a escala y rotación:** Gracias a la construcción de la pirámide gaussiana y la asignación de orientación, los keypoints detectados son robustos a cambios de tamaño y rotación en la imagen.
- **Alta discriminación y robustez:** Los descriptores SIFT tienen una alta capacidad para diferenciar objetos en diferentes condiciones de iluminación y perspectiva.
- **Resistencia a ruido y oclusiones parciales:** Debido a su método de detección basado en diferencias de gaussianos y análisis de gradientes, SIFT es menos sensible al ruido y sigue funcionando incluso cuando partes de la imagen están ocultas.
- **Aplicabilidad en reconocimiento de patrones y emparejamiento de imágenes:** El uso de descriptores de 128 dimensiones permite realizar comparaciones eficientes entre imágenes, facilitando tareas como el reconocimiento de objetos.

## 2.4. HOG (Histogram of Oriented Gradients)

# 3. Implementación

## 3.1. Tecnologías

Para este proyecto se ha utilizado Python como lenguaje de programación junto con varias librerías especializadas en procesamiento de imágenes, aprendizaje automático y visualización de datos. A continuación, se detallan las principales tecnologías empleadas:

- **OpenCV** (web oficial): Biblioteca de código abierto para visión por computadora y procesamiento de imágenes. Se utilizó para la lectura, preprocesamiento y transformaciones de las imágenes.
- **Numpy** (web oficial): Biblioteca especializada en cálculo numérico y gestión de matrices multidimensionales. Se usó para la manipulación de datos y cálculos matemáticos en la transformación de imágenes.
- **Matplotlib** (web oficial): Biblioteca de visualización de datos. Se empleó principalmente para representar imágenes, mostrar los puntos de interés detectados y visualizar los resultados obtenidos.
- **Scikit-image** (web oficial): Biblioteca de procesamiento de imágenes que proporciona herramientas adicionales para realizar operaciones de filtrado, detección de bordes y transformaciones de imágenes.
- **Scikit-learn** (web oficial): Biblioteca de aprendizaje automático utilizada para entrenar y evaluar el modelo SVM que predice la presencia de peatones en las imágenes procesadas.

## 3.2. Técnicas y Algoritmos Empleados

A lo largo del proyecto se implementaron varias técnicas de procesamiento de imágenes y detección de características, con el objetivo de extraer información relevante para alimentar un modelo de clasificación basado en **SVM (Support Vector Machine)**. A continuación, se detallan los algoritmos utilizados y su tratamiento sobre las imágenes:

### 3.2.1. Harris Corner Detection

**Proceso aplicado:**

### 3.2.2. SURF (Speeded-Up Robust Features)

**Proceso aplicado:**

### 3.2.3. SIFT (Scale-Invariant Feature Transform)

El algoritmo **SIFT** es un método robusto para detectar características invariantes a escala y rotación en una imagen. Fue utilizado para extraer puntos de interés y construir descriptores característicos que permitieran la identificación de patrones en las imágenes.

### 3.2.4. Proceso de Aplicación de SIFT

A continuación, se detallan los pasos seguidos en la implementación del algoritmo SIFT:

**Paso 1: Cargar y Mostrar la Imagen**

Antes de aplicar el algoritmo, se importan las librerías necesarias y se carga la imagen con la que se va a trabajar.


Figura 4: Imagen original cargada.

**Paso 2: Conversión de la Imagen a Escala de Grises**

La imagen se convierte a escala de grises, ya que SIFT opera sobre la intensidad de los píxeles en lugar de los colores, lo que mejora la estabilidad de la detección de características.


Figura 5: Conversión a escala de grises.

**Paso 3: Configuración del Detector SIFT**

El detector **SIFT** requiere la configuración de varios parámetros clave que afectan la detección

de características en la imagen:

- **nfeatures = 5:** Número máximo de características a detectar. Si se establece en 0, se detectan todas las posibles características.
- **nOctaveLayers = 3:** Número de capas por octava en la pirámide Gaussiana.
- **contrastThreshold = 0.04:** Umbral para descartar características de bajo contraste.
- **edgeThreshold = 10:** Umbral para eliminar puntos en bordes poco definidos.
- **sigma = 1.6:** Desviación estándar inicial para el filtro Gaussiano.

Para la implementación, se ha utilizado la clase correspondiente en el módulo `algoritmos/sift.py`.

En las pruebas iniciales, se estableció `nfeatures` en **5** para reducir la cantidad de puntos clave detectados, facilitando la visualización y comprensión teórica del proceso.

**Paso 4: Construcción de la Pirámide Gaussiana**

La pirámide Gaussiana se genera aplicando convoluciones sucesivas con filtros gausianos de distinta desviación estándar ($\sigma$), reduciendo la resolución de la imagen en cada octava. Este proceso permite analizar la imagen a diferentes escalas para detectar características invariantes.


Figura 6: Construcción de la Pirámide Gaussiana.

**Paso 5: Construcción de la Pirámide de Diferencia de Gaussianos (DoG)**
Después de generar la pirámide Gaussiana, se construye la pirámide de Diferencia de Gaussianos (DoG) restando imágenes consecutivas de la pirámide. Esto resalta las variaciones de intensidad entre niveles de suavizado, facilitando la detección de puntos de interés.

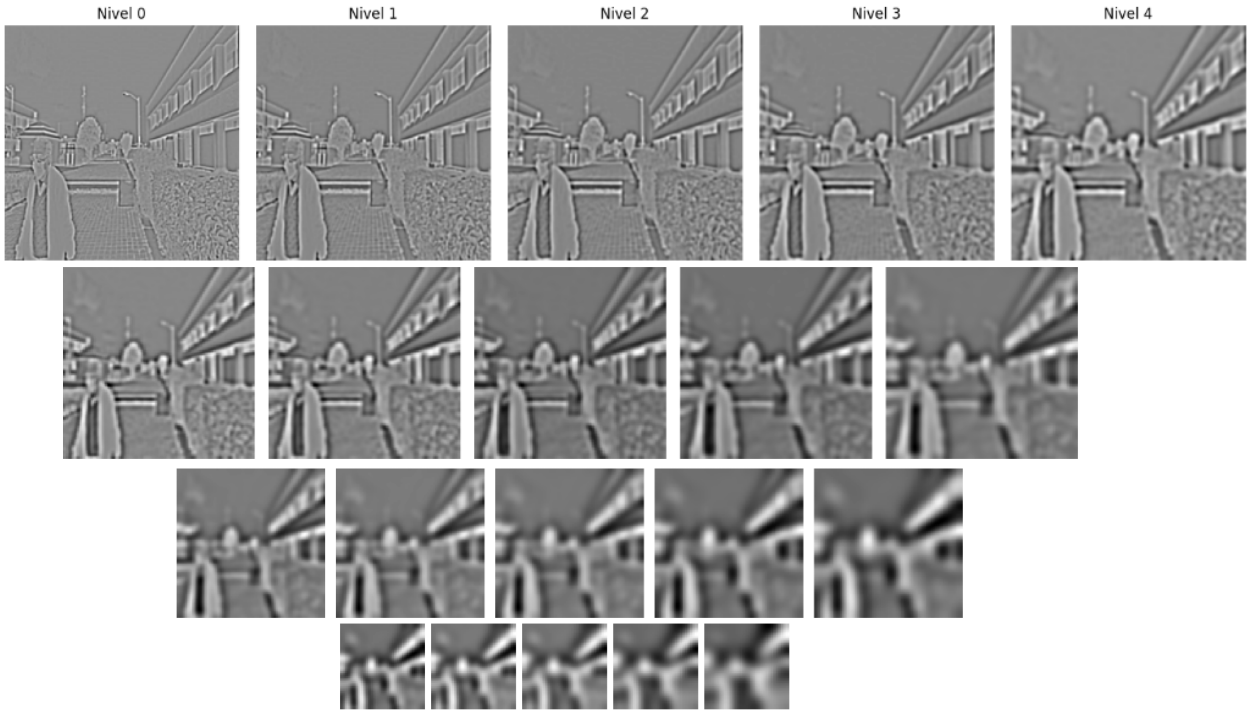


Figura 7: Construcción de la Pirámide de Diferencia de Gaussianos (DoG).

**Paso 6: Detección de Keypoints**

Los keypoints se detectan identificando los extremos locales en la pirámide DoG. Se comparan píxeles con sus vecinos en los niveles superiores e inferiores de la pirámide para encontrar puntos máximos y mínimos. Como resultado, obtenemos 5 keypoints debido a la configuración propuesta anteriormente.

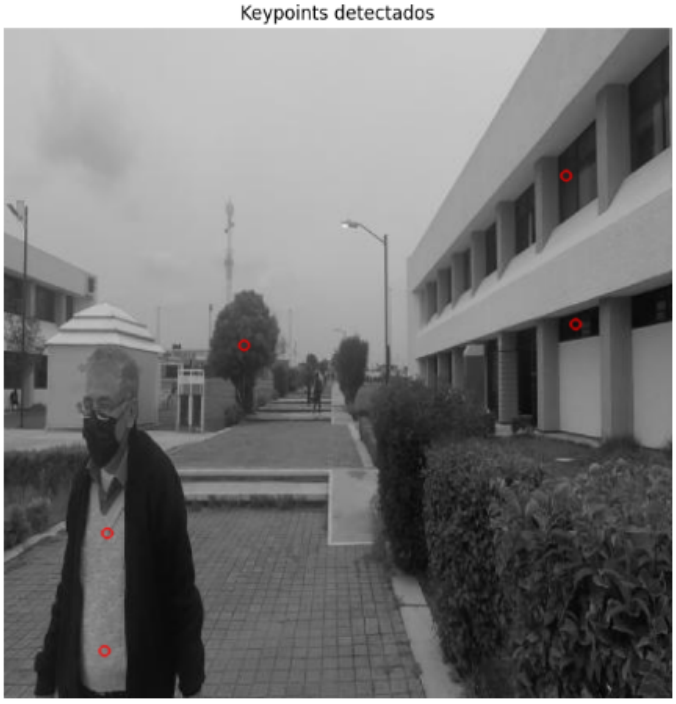


Figura 8: Detección de 5 keypoints en la imagen.

En el caso de aplicar el valor 0 al parámetro **nfeatures**, se detectarían todas las características

posibles en la imagen, lo que aumentaría significativamente el número de keypoints detectados, como se observa en la siguiente imagen:


Figura 9: Detección de todos los keypoints con `nfeatures` = 0.

**Paso 7: Asignación de Orientación a los Keypoints**

Cada keypoint recibe una orientación basada en los gradientes locales de la imagen. Esto permite que los descriptores sean invariables a la rotación.

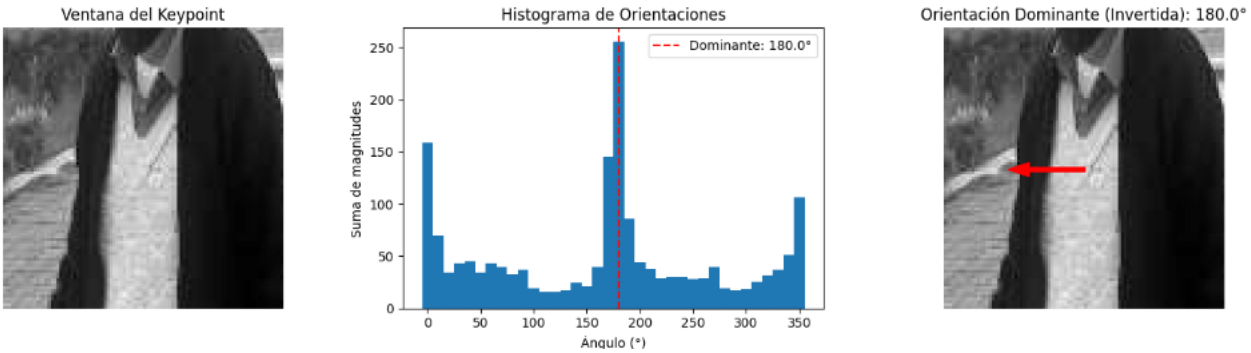


Figura 10: Orientación asignada a cada keypoint.

**Paso 8: Visualización de los Keypoints con su Orientación**

Los keypoints detectados y sus orientaciones asignadas se visualizan para evaluar la robustez del algoritmo.

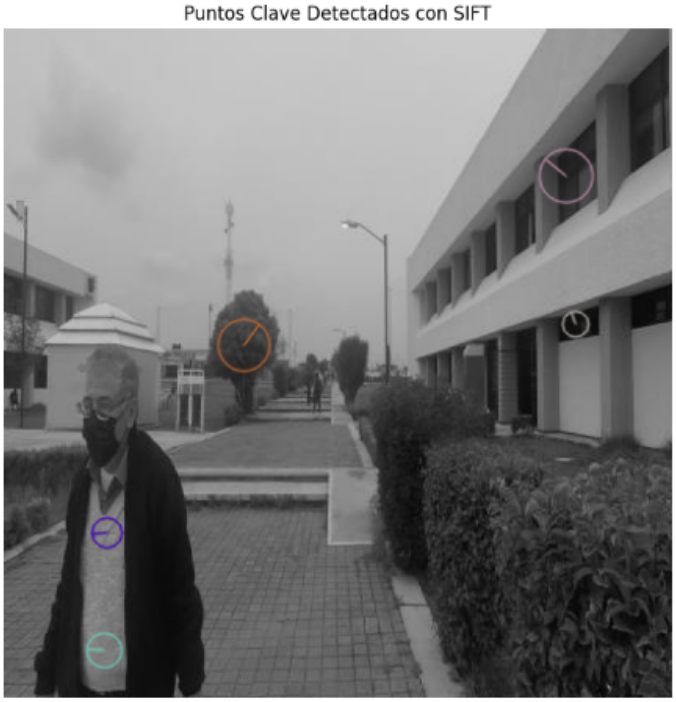


Figura 11: Visualización de keypoints con orientación.

**Paso 9: Cálculo de Descriptores**

Cada keypoint se describe mediante un vector de características basado en la distribución de gradientes en su vecindad.

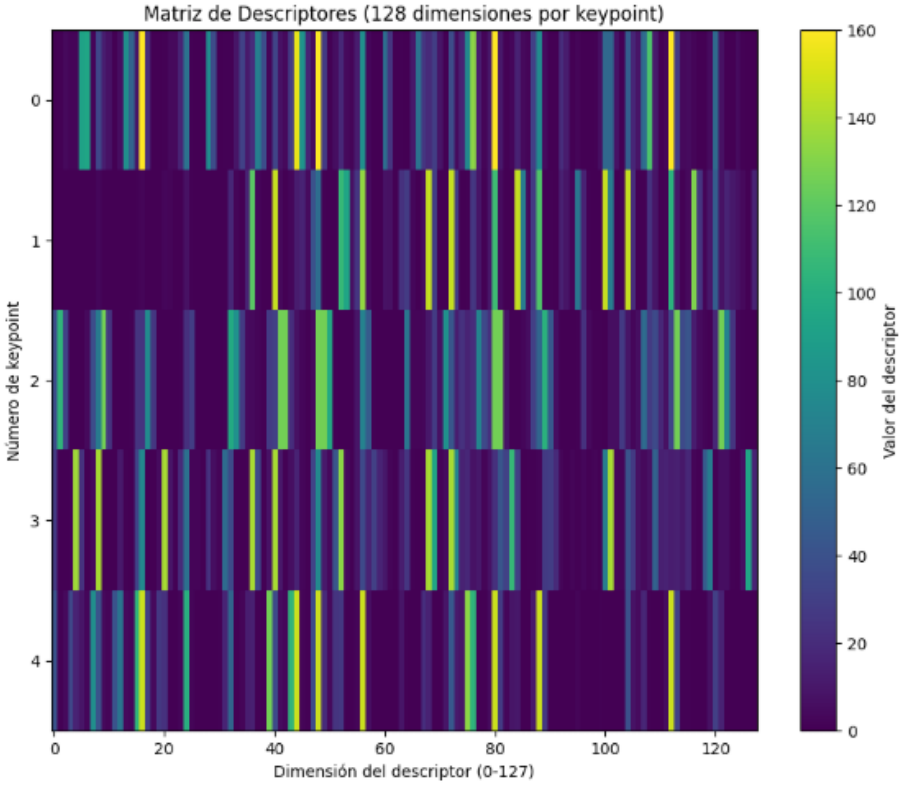


Figura 12: Ejemplo de descriptor SIFT calculado.

Estos descriptores serán los que permitan al modelo de aprendizaje automático emparejar características entre diferentes imágenes, facilitando tareas como el reconocimiento de objetos y la detección de peatones en escenas complejas.

#### 3.2.5. HOG (Histogram of Oriented Gradients)

**Proceso aplicado:**

## 3.3. Decisiones de Diseño

- **Separación de los algoritmos en archivos independientes:** Cada método de detección (Harris, SURF, SIFT y HOG) se implementó en archivos de Python independientes dentro de una carpeta central denominada `algoritmos`.
- **Presentación didáctica en Jupyter Notebooks:** Se desarrolló un `notebook.ipynb` por cada algoritmo donde se explican los algoritmos paso a paso.
- **Experimentación con un banco de imágenes de prueba:** Se realizó una prueba con un conjunto de imágenes para entrenar un modelo **SVM** y predecir si en una imagen hay o no un peatón.
- **Estructura organizada del proyecto:**
  - Se creó una carpeta `images` donde se almacenan los bancos de imágenes de entrenamiento del **SVM** y las pruebas para cada algoritmo. Las imágenes incluyen ejemplos con y sin personas, extraídas de `roboflow.com`.
  - Se añadió una carpeta `doc` para documentar el desarrollo del proyecto y actualizarlo progresivamente.
  - Se estableció una carpeta `machinelearning` dedicada específicamente al entrenamiento del modelo **SVM**.
  - Los Jupyter Notebooks están ubicados en la raíz del proyecto para facilitar su acceso y ejecución.

# 4. Experimentación

La experimentación es una parte fundamental de este trabajo, ya que permite evaluar el rendimiento de los algoritmos implementados en combinación con el modelo **SVM** (Support Vector Machine). Para obtener resultados significativos, hemos realizado pruebas sistemáticas con las configuraciones más eficientes con cada algoritmo y un conjunto variado de imágenes.

Una vez implementados y probados los algoritmos de detección de características (**Harris, SURF, SIFT y HOG**), decidimos entrenar un modelo **SVM** utilizando un banco de datos con más de 1000 imágenes, en las que se incluyen ejemplos con y sin peatones. El objetivo es evaluar la capacidad del modelo para predecir correctamente la presencia de un peatón en una imagen.

## 4.1. Preparación del Conjunto de Datos

Para garantizar un entrenamiento adecuado del **SVM**, cada imagen del banco de datos fue sometida a un tratamiento previo, el cual depende del algoritmo de detección de características a evaluar. Este preprocesamiento incluyó:

- Conversión a escala de grises (en los algoritmos que lo requieren).
- Aplicación del método de detección de características correspondiente (Harris, SURF, SIFT o HOG).
- Extracción de descriptores característicos de cada imagen.
- Normalización de los descriptores para mejorar la robustez del modelo.

## 4.2. Entrenamiento del Modelo SVM

Tras procesar el conjunto de datos, se procedió al entrenamiento del **SVM** con los descriptores extraídos de cada imagen. Para ello, se utilizó un conjunto de entrenamiento compuesto por imágenes etiquetadas en las que tomaba el valor true en caso de haber una o más personas (señalando su ubicación), y false en caso contrario, para diferenciar aquellas que contienen peatones de aquellas que no.

## 4.3. Pruebas de Evaluación

Una vez entrenado el modelo con los descriptores obtenidos de cada algoritmo, se realizó una prueba final con un conjunto de imágenes con y sin personas, que no habían sido utilizadas durante el entrenamiento. El objetivo era medir la eficiencia del modelo en la tarea de detección de peatones.

Cada imagen de prueba fue procesada de la misma manera que las imágenes de entrenamiento, aplicando el algoritmo de detección de características correspondiente y extrayendo sus descriptores. Posteriormente, estos descriptores fueron introducidos en el **SVM** entrenado para predecir si la imagen contenía un peatón o no.

## 4.4. Resultados y Análisis

Para evaluar el desempeño del modelo, se calcularon métricas como:

- **Precisión**: Porcentaje de imágenes correctamente clasificadas.
- **Tiempo de Procesamiento**: Tiempo medio necesario para procesar una imagen.

## 4.5. Resultados Obtenidos

# 5. Manual de usuario

El manual de usuario del proyecto se encuentra disponible en el archivo `README.md` dentro del repositorio del proyecto en GitHub. Dicho manual proporciona una guía detallada sobre el uso de la aplicación, incluyendo su instalación, configuración y funcionalidades principales.

Para acceder al manual de usuario, visite el siguiente enlace donde encontrará el repositorio del proyecto en GitHub:

https://github.com/Lidiajim/AnalisisDeAlgoritmosPID

# 6. Conclusiones

Se debe introducir una sección de conclusiones que incluyan propuestas claras de mejora o extensión del trabajo (por ejemplo, si no se han podido alcanzar todos los objetivos iniciales). También conclusiones sobre los resultados obtenidos, en qué medida difieren de los esperados. además, son apropiadas conclusiones sobre las desviaciones en cuanto a la planificación inicial, así como conclusiones sobre la experiencia de la realización del trabajo (lecciones aprendidas).

# 7. Autoevaluación

A continuación se presenta la autoevaluación individual realizada por cada miembro del equipo según la rúbrica proporcionada:

| **Rúbrica para evaluar los trabajos dirigidos** | | | | |
|---|---|---|---|---|
| **Exposición** | **Álvaro** | **Lidia** | **Víctor** | **Óscar** |
| Comprensión y dominio (puntuación triple) | Excelente | Excelente | Excelente | Excelente |
| Exposición didáctica | Excelente | Excelente | Excelente | Excelente |
| Integración del equipo | Excelente | Excelente | Excelente | Excelente |
| **Implementación** | **Álvaro** | **Lidia** | **Víctor** | **Óscar** |
| Objetivos (puntuación doble) | Excelente | Excelente | Excelente | Excelente |
| Aspectos didácticos | Excelente | Excelente | Excelente | Excelente |
| Experimentación y conclusiones | Excelente | Excelente | Excelente | Excelente |
| **Documentación** | **Álvaro** | **Lidia** | **Víctor** | **Óscar** |
| Contenidos | Excelente | Excelente | Excelente | Excelente |
| Divulgación contenidos | Excelente | Excelente | Excelente | Excelente |
| Bibliografía y recursos científicos | Excelente | Excelente | Excelente | Excelente |

Cuadro 1: Rúbrica para evaluar trabajos dirigidos (ponderada sobre 7 puntos)

# 8. Tabla de tiempos

El seguimiento del trabajo de cada participante del proyecto se ha dividido en cuatro fases claramente definidas, en las cuales cada miembro del equipo se centró en tareas específicas:

| **Fase** | **Descripción y seguimiento asociado** |
|---|---|
| Fase 1 | Desarrollo de la idea inicial e investigación (seguimiento 0 y 1) |
| Fase 2 | Implementación inicial mediante código (seguimiento 2) |
| Fase 3 | Finalización de la implementación y documentación (seguimiento 3) |
| Fase 4 | Preparación y presentación del proyecto (seguimiento 4) |

## Seguimiento Individual

| Fase | Álvaro Ruiz Gutiérrez | Lidia Jiménez Soriano | Víctor Flores González | Óscar Menéndez Márquez |
|---|---|---|---|---|
| Fase 1 | 23,74 horas | 0 horas | 0 horas | 0 horas |
| Fase 2 | 26,89 horas | 1 horas | 1 horas | 1 horas |
| Fase 3 | x horas | 2 horas | 2 horas | 2 horas |
| Fase 4 | x horas | 3 horas | 3 horas | 3 horas |
| **Total** | **70** | **70** | **70** | **70** |

Para consultar con mayor detalle las fechas específicas, tareas asignadas, tiempo dedicado por cada participante y otra información relevante, puedes acceder al siguiente enlace a la ficha de datos de Clockify: [LINK].

# Referencias


[1] C. Harris y M. Stephens, *A Combined Corner and Edge Detector*, Proceedings of The Fourth Alvey Vision Conference, pp. 147–151, 1988. Disponible en: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=88cdfbeb78058e0eb2613e79d1818c567f0920e2

[2] H. Bay, T. Tuytelaars y L. Van Gool, *Surf: Speeded up robust features*, en Computer Vision–ECCV 2006, Springer Berlin Heidelberg, pp. 404-417, 2006. Disponible en: https://link.springer.com/chapter/10.1007/11744023_32

[3] I. Rey-Otero y M. Delbracio, *Anatomy of the SIFT Method*, Image Processing On Line, vol. 4, pp. 370–396, 2014. Disponible en: https://doi.org/10.5201/ipol.2014.82

[4] N. Dalal y B. Triggs, *Histograms of oriented gradients for human detection*, en IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, EE.UU., vol. 1, pp. 886-893, 2005. DOI: 10.1109/CVPR.2005.177. Disponible en: https://ieeexplore.ieee.org/abstract/document/1467360