Visión Artificial

Juan Diego Gallego Nicolás
jdiego.gallego@um.es

23/03/2025

Entrega Parcial

Índice

Índice de figuras

1. Introducción

Este documento recoge explicaciones y resultados de los ejercicios realizados durante el desarrollo de la asignatura siguiendo las instrucciones del repositorio de GitHub albertoruiz/umucv/blob/master/notebooks/ejercicios.ipynb (revisado el 20/03/2025). Cada ejercicio se corresponde con la sección del documento con el mismo nombre. A su vez, cada sección se divide en los subapartados: Marco Teórico, Resultados y Comentarios.

La sección Marco Teórico servirá como resumen de los contenidos teóricos desarrollados en la teoría necesarios para la realización de la práctica. A continuación, en el apartado de Resultados se hablará del trabajo realizado (código y pruebas) y de cómo se han aplicado los conocimientos teóricos. Finalmente, la sección Comentarios se reserva para realizar alguna reflexión y plasmar las conclusiones y opiniones sobre el ejercio.

Las imágenes incluidas son extraídas de:

- Geogebra clásico: www.geogebra.org/classic?lang=es
- Geogebra 3D: www.geogebra.org/3d?lang=es
- Material de la asignatura: material de la asignatura
- Gráficas generadas con matplotlib
- Cámara de un Google Pixel 8 Pro
- Google Earth

A lo largo del documento se hará mención a una serie de scripts de python desarrollados por el profesor Alberto Ruiz. Estos scripts se encuentran en el repositorio de GitHub albertoruiz/umucv/ (revisado el 20/03/2025).

2. Calibración

2.1. Calibración: Marco Teórico

Una cámara es un dispositivo compuesto esencialmente por una lente que enfoca luz en un plano de proyección. Ese plano de proyección físico se sitúa detrás de la lente y tomando la imagen de forma invertida. Tras deshacer la inversión, los dispositivos de visualización presentan la imagen en la orientación correcta. Nótese que el proceso de reconstrucción de la imagen genera un plano de proyección virtual con las mismas dimensiones y a la misma distancia a la lente que el plano real 1.

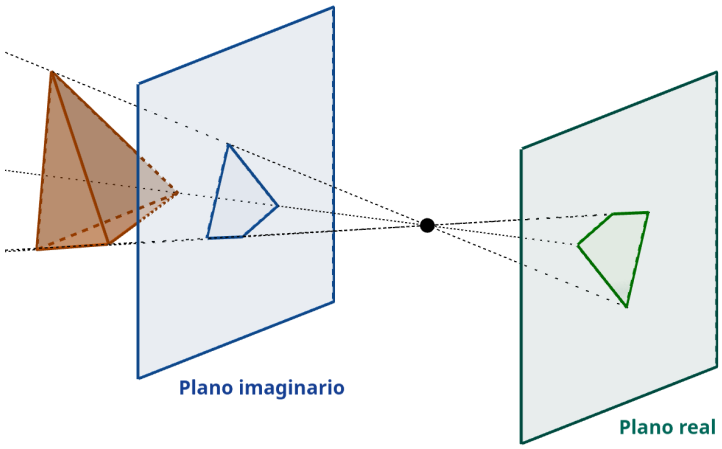

Figura 1: Modelo de una cámara.

Hay dos parámetros esenciales que describen las propiedades de una cámara: su resolución y su distancia focal. La resolución hace referencia a las dimensiones del plano de proyección. Esta se mide en píxeles (unidad mínima de representación de color) y se puede presentar como el par $W \times H$ o como el número de total píxeles en el plano. La distancia focal es la distancia entre el foco de la lente y el plano de proyección. También se mide en píxeles.

Calibrar la cámara consiste en aproximar lo máximo posible estos parámetros. Una vez conocidos, se pueden hacer mediciones en el mundo real de distancias, dimensiones y ángulos como veremos más adelante. En el caso de esta práctica, usaremos un patrón cuadriculado para este proceso 2, aprovechando las distorsiones de tamaños y de los ángulos rectos.

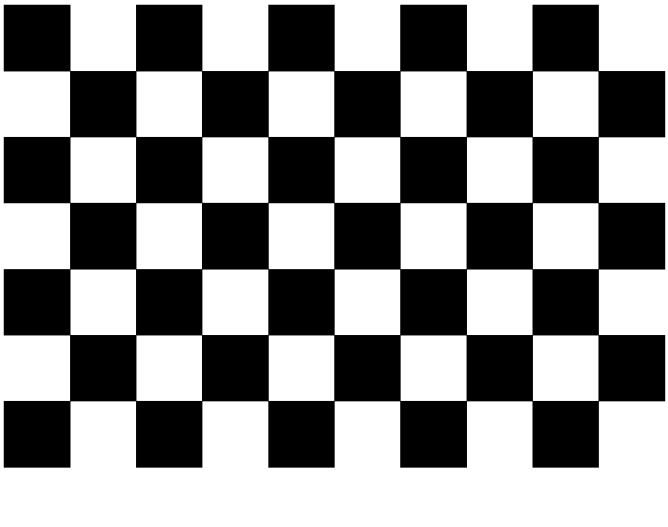

Figura 2: Patrón para la calibración de la cámara.

El proceso de calibración da como resultado una matriz $K \in \mathcal{M}_{3\times3}(\mathbb{R})$:

$$K = \begin{bmatrix} f & 0 & o_x \\ 0 & f_r & o_y \\ 0 & 0 & 1 \end{bmatrix}$$

donde:

- f, f_r es la distancia focal y un valor cercano
- (o_x, o_y) son las coordenadas en las que se sitúa la proyección del foco de la cámara (normalmente $(W/2, H/2)$)

Una propiedad de la cámara interesante, derivada de los parámetros anteriores, es el campo de visión o FOV (Field Of View). Es una medida angular que representa las amplitudes horizontal y veltical máximas reconocibles por la cámara 3. Aplicando trigonometría básica se derivan las siguientes fórmulas para el FOV horizontal y vertical:

$$FOV_H = 2 * arctan\left(\frac{w}{2f}\right);\ FOV_V = 2 * arctan\left(\frac{h}{2f}\right)$$

También, a partir del FOV se deduce la unidad angular mínima medible como $\frac{FOV_H}{W} = \frac{FOV_V}{H}$.

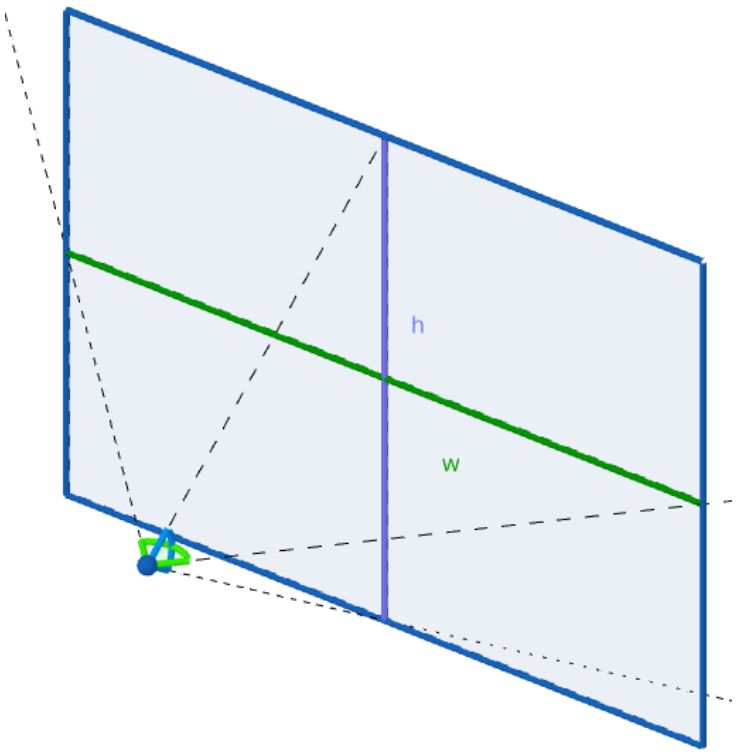

Figura 3: FOV

La última propiedad que necesitamos para la realización de la práctica es la relación entre tamaño, tamaño aparente, distancia a la lente y distancia focal. Esta la podemos obtener aplicando el Teorema de Tales aplicado a los triángulos formados por el foco de la cámara y los pares de extremos reales y virtuales del objeto proyectado 4.

Claro está que para que esto funcione el objeto a medir ha de estar lo más centrado posible, ya que en los extremos de la imagen se produce distorsión. En las imágenes tomadas para este proyecto se han intentado centrar lo máximo posible los objetos de interés evitando así dicho efecto.

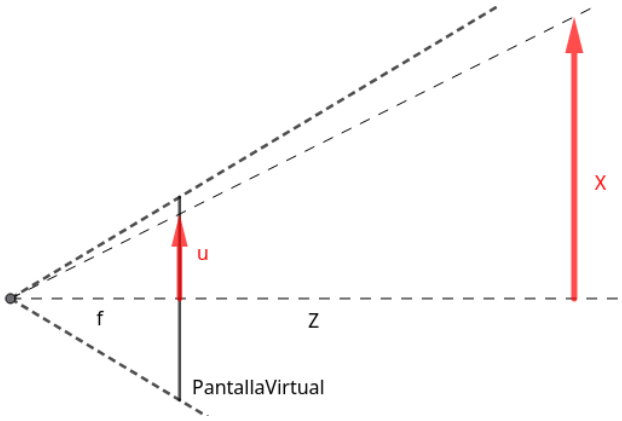

Figura 4: Medidas sobre la imagen.

2.2. Calibración: Resultados

La realización de los ejercicios se puede ver en el notebook de python 'calibracion.ipynb'.

2.2.1. Obtención de la matriz de calibración

El primer paso para la realización de la práctica es la calibración de la cámara. Para comenzar, he utilizado el programa 'stream.py' para tomar imágenes del patrón 2 desde una variedad de posiciones y ángulos 5.

Figura 5: Fotografía tomada para calibración.

Es importante que exista una variedad en los ángulos de captura ya que la funcionalidad findChessboardCorners de OpenCV utiliza la distorsión en

los ángulos de las esquinas de los cuadrados para calcular los parámetros que buscamos.

Una vez tenemos las imágenes, usamos el programa 'calibracion.py' para calibrar la cámara, dando como resultado la matriz de calibración:

$$K = \begin{bmatrix} 561{,}11 & 0{,}0000 & 317{,}80 \\ 0{,}0000 & 561{,}09 & 234{,}53 \\ 0{,}0000 & 0{,}0000 & 1{,}0000 \end{bmatrix}$$

para una resolución de 640×480. Con los programas 'triangulate.py' y 'verify.py' podemos comprobar que el proceso se ha completado con éxito.

2.2.2. FOV y pista de baloncesto

Con la distancia focal obtenida podemos calcular el FOV de la cámara:

$$FOV_H = 2 * arctan\left(\frac{w}{2f}\right) = 59{,}39^\circ$$

$$FOV_V = 2 * arctan\left(\frac{h}{2f}\right) = 46{,}32^\circ$$

Con este valor, podemos calcular la altura a la que tendríamos que situar la cámara para capturar por completo una pista de baloncesto a partir de sus dimensiones (28×15 metros):

$$tan\left(\frac{FOV_H}{2}\right) = \frac{28/2}{h_1} \Rightarrow h_1 = \frac{14}{tan(FOV_H/2)} = 24{,}5485625m$$

$$tan\left(\frac{FOV_V}{2}\right) = \frac{15/2}{h_2} \Rightarrow h_2 = \frac{7{,}5}{tan(FOV_V/2)} = 17{,}5346875m$$

De entre los dos resultados tenemos que tomar el mayor, pues necesitamos que la pista se vea tanto a lo largo como a lo ancho.

Buscando el tamaño del resto de líneas de la pista podemos hacer una representación en 3D que nos hace a la idea de la situación que experimentaría la cámara en estas condiciones 6:

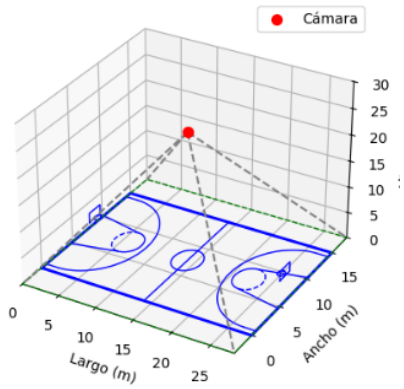

Figura 6: Cámara sobre la pista de baloncesto.

Como se puede observar, sobra espacio mas a allá bandas debido a que la altura necesaria para ver la distancia entre los laterales era menor. También se intuye que al estar a una determinada altura las canastas no son entran dentro de la pirámide que constituye el espacio de visión de la cámara. Aun así, posteriormente supondremos que las canastas se encuentran a nivel de suelo.

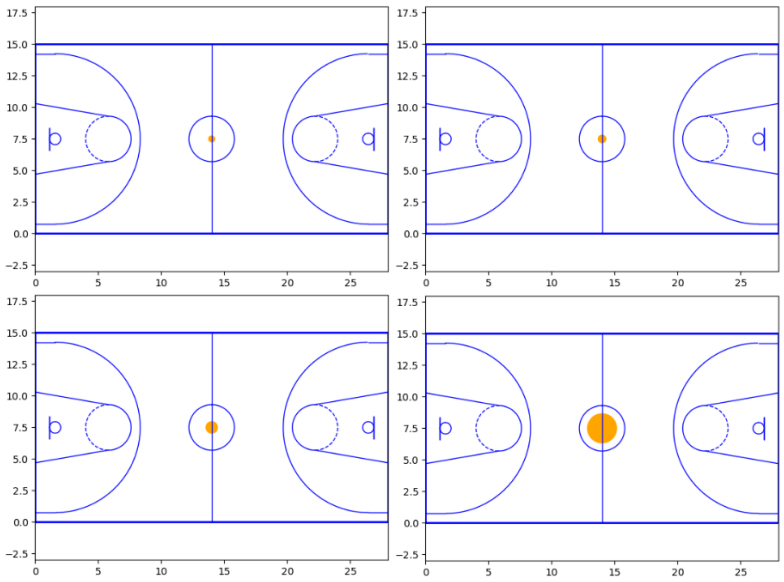

Figura 7: Pelota al 0 %, 25 %, 50 % y 80 % de la altura de la cámara.

En la imagen anterior 7 aprovechamos el modelo de la pista para simular el tamaño de una pelota de baloncesto situada a diferentes alturas del círculo central. Para la primera instancia, el tamaño de la pelota ($48cm$ de diámetro) es muy pequeño en comparación de la pista. En el tramo de 0 % a 50 % el tamaño aparente solo se duplica ya que esta magnitud es inversamente proporcional a la distancia a la cámara. Al 80 % de la distancia al suelo el tamaño se ha multiplicado por 5. Para valores más altos la pelota empieza a ser demasiado grande como para caber en el espacio visual de la cámara.

2.2.3. Medir objetos

Vamos ahora a explorar ahora las posibilidades de la relación

$$\frac{u}{f} = \frac{X}{Z}$$

utilizando la herramienta 'mouse.py' desarrollada en las prácticas que nos permite medir distancias en una imagen conociendo la matriz de calibración.

En primer lugar, podemos predecir la altura en píxeles que tendrá una persona a una cierta distancia de la cámara. Por ejemplo, en la imagen 8 aparezco yo mismo situado a 7,2 metros de la cámara. Sabiendo que mi altura es de 1,74 metros mi tamaño aparente debería de ser:

$$u = \frac{X}{Z} \times f = \frac{1{,}74}{7{,}2} \times 561{,}11 = 135{,}60px$$

Por tanto, la altura medida entra dentro de la precisión esperada.

Figura 8: Altura de una persona

Por otro lado, vamos a intentar medir un objeto directamente con la cámara. Para ello, situamos un muñeco a 19,5cm de la cámara y utlizamos el mismo programa para medir su tamaño aparente 9.

Figura 9: Tamaño de un objeto

Con el dato de 279px utilizamos la misma relación para obtener el tamaño del muñeco:

$$X = \frac{u}{f} \times Z = \frac{279}{561{,}11} \times 19{,}5 = 9{,}7cm$$

lo que coincide con el tamaño real con una precisión milimétrica.

Lo último que podemos hacer directamente con esta relación es calcular la distancia a la que se encuentra un objeto cuyas dimensiones conocemos. Por ejemplo, un coche que mida 4 metros de largo y que ocupe 20 píxeles en pantalla estará a una distancia de:

$$Z = \frac{561{,}11}{20} * 4 = 112{,}2m$$

2.2.4. Triangulación

El programa 'angle.py' es una variación de 'mouse.py' que permite medir ángulos sobre una imagen. Esta medición se calcula sobre el segmento que une dos puntos seleccionados. Para ello, se completan las coordenadas sobre la imagen (con respecto al centro calculado en la matriz de calibración) con una tercera componente dada por la distancia focal. Después, el ángulo entre los dos puntos se deriva de la fórmula:

$$\langle \vec{u}, \vec{v} \rangle = |\vec{u}||\vec{v}| \cos(\widehat{\vec{u}\vec{v}})$$

Con la capacidad de calcular ángulos sobre una imagen vamos a intentar triangular la posición desde la cual se ha tomado una fotografía 10.

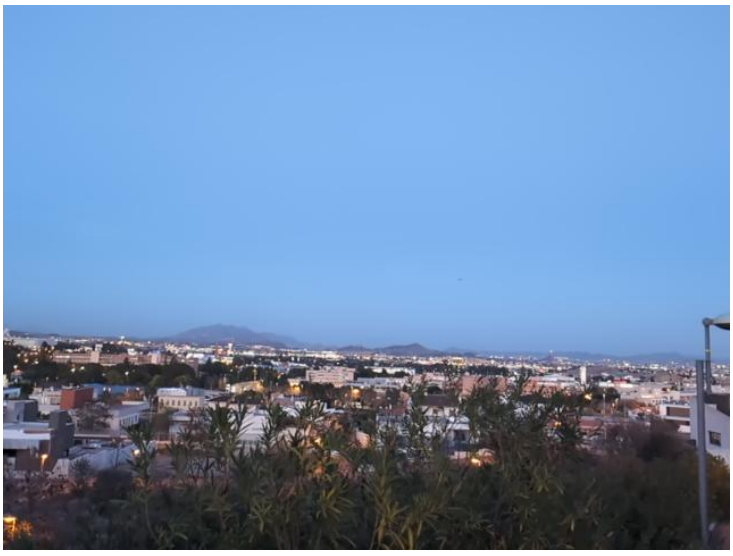

Figura 10: Paisaje sobre el que triangular.

El primer paso para hacer la triangulación es elegir tres puntos distinguidos. En mi caso y debido a lo sencillos que son de reconocer serán:

NC: El poste del centro comercial Nueva Condomina

TH: El psote del centro comercial Thader

MA: El Cristo de Monteagudo

Para medir su distancia angular de la forma más precisa posible primero debemos fijar una línea recta sobre la cual tomar las medidas. En mi caso tomaré la línea del horizonte 11

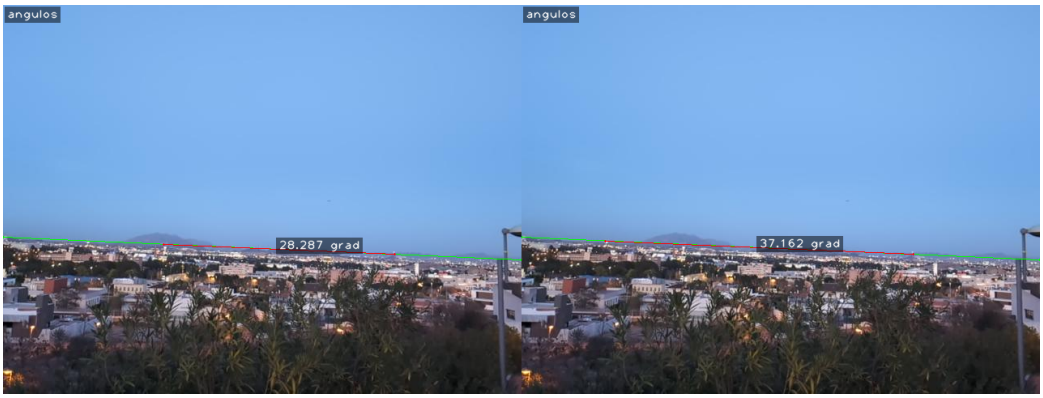

Figura 11: Midiendo ángulos.

Eligiendo los puntos de esta línea (verde) más cercanos a los puntos seleccionados podemos determinar que el ángulo $NC - MA$ es de 37,162° y el ángulo $TH - MA$ es de 28,287°. Con acceso a las distancias relativas entre estos puntos y conociendo desde dónde he tomado esta fotografía podemos ver en la imagen 12 que el error absoluto no llega a los 0,6°.

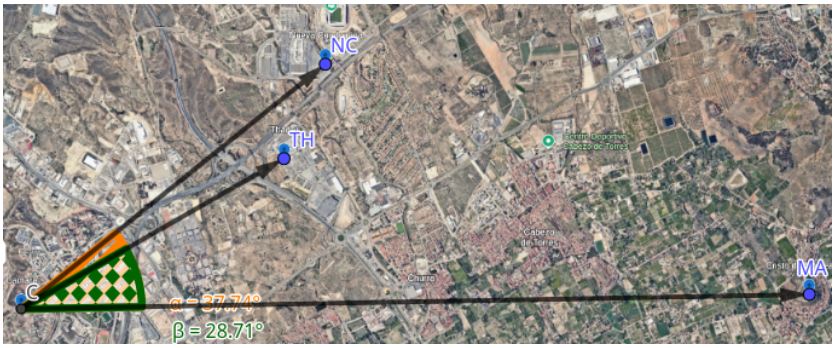

Figura 12: Ángulos reales.

Utilizando Google Earth para calcular las distancias entre los tres puntos, los ángulos medidos y el Teorema del Ángulo exterior tenemos ya todo lo necesario para terminar la triangulación. El último fragmento de código del notebook 'calibracion.ipynb' calcula los respectivos centros y radios de las dos circunferencias en cuya intersección se encuentra el punto desde el que se toma la fotografía, fijando el punto TH como origen de coordenadas y el eje $TH \rightarrow MA$ como dirección positiva del eje de abcisas. Trasladando los datos a la calculadora gráfica de Geogebra y teniendo en cuenta la posición relativa de los puntos observados para desambiguar entre las dos posibles intersecciones obtenemos una aproximación al punto real con una precisión de unos 89 metros 13.

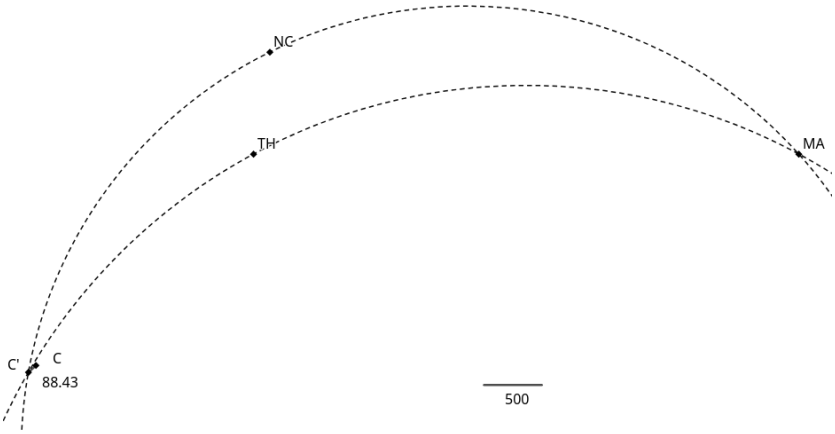

Figura 13: Triangulación.

3. Filtros

3.1. Filtros: Marco Teórico

Un filtro de imagen es una transformación que se aplica a la misma para modificar o extraer algunos aspectos de la misma. El valor del filtrado para cada píxel se calcula a partir de un entorno o vecindad del mismo. Los filtros se pueden clasificar según de la manera que se apliquen o de los efectos que tengan en la iamgen. Para el desarrollo de este ejercicio destacamos las clases:

- Filtros de suavizado: atenuan el ruido y revierten la discretización de la imagen. De ahí el calificativo de 'suave', ya que aproximan un conjunto de valores discretos a una curva suave.
- Filtros de detección de bordes: resaltan los perfiles de los objetos de una imagen destacando las transiciones entre ellos.
- Filtros lineales: el valor de un pixel se calcula como combinación lineal de sus elementos vecinos.
- Filtros no lineales: el cálculo del filtro no se realiza como una combinación lineal de sus elementos vecinos.

En esta sección comentaremos el funcionamiento de los filtros y en la siguiente hablaremos de sus efectos en la imagen.

3.1.1. Filtro de medias

El filtro de medias es un filtro lineal de suavizado. Para cada píxel, se calcula como una operación de promedio sobre una vecindad del mismo. Esta vecindad es un área 3×3, 5×5... centrada en el píxel. Un ejemplo de cálculo del filtro para una vecindad 3×3 en una imagen con un solo canal de color podría ser:

$$\begin{bmatrix} 0 & 100 & 133 \\ 24 & 112 & 67 \\ 3 & 129 & 33 \end{bmatrix} \Rightarrow \frac{0 + 100 + 133 + 24 + 112 + 67 + 3 + 129 + 33}{9} = 66,78$$

Para píxeles del borde existen formas de adaptar sus vecindades como completar con ceros o sustituirlas por los rectángulos sobrantes. La complejidad óptima en tiempo y en espacio para el cálculo de este filtro para una imagen de $n \times m$ píxeles con vecindades de tamaño $k \times k$ es $O(n \times m)$. Para con-

seguir esta complejidad es necesario mantener una matriz auxiliar de sumas acumuladas para calcular las sumas en rangos en $O(1)$.

3.1.2. Filtro Gaussiano

El filtro Gaussiano es otro filtro lineal de suavizado que se define mediante una función gaussiana discreta. Dadas las coordenadas relativas (x,y) de un píxel al central, el coeficiente por el que se multiplica es:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{\frac{-x^2-y^2}{2\sigma^2}}$$

donde σ es la desviación típica de la distribución y el parámetro que determina la cantidad de suavizado.

La principal diferencia de este filtro con el anterior es que mientras que en el filtro de medias todos los píxeles tienen el mismo peso, el filtro gaussiano da mayor peso a los píxeles más próximos al central. Además, la distribución gaussiana en dos dimensiones tiene dos propiedades muy deseables:

1. Separabilidad: el filtro puede ser descompuesto en dos filtros unidimensionales $G(x, y) = G_1(x) \times G_2(y)$ donde G_1 y G_2 son las distribuciones para las filas y las columnas respectivamente.
2. Cascading: aplicar dos filtros gaussianos con desviaciones típicas σ_1 y σ_2 tiene el mismo efecto que aplicar un filtro con desviación típica $\sigma = \sqrt{\sigma_1^2 + \sigma_2^2}$

La primera propiedad permite reducir la complejidad temporal en la aplicación del filtro de $O(n \times m \times k^2)$ a $O(n \times m \times k)$ donde k es el tamaño de vecindad que OpenCV considera apropiado para un σ dado.

3.1.3. Filtro bilateral

Un filtro bilateral es un filtro no lineal de suavizado que preserva los bordes. Además de utilizar la proximidad espacial, también tiene en cuenta la cercanía de los colores a la hora de dar más o menos peso a píxeles vecinos. El efecto que se consigue es el suavizado del ruido y de las texturas de los objetos mientras que los bordes que los separan aparecen inalterados. Al no ser un filtro lineal y dependiente de la estadística de la imagen resulta difícil encontrar una optimización. Así pues, este es el filtro más lento en su aplicación, como veremos más adelante. Su complejidad temporal es $O(n \times m \times k^2)$

3.1.4. Filtro de medianas, mínimos y máximos

Como su propio nombre indica, estos filtros utilizan la mediana, el mínimo y el máximimo de una vecindad como filtro. Claramente, son filtros no lineales. El de medianas se puede considerar como un filtro de eliminación de ruido granular ya que permite discriminar valores extremos de una distribución. Los filtros de mínimos y máximos son sensibles a píxeles oscuros y luminosos.

3.1.5. Filtros de borde

Para la realización de este ejercicio se han añadido dos filtros lineales que permiten la visualización de bordes. El primero permite detectar bordes horizontales convolucionando con el núcleo $[-1, 0, 1]$. En texturas uniformes, el resultado de aplicar este filtro da números cercanos a 0. El segundo es un laplaciando discreto de tamaño 3×3 el cual permite detectar bordes en todas las direcciones.

3.2. Filtros: Resultados

Para este ejercicio se ha desarrollado el script 'filtros.py'. Al iniciarlo, se muestra una ventana con el dispositivo de entrada elegido y se muestra por terminal un mensaje que sugiere pulsar la tecla 'h' para mostrar la ventana de ayuda 14.

La aplicación permite:

- Seleccionar hasta 8 filtros diferentes (teclas 1 a 8) o mostrar la fuente sin alterar (0)
- Alternar entre la vista en color y en escala de grises (tecla c)
- Aplicar el filtro únicamente a una región de interest (ROI) rectangular seleccionada con el ratón o a la imagen completa (tecla r)
- Desmarcar la región de interés (tecla x)
- Invertir los colores (tecla i)
- Pausar el vídeo (tecla SPACE)
- Tomar una captura de pantalla (tecla s)
- Mostrar el menú de ayuda (tecla h)

– Cerrar la aplicación (tecla q)

Figura 14: Menu de ayuda en filtros.py.

3.3. Filtro de medias

Pulsando la tecla '1' seleccionamos el filtro de medias o de caja. Con el slider RAD podemos controlar el tamaño de las vecindades. Como vemos en 15, se consigue el efecto de suavizado o desenfoque. Al sobrepasar cierto radio, algunos de los elementos de la imagen dejan de intuirse como algunos puntos negros de la pelota, el hueco de la cantimplora o la pantalla del teléfono. Esto es debido a que su grosor es pequeño comparado con el radio. También se intuyen líneas verticales y horizontales artificiales consecuencia de tomar vecindades perfectamente cuadradas y contar con solo dos direcciones de filtrado. En cuanto a la latencia, podemos observar en la esquina superior izquierda de la pantalla que es del orden de 1 milisegundo, resul-

tando altamente eficiente. Además, esta latencia no varía con el tamaño de vecindad elegido como se comentaba en la sección previa.

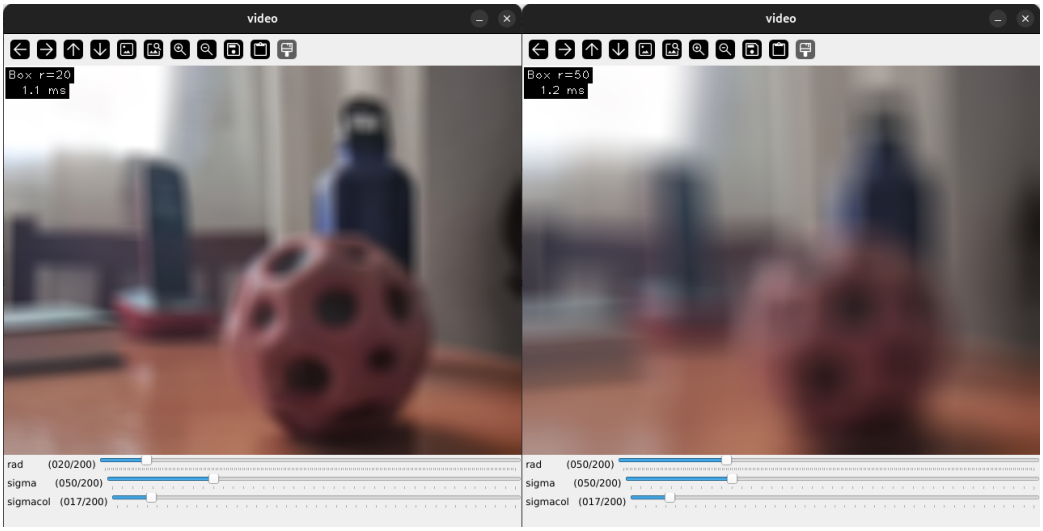

Figura 15: Filtro de medias.

3.4. Filtro gaussiano

4. Clasificador

Aquí se hablará sobre los clasificadores utilizados para la tarea específica. Se puede incluir la descripción de los algoritmos empleados, su rendimiento y los datos con los que se entrenaron.

Figura 16: Una imagen de ejemplo.

Como se puede ver en la 16, esta es una imagen de ejemplo que se utiliza en el documento.