Imanzaneq3@alumno.uned.es

Visión Artificial

PEC 1 curso 2023-2024

Índice de contenido

[Eliminación de ruido en imágenes escaneadas 2](#_Toc149666210)

[1. Análisis del ruido encontrado. 2](#_Toc149666211)

[2. Descripción de la solución. 4](#_Toc149666212)

[3. Ejemplo de ejecución. 9](#_Toc149666213)

[Transformaciones geométricas 14](#_Toc149666214)

[1. Apartado a. 16](#_Toc149666215)

[2. Apartado b 16](#_Toc149666216)

[3. Apartado c 20](#_Toc149666217)

[4. Apartado e 22](#_Toc149666218)

# Eliminación de ruido en imágenes escaneadas

## Análisis del ruido encontrado.

**Ruido “Sal y Pimienta”**

Se caracteriza por cubrir de forma dispersa la imagen con una serie de píxeles blancos y negros. El pixel, en lugar de tomar un valor ideal similar al de su vecindario, toma un valores muy altos o bajos. Puede ocasionarse por perturbaciones que afectan a la señal de la imagen, errores en la transmisión de datos o fallos en los sensores de la cámara. Este ruido puede tratarse aplicando un filtro de mediana y posteriormente restaurando los detalles que puedan haberse perdido.

A drawing of a cube

Description automatically generated

**Ruido en forma de mancha**

Este ruido se presenta como irregularidades en la imagen que se asemejan a manchas oscuras. Como tratamos con imágenes escaneadas, pueden deberse a problemas durante el escaneo como suciedad en el vidrio del escáner o en el mismo documento. Como hay muchas tonalidades de gris, podemos eliminar gran parte del ruido utilizando un filtro de umbralización y después restaurar los detalles que puedan haberse perdido de los contornos.

A black and white image of a circle

Description automatically generated

**Ruido en forma de patrones**

Podemos observar ruido que se caracteriza por seguir patrones de píxeles que son diferentes en color o intensidad a los píxeles circundantes. Al ser parecido al ruido “sal y pimienta” (aunque formando un patrón), gran parte del ruido podrá eliminarse aplicando un filtro de mediana a la imagen. Si después sigue quedando algún resto, se puede utilizar un filtro de umbralización binaria en la región de interés.

A drawing of two houses

Description automatically generated

**Ruido en forma de líneas**

Este ruido es creado por píxeles de diferente color / intensidad a los de sus píxeles vecinos y que forman una línea que atraviesa la imagen verticalmente. Las lineas que vemos en estos dibujos tienen la particularidad de ser de una tonalidad de gris diferente a la del resto del dibujo. Se puede utilizar un filtro de umbralización binaria de manera que los píxeles se clasifiquen como negro o blanco en función de si el valor de su intensidad está por debajo o por encima de un nivel de umbral.

A drawing of a house

Description automatically generated

**Anotaciones**

Información adicional (como comentarios o marcas) que se ha añadido a una imagen. Estas anotaciones se colocan con el propósito de explicar o resaltar ciertas áreas o características y, aunque pueden resultar útiles para la comunicación, a veces pueden ser consideradas ruido y en tal caso puede ser necesario eliminarlas. Si este es el caso, podemos utilizar un filtro de umbralización en la región de interés que establezca todos los píxeles cuya intensidad es diferente de la del fondo a la intensidad de los píxeles del fondo.

A drawing of a fish

Description automatically generated

## Descripción de la solución.

Como solución a este ejercicio, se ha creado una aplicación sencilla que permite al usuario especificar la imagen sobre la que realizar los cambios y aplicar distintos operadores de manera secuencial. A continuación se da una explicación de cada operador y un ejemplo de su aplicación.

**Filtro Non-Local-Means**

Elimina el ruido impulsivo utilizando la función “fastNlMeansDenoising”. Esta función implementa un algoritmo de eliminación del ruido basado en un método de filtrado no local y adaptativo: en lugar de calcular el valor de un píxel basándose solo en sus píxeles vecinos, se consideran regiones similares en toda la imagen. Este enfoque es más efectivo para preservar los bordes y las estructuras en la imagen mientras se elimina el ruido. A más intensidad de filtro, mayor eliminación del ruido y mayor pérdida potencial de los detalles.

A screenshot of a computer screen

Description automatically generated

**Filtro Mediana**

Elimina el ruido impulsivo aplicando la función “medianBlur”. Esta función sustituye el valor de cada píxel en la imagen por la mediana de los valores de los píxeles vecinos en una ventana definida. A diferencia del filtro anterior, este es un método de suavizado local y no es tan efectivo para preservar los bordes y estructuras de la imagen. A mayor tamaño de ventana más se elimina el ruido, pero existe una mayor pérdida de detalle. Ejemplo en la misma imagen utilizada anteriormente:

Para un tamaño de ventana = 3

A screenshot of a drawing

Description automatically generated

Para un tamaño de ventana = 5

A screenshot of a computer screen

Description automatically generated

Puede apreciarse como este filtro es menos efectivo que el filtro Non-Local-Means ya que o bien no elimina completamente el ruido o bien lo hace a costa de una gran pérdida de detalle.

**Filtro Umbralización**

Elimina el ruido en tonalidades de gris aplicando la función “threshold”. Este filtro de umbralización binaria convierte una imagen en escala de grises a una imagen binaria: los píxeles se clasifican como blanco si el valor de su intensidad está por encima del umbral y como negro si está por debajo.

**A drawing of a house

Description automatically generated**

**Filtro negro**

Este filtro compara los píxeles de la imagen con un valor umbral. Si la intensidad del pixel es menor a la del valor umbral, establece el pixel a blanco. Útil para eliminar detalles aislados e indeseados de la imagen. En el ejemplo se ha aplicado a la región de donde están las anotaciones, en caso de que quisiéramos eliminarlas.

**A drawing of a kite

Description automatically generated**

**Realzar bordes**

Utiliza la función “filter2D” para realzar los bordes de una imagen aplicando una operación de convolución y un kernel predefinido de refinado de realce de bordes. Muy útil cuando los bordes de la imagen se han difuminado como consecuencia de una operación anterior. A continuación, un ejemplo en el que se realzan los bordes difuminados tras aplicar el filtro Non-Local-Means:

**A drawing of a cube

Description automatically generatedA drawing of a cube

Description automatically generated**

**Erosionar**

Utiliza la función “dilate” para dilatar el fondo y así erosionar los contornos de los dibujos, reduciendo el grosor de sus bordes. Esto es útil también para la eliminación de pequeños detalles y ruido, como se puede ver en el ejemplo:

A screenshot of a screenshot of a computer screen

Description automatically generated

**Dilatar**

Utiliza la función “erode” para erosionar el fondo y así dilatar los contornos de los dibujos, aumentando el grosor de sus bordes. Esto también aumentará el grosor de los pequeños detalles y el ruido, como se puede ver en el ejemplo:

A screenshot of a screenshot of a computer screen

Description automatically generated

**Seleccionar ROI**

Cualquiera de las anteriores funciones puede utilizarse sobre la imagen al completo o sobre una región de interés. Esta opción permite al usuario seleccionar una región sobre la que aplicar alguno de los operadores, sin que el resto de la imagen se vea afectada. Como ejemplo, se aplica el filtro Non-Local-Means con una gran intensidad de filtro a una región aislada de una imagen:

A drawing of a house

Description automatically generated

**Mostrar resultado**

Muestra 3 imágenes: La imagen original, la imagen modificada y una comparación Antes-Después de los cambios realizados.

**Restablecer**

Restablece la imagen original, eliminando todos los cambios realizados.

## Ejemplo de ejecución.

**N\_328\_THS\_TOTAL-ev1-h.png**

En esta imagen podemos identificar un poco de ruido “sal y pimienta” por toda la imagen y una mancha en la parte baja de una de las figuras. Aplicamos el **Filtro Umbralizacion** con threshold = 1 en la región de la mancha. Esto hará que se eliminen todas las tonalidades de gris intermedias

A drawing of a house

Description automatically generatedA drawing of a house and a wire

Description automatically generated

A screenshot of a computer screen

Description automatically generated

Aplicamos el **Filtro** **Non-Local-Means** con intensidad de filtro = 30 para eliminar el ruido “sal y pimienta”

A screenshot of a computer screen

Description automatically generated

Aplicamos el **Filtro** **Negro** con umbral = 255 para eliminar parte del ruido residual negro de la mancha:

A screenshot of a computer screen

Description automatically generated

Por último, realizamos **Erosión** seguido de **Dilatación** en el borde inferior de la figura para restablecer la línea

A screenshot of a computer screen

Description automatically generated

El resultado se muestra a continuación:

A drawing of a house and a house

Description automatically generated

**N\_307\_GLS\_TOTAL-ev5- h.png**

En esta imagen podemos identificar ruido “sal y pimienta” por toda la imagen y un patrón vertical el el lateral izquierdo de la imagen. Este patrón está formado por píxeles ruidosos en forma de círculos.

A drawing of a house

Description automatically generated

Aplicamos el **Filtro** **Non-Local-Means** con intensidad de filtro = 45 para eliminar el ruido “sal y pimienta”. Como el patrón está también compuesto de píxeles ruidosos tipo “sal y pimienta”, este filtro eliminará también parte del patrón.

A drawing of a house

Description automatically generated

Al haber zonas en las que el ruido no se ha eliminado completamente, en lugar de aumentar la intensidad del filtro Non-Local-Means (y la consecuente pérdida de definición de los bordes), vamos a aplicar el **Filtro Umbralizacion** con threshold = 5 en las regiones de interés.

A screenshot of a screenshot of a drawing of a house

Description automatically generated

A drawing of a house

Description automatically generated

Tras aplicar todos los cambios, el resultado obtenido es:

A screenshot of a phone

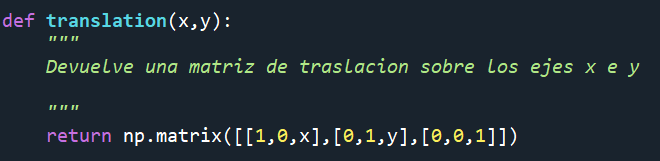
Description automatically generated

# Transformaciones geométricas

Para la resolución de este ejercicio, se ha optado por reutilizar parte del código del ejercicio 1 y crear una interfaz sencilla en la que se puede seleccionar cada apartado del ejercicio.

Comenzamos con una breve mención a las matrices que se han construido para las distintas transformaciones. En OpenCV, las transformaciones afines 2D se representan comúnmente mediante matrices de transformación 2x3, por lo que la función cv2.warpAffine() tomará únicamente las dos primeras filas de las matrices devueltas.

**Matriz de traslación**

A screenshot of a math problem

Description automatically generated

**Matriz de escalado**

A computer screen shot of a code

Description automatically generatedA screenshot of a cell phone

Description automatically generated

**Matriz de rotación**

A computer screen shot of a computer code

Description automatically generatedA screenshot of a math problem

Description automatically generated

**Matriz de cizallamiento**

A computer screen shot of a code

Description automatically generatedA close-up of a graph

Description automatically generated

## Apartado a.

Opción número 1 en la interfaz: **Reescalar y rotar**, implementada en **def reescalarYrotar(img).** En esta implementación, se utilizan las matrices anteriormente definidas para realizar las siguientes transformaciones:

* Reescalado de la imagen a 200 x 200 pixel
* Traslación del centro de la imagen al centro de coordenadas
* Rotación de 45 grados
* Reescalado 1:0.7
* Traslación del centro de la imagen a su posición original

El resultado es una imagen reescalada y rotada cuyas esquinas tocan los bordes de la imagen de salida. Como los reescalados y las rotaciones se producen en función al centro de coordenadas, trasladamos primero la imagen para que su centro coincida con el centro de coordenadas, realizamos los cambios y por último devolvemos el centro de la imagen a sus coordenadas originales.

A screenshot of a cellphone

Description automatically generated

## Apartado b

**Subapartado b.1.1**

Opción número 2 en la interfaz: **Transformacion afin compuesta 1**, implementada en **def transformacionAfinCompuesta1(img).**

Comenzamos reescalando la imagen a 400x400 pixels para trabajar con ella más cómodamente. La implementación se divide en apartados en los que se realizan las y aplican las 3 transformaciones por separado. Note que aunque se muestran los resultados de cada transformación individual, la función solo devuelve la imagen final. Se han conseguido las imágenes intermedias comentando las transformaciones posteriores y devolviendo el resultado de la transformación actual.

* Transformación 1:

Se aplica la matriz de cizallamiento sobre el eje X con un ángulo de -30 grados seguida de una traslación para devolver la imagen al centro de la ventana de la imagen de salida.

A screenshot of a phone

Description automatically generated

* Transformación 2:

Se aplica a la imagen una traslación de su centro al centro de coordenadas, se realiza la rotación de 90 grados y se hace otra traslación para devolver el centro de la imagen a su posición original (teniendo en cuenta que la rotación de 90 grados intercambia el ancho y el largo de la imagen). El resultado es el siguiente:

A black and white photo of a bridge

Description automatically generated

* Transformación 3:

Se aplica a la imagen una matriz de escalado a la mitad del tamaño de la imagen anterior en ambos ejes:

**A train tracks in the distance

Description automatically generated with medium confidence**

**Subapartado b.1.2**

Opción número 3 en la interfaz: **Transformacion afin compuesta 2**, implementada en **def transformacionAfinCompuesta2(img).** El resultado es el mismo que en el caso anterior pero esta vez, en lugar de aplicar las matrices una a una, se definen las matrices de las transformaciones individuales y se componen en una única matriz transformación. Además, nos ahorramos una traslación.

**A train tracks in the distance

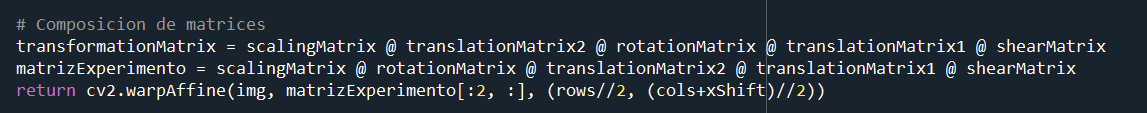
Description automatically generated with medium confidence**

**Subapartado b.2**

El orden de las transformaciones sí es importante debido a que la multiplicación de matrices no es conmutativa. Esto significa que el resultado de multiplicar dos matrices M1 y M2 no es necesariamente el mismo a multiplicar las matrices M2 y M1:

**M1 X M2 /= M2 X M1**

En nuestras imágenes, si se aplican las transformaciones con matrices en un orden diferente esto puede conducir a resultados diferentes. Podemos hacer un pequeño experimento modificando el orden de nuestra composición de matrices del apartado b.1.2:



El resultado de aplicar la matrixExperimento sobre la imagen da un resultado totalmente distinto:

A black and white photo of a black and white photo of a black and white photo of a black and white photo of a black and white photo of a black and white photo of a black and

Description automatically generated

**Subapartado b.3**

Opción número 7 en la interfaz: **Llevar rectangulo a objetivo**, implementada en **def rectanguloAobjetivo ().** En este apartado comenzamos creando una imagen sintética con las especificaciones del enunciado: un rectángulo de lado 40 x 20 sobre un fondo con una inclinación de 60 grados con el eje X.

**A screenshot of a computer

Description automatically generated**

La secuencia de transformaciones simples que se van a realizar para posicionarlo en la posición objetivo son las siguientes:

* Traslación de (-100, -80)las
* Rotación de 30 grados
* Reescalado 1:0.5. Por posiciones finales de los puntos A, B, C y D se puede ver que el rectángulo final tiene un tamaño de 20 x 10.

Aplicamos a la imagen una matriz transformación que es composición de las matrices que definen las transformaciones simples anteriores. Como la transformación a la imagen se realiza en un solo paso, no se pierde información en ninguna subimagen.

**A screenshot of a computer

Description automatically generated**

## Apartado c

**Subapartado c.1**

Opción número 4 en la interfaz: **Transformacion polar**, implementada en **def transformacionPolar(img).** Obtenemos el siguiente resultado:

**A screenshot of a phone

Description automatically generated**

El enunciado especifica que el resultado debe ser “una imagen logPolar cuadrada, centrada en el centro de la imagen original con una fila por grado y ajustada para ver toda la imagen original. Es por esto que los parámetros que se pasan a la función son:

* El tamaño de salida será de 360 x 360, lo que resultará en una imagen cuadrada con una fila por grado
* imgCenter: El centro será el de la imagen original
* radius: El radio máximo desde el centro de la imagen a los bordes
* El flag cv2.WARP\_POLAR\_LOG, para realizar una transformación polar-logarítmica

Este tipo de transformación convierte una imagen en coordenadas polares: los ángulos se representan en el eje horizontal y las distancias radiales desde el centro de la imagen el eje vertical.

**Subapartado c.2**

Opción número 5 en la interfaz: **Deshacer transformacion polar**, implementada en **def** **deshacerPolar(img).** Obtenemos el siguiente resultado:

**A train tracks in a valley

Description automatically generated**

Se ha producido una pérdida de calidad en la imagen resultado. Esto puede deberse a que durante la transformación polar, cierta información de la imagen original puede perderse o representarse diferente en el espacio polar. Al invertir la transformación, esta pérdida de información puede ocasionar que la imagen resultado no sea igual que antes de realizar la transformación polar.

## Apartado e

Opción número 6 en la interfaz: **Transformacion no lineal**, implementada en **def** **transformacionNoLineal(img).** Obtenemos el siguiente resultado:

A black and white image of a bridge

Description automatically generated

Para llegar a este resultado se ha optado por dividir la imagen original en dos mitades. Tras separar ambas mitades, se han utilizado matrices de escalado para estrechar la imagen izquierda a un tercio del tamaño de la imagen original y ensanchar la mitad derecha a dos tercios de la imagen original. Por último, se unen ambas mitades.

El operador a utilizar depende del tipo de transformaciones a aplicar: warpAffine se emplea para transformaciones que conservan las líneas y el paralelismo y warpPerspective para transformaciones no lineales. En una transformación no lineal como esta asumiríamos que debemos utilizar la segunda, pero como mi solución consiste en la composición de dos transformaciones lineales, he utilizado warpAffine en ambas mitades.

Aunque haya realizado esta transformación con dos transformaciones lineales, la transformación resultado es una transformación no lineal porque no conserva las proporciones de las formas de la imagen. En las transformaciones lineales las relaciones entre las partes de la imagen se mantienen, cosa que no ocurre en la imagen resultado.