2. Filtrado espacial

Resumen

Aprenderemos a inyectar diferentes tipos y cantidades de ruido en la imagen, y a eliminarlo o reducirlo mediante filtrado de dominio espacial. En la práctica, introducir ruido intencionalmente puede tener dos propósitos: conocer estos ruidos y estudiar el efecto de diferentes tipos de filtrado en condiciones controladas. Analizaremos diferentes filtros y sus pros y contras. Es fundamental que comprendas qué filtro es más adecuado para cada tipo de ruido.

Palabras clave

Ruido de sal y pimienta • Ruido gaussiano • Filtros media, media y gaussiano

Contenido

1. Generando imágenes ruidosas

Ruido de sal y pimienta

Produciremos ruido de sal y pimienta con operaciones básicas de NumPy. Como en otros casos en los que se proporciona código, es muy importante que comprenda qué hace cada parte del código y con qué propósito, de modo que pueda aprender a escribir o modificar código para situaciones similares.

Ruido gaussiano Para

producir ruido gaussiano, utilizamos el módulo numpy.rando metro. Tenga en cuenta que no utilizamos una función que aplica directamente el ruido a una imagen, sino que primero generamos ruido gaussiano en una matriz del mismo tamaño de la imagen original y luego agregamos
E4 este resultado a la imagen. Pruebe también con ruido gaussiano con σ el doble del valor máximo utilizado en el código original.

2. Filtrado de imágenes

Para evaluar el valor práctico de un filtro como los estudiados aquí, debemos considerar, al menos: (1) su eficacia para eliminar el ruido; (2) su comportamiento preservando los bordes del objeto; y (3)
E1 su coste computacional en función del tamaño de la máscara (filtro).

Filtro de media

Para aplicar el filtro de media, procederemos de forma general: primero se crea la máscara del tamaño dado y luego se aplica una convolución
E2 . Asegúrese de comprender cómo se define la máscara en el código proporcionado. Para convolución, el módulo sci py.ndimage.filtros se utiliza.

Tenga en cuenta que el código proporcionado aplica el filtro medio a imágenes con ruido de sal y pimienta. Analice qué tan efectivo es este filtro para ese tipo de ruido.

Filtro gaussiano

Para el filtro gaussiano, utilizamos el módulo scipy.ndimage .filters. Como puedes ver, este filtro se aplica a imágenes con ruido gaussiano. ¿Este filtro funciona para este ruido?

Experimente con varios valores de la desviación estándar del ruido y varios valores de la desviación estándar del filtro gaussiano. Por favor, preste atención al hecho de que las funciones gaussianas se utilizan tanto para el ruido como para el filtro, pero con significados diferentes; No te confundas y asegúrate de entender qué representa la función gaussiana en cada caso.

Para apreciar mejor el resultado del filtrado, puede ampliar un área relativamente pequeña de la imagen (por ejemplo, la nariz, un ojo o una parte del sombrero). E7

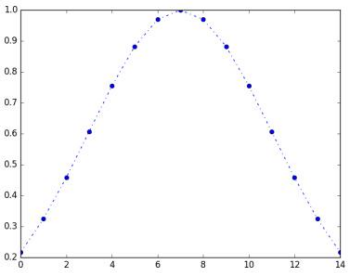
Filtro de

mediana Ahora usamos el módulo scipy.signal. Observe el efecto del filtro mediano sobre el ruido de sal y pimienta. Con respecto al filtro medio, ¿el filtro mediano es más o menos efectivo? ¿Por qué? ¿Ocurre lo mismo si el tamaño del filtro es “muy alto”?

3. Ejercicios

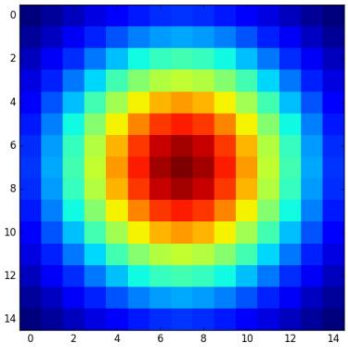
1. Elaborar una tabla de doble entrada con tipos de ruido y tipos de filtros. En cada celda de la tabla, escriba qué tan efectivo es ese filtro para ese ruido. Agregue algunos comentarios o notas sobre el nivel de ruido que se puede considerar, o qué parámetros de filtro pueden ser adecuados, y posibles "efectos secundarios" sobre la imagen (por ejemplo, tal vez un filtro determinado disminuya la cantidad de ruido pero también tenga un impacto negativo en otros). aspectos de la imagen). Finalmente, puede incluir comentarios adicionales que resalten las características clave de cada filtro o comparen entre ellas.

2. Como sabes, el filtro medio es separable. ¿Qué implica esto? Bueno, podemos obtener el mismo resultado aplicando secuencialmente dos filtros 1D (uno sobre filas y otro sobre columnas) que aplicando un filtro 2D, con el beneficio de que el coste de aplicar dos filtros unidimensionales es menor que el de aplicar un único filtro 2D. Escribe el promedio de la versión separable.



(a)gv1d

(b) gv1d como imagen



(c) gv2d como imagen

Figura 1. La Gaussiana discretizada: 1D (a,b) y 2D (c), para n = 15 y σ = 4.

FilterSep(), del filtro medio dado en Filter() promedio.

Compruebe que los resultados sean equivalentes y compare los tiempos de ejecución con tamaños crecientes de imagen y máscara. Traza estos tiempos usando Matplotlib para un análisis visual.

3. En nuestro código, utilizamos directamente la función filtro gaussiano. r() para aplicar el filtro gaussiano. Ahora lo haremos con una convolución explícita, como hicimos con el filtro medio, es decir, creación de máscara + convolución. Procedamos paso a paso:

   (a) Utilice la función gaussiana() de scipy.signal para generar una matriz de tamaño n×1 con los valores de una gaussiana 1D del tamaño n dado y la desviación estándar σ . Sea gv1d este vector. Muestre gv1d con Matplotlib, como si fuera una imagen, sin ninguna interpolación (interpolación='none').

   (b) Genere una matriz n × n con los valores gaussianos 2D (como antes, para n y σ dados). Sea gv2d esta matriz. Puede obtener gv2d mediante un producto matricial de gv1d y su transpuesta, ya sea con gv1d * gv1d.T o np.outer(gv1d, gv1d). Muestra gv2d como una imagen. La figura 1 muestra un ejemplo de gv1d y gv2d usando un mapa de colores.

   (c) Finalmente, use gv2d como máscara (kernel) para convolucionar la imagen a filtrar. Compare el resultado obtenido ahora con el que obtuvimos usando gaussi y filter().

   (d) El filtro gaussiano también es separable, como usted sabe. Por tanto, en aras de la eficiencia, en lugar de aplicar la convolución con la máscara 2D, vamos a aplicarla con dos convoluciones de las máscaras 1D (gv1d y su transpuesta). Compare el resultado con el de la convolución 2D en el paso anterior.

4. Generalice la función addGaussianNoise() para que funcione tanto para imágenes en nivel de grises como en color. Como es habitual, aplica el filtro a cada una de las bandas de color, por separado. Luego, averigüe si esta aplicación del filtro en banda ancha es realmente necesaria y por qué (no).

5. La imagen cociente se obtiene mediante la división en píxeles de una imagen I y su versión borrosa I Gσ , donde Gσ es un núcleo gaussiano 2D con desviación estándar σ. Escribir una función quotientImage(im,sigma) que devuelve la imagen del cociente para I=im con σ =sigma.
   Pruébelo en algunas imágenes y piense en su posible utilidad. ness.

6. (Opcional) Anteriormente, cuando mostramos las imágenes originales y modificadas en la misma figura de Matplotlib, mostramos los parámetros de la operación como parte del título de la figura. Sin embargo, en este y otros casos, es conveniente tener, además de este título general, títulos individuales para cada parte de la figura. Así, esto serviría para mostrar los valores de los parámetros al lado de cada imagen correspondiente, para mayor claridad. Incluya esta posibilidad mediante un parámetro adicional a showInGrid() que consta de una lista de cadenas (una cadena por elemento a mostrar). Por otro lado, descubra cómo podemos mostrar texto que incluya letras griegas y símbolos matemáticos, como “σ = 2” en lugar de “sigma=2”.

7. (Opcional) Arriba te sugerimos hacer zoom en un área dentro de una imagen para percibir mejor el ruido o el efecto de algún filtro. Hacer esto de forma interactiva es soportable si tenemos que repetirlo sólo unas pocas veces, pero se vuelve tedioso cuando se realiza varias veces. ¿ Cómo podemos ser más eficientes? Podemos mostrar directamente (por código) un área de interés de la imagen, en lugar de la imagen completa. Para hacerlo, agregue un parámetro adicional a showInGrid() que representa la región rectangular que queremos mostrar. Pruébelo, digamos, con la nariz de Lena.
   Otra cuestión es: ¿cómo podemos conocer de forma rápida y aproximada las coordenadas de alguna región de interés ? Podemos pasar el mouse sobre una imagen mostrada con Matplotlib y mirar la parte inferior de la ventana, donde se muestran las coordenadas.