Aplicaciones del filtrado

Temario

- Template matching.
- Multi-template matching.
- Pirámides de imágenes.
- Registro de imágenes.
- Eliminación de ruido (denoising).

Template matching

- **Template Matching**. Método para buscar la ubicación de una plantilla (template) en una imagen.
- Ejemplo: buscar  en esta imagen:
- Principal desafío: definir una medida de comparación entre dos imágenes.

Template matching en OpenCV

result = cv2.matchTemplate(image, template, método)

- Métodos en OpenCV (cada uno viene con y sin normalización):
- Suma de diferencias de cuadrados (SSD – Sum Square Difference).
  - cv2.TM_SQDIFF y cv2.TM_SQDIFF_NORMED
- Correlación cruzada (cross-correlation).
  - cv2.TM_CCORR y cv2.TM_CCORR_NORMED
- Coeficiente de correlación (correlation coefficient).
  - cv2.TM_CCOEFF y cv2.TM_CCOEFF_NORMED
- El tutorial de OpenCV recomienda usar SSD o correlación cruzada.

Métodos de comparación SSD

- SSD: $R(x,y) = \sum_{x',y'} \left(T(x',y') - I(x+x', y+y')\right)^2$

Interpretación del resultado

- cv2.matchTemplate() devuelve una imagen en escala de grises.
- Cada pixel indica la similitud entre el template y la región de la imagen de entrada en esa posición específica.
- Los valores dependen del método.
- Para cv2.TM_SQDIFF y cv2.TM_SQDIFF_NORMED, los valores más bajos indican una mejor coincidencia.
- Para cv2.TM_CCORR, cv2.TM_CCORR_NORMED, cv2.TM_CCOEFF y cv2.TM_CCOEFF_NORMED, los valores más altos indican una mayor correlación.

Interpretación del resultado

- La matriz de resultados es un array NumPy de punto flotante de 32 o 64 bits y un solo canal.
- Para encontrar la mejor coincidencia, se puede utilizar la función cv2.minMaxLoc() en la matriz de resultados. Esta función devuelve:

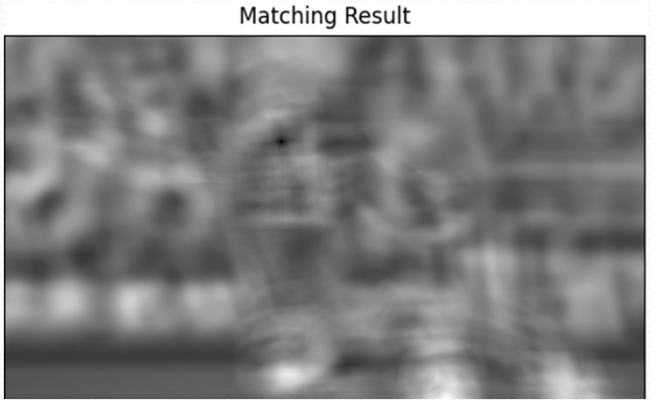
a) min_val. El valor mínimo de la matriz de resultados.
b) max_val. El valor máximo de la matriz de resultados.
c) min_loc. Las coordenadas (x, y) del valor mínimo.
d) max_loc. Las coordenadas (x, y) del valor máximo.

SSD

- Original

SSD

Detectado

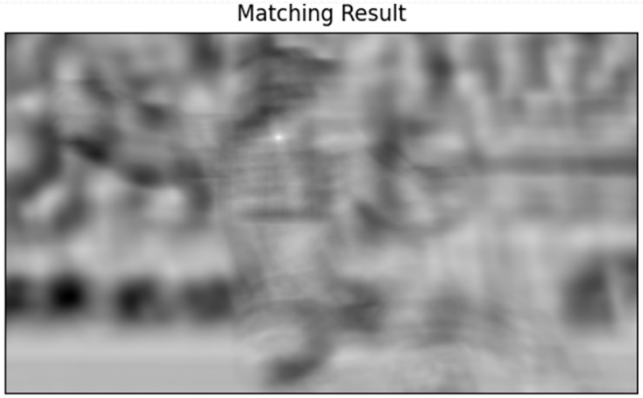


Correlación cruzada normalizada

- Original

CCORR_NORM

Detectado



¿Cuál de los dos es mejor?

- Depende:
- SSD: rápido, sensible a la intensidad global.
- Correlación cruzada normalizada: más lento, invariante a la intensidad y al contraste promedio local.

Variantes

- Pregunta: ¿Y se quiere encontrar todas las ubicaciones de un template en una imagen?
- Respuesta: Se usa multi-template matching.
- Pregunta: ¿Y se quiere encontrar objetos más grandes o más pequeños?
- Respuesta: Se usa una pirámide de imágenes.

Multi-template matching

- Opciones:

1. Usar OpenCV.
2. Usar skimage (scikit-image).

Multi-template Matching con OpenCV

- Algoritmo:

1. Aplicar $\mathrm{cv2.matchTemplate}$ como en el caso simple.
2. Encontrar todas las coordenadas (x, y) en donde el arreglo de resultados es mayor que un umbral.
3. Extraer todas estas regiones.
4. Aplicar supresión no máxima (non-maximum supression – NMS).

Supresión no máxima

- **Supresión no máxima (NMS)**. Permite seleccionar una región entre varias regiones superpuestas.

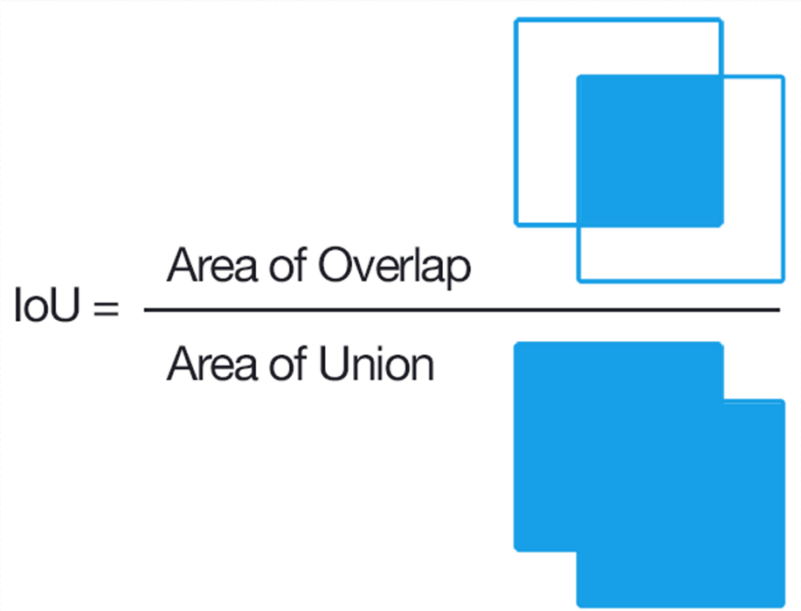
Ejemplo

Fuente: https://learnopencv.com/non-maximum-suppression-theory-and-implementation-in-pytorch/

Criterio de selección

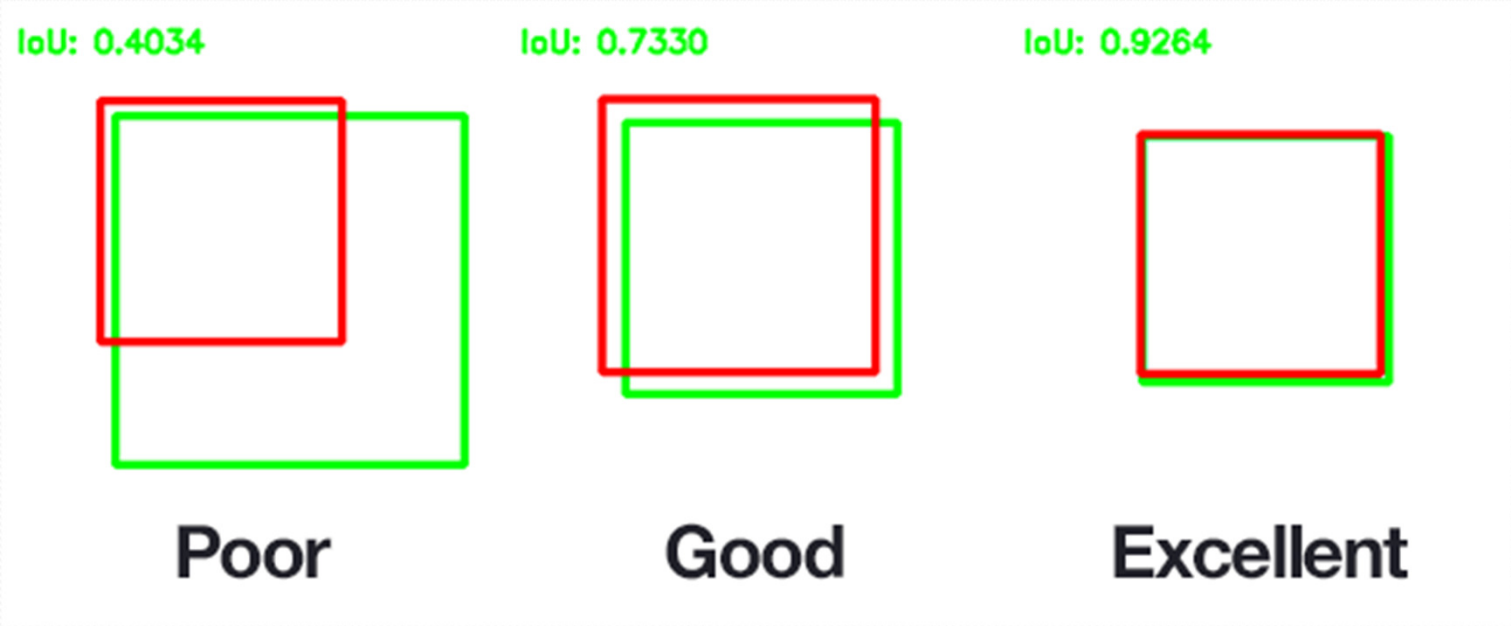
- Necesario para distinguir entre las distintas regiones.
- Intersección sobre unión (intersection over union – IoU).
- Consiste en dividir el área de la intersección entre el área de la unión.

IoU



By Adrian Rosebrock - http://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=57718560

Ejemplos de IoU



Algoritmo general de NMS

- **Entrada:**
- Una lista P de rectángulos de la forma $(x1, y1, x2, y2, c)$, donde $(x1, y1)$ y $(x2, y2)$ son los extremos de los rectángulos y c es un índice de confianza.
- Un umbral de traslape thresh_iou.
- **Salida:**
- Una lista keep de rectángulos.

Algoritmo general de NMS

- **Paso 1:** Seleccionar la predicción S con la puntuación de confianza más alta, eliminarla de P y agregarla a la lista keep. (Inicialmente, keep está vacía).
- **Paso 2:** Comparar esta predicción S con todas las predicciones en P. Calcular el IoU de esta predicción S con todas las demás predicciones en P. Si el IoU es mayor que el umbral thresh_iou para cualquier predicción T presente en P, eliminar T de P.
- **Paso 3**: Si todavía quedan predicciones en P ir al **Paso 1**; de lo contrario, devolver la lista keep que contiene las predicciones filtradas.

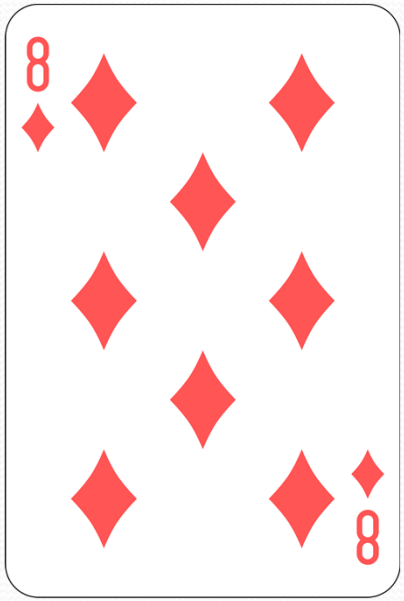
Multi-template matching con skimage

- Algoritmo:

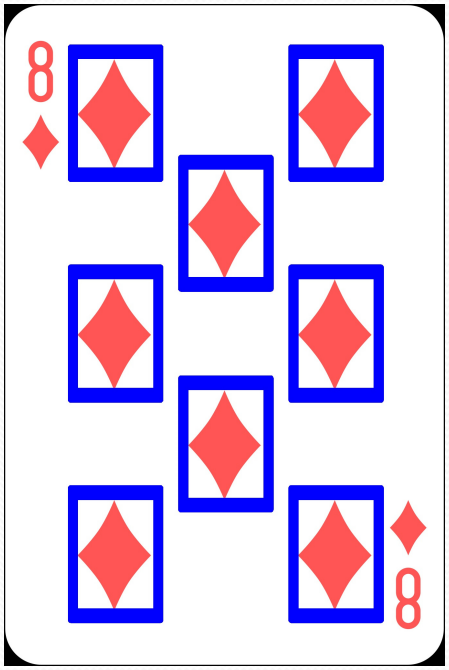
1. Aplicar sk.feature.match_template().
2. El máximo se obtiene con np.unravel_index().
3. Las coincidencias arriba de un umbral se obtienen con sk.feature.peak_local_max().

Ejemplo

- Buscar todos los diamantes dentro de la carta.



Resultado intermedio

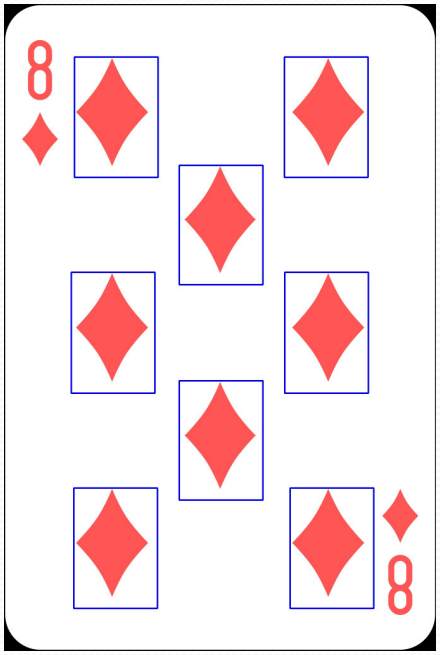


- Salida: **[INFO] 8911 matched locations *before* NMS**

Explicación

- Template Matching encontró 8911 bounding boxes alrededor de los diamantes grandes.
- No detectó los diamantes pequeños.
- Template Matching **no** es invariante al tamaño.

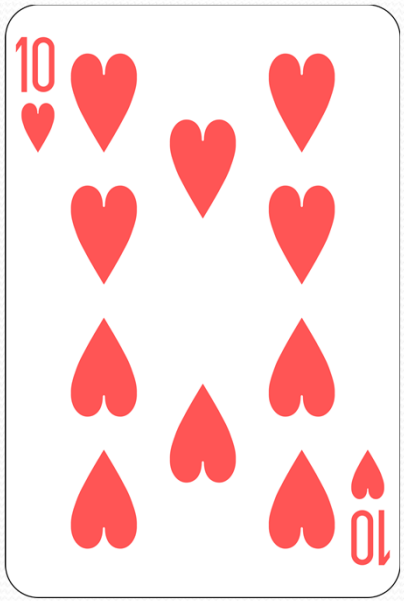
Después de NMS



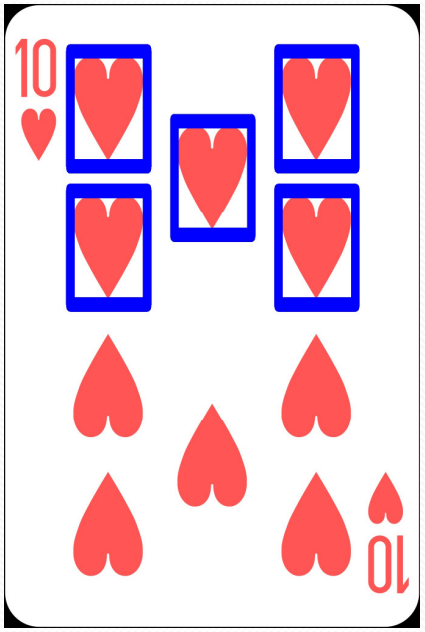
- Salida: **[INFO] 8 matched locations *after* NMS**

Otro ejemplo

- Detectar los corazones en la carta.

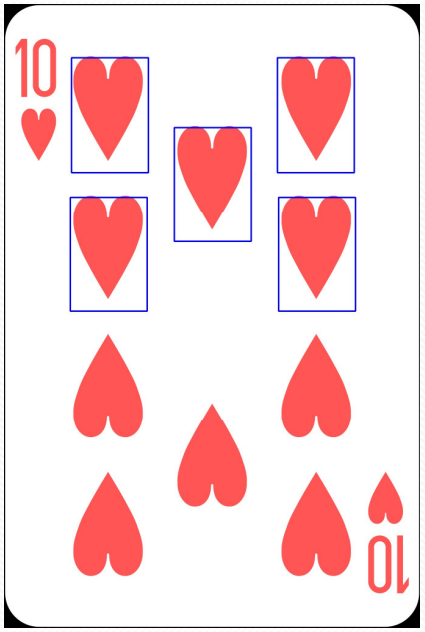


Resultado intermedio



Salida: **[INFO] 4376 matched locations *before* NMS**

Resultado final



Salida: **[INFO] 5 matched locations *after* NMS**

Explicación

- Template Matching no detectó los corazones rotados.
- Template Matching **no** es invariante a la rotación.

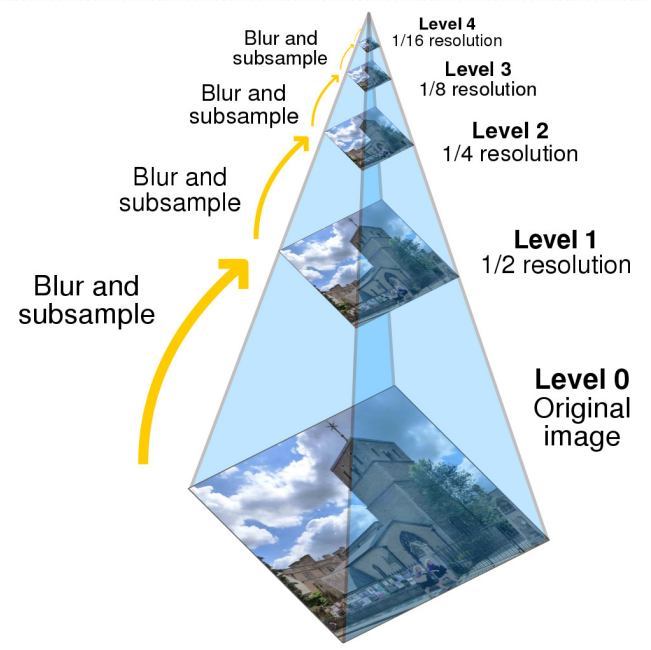
Variantes

- Pregunta: ¿Y se quiere encontrar todas las ubicaciones de un template en una imagen?
- Respuesta: Se usa multi-template matching.
- **Pregunta: ¿Y se quiere encontrar objetos más grandes o más pequeños?**
- **Respuesta: Se usa una pirámide de imágenes.**

Pirámides de imágenes

- **Pirámide de imágenes**. Representación multiescala en la que una imagen está sujeta repetidamente a un proceso de suavizado y subsampling.
- Subsampling significa conservar solo un subconjunto de los pixeles originales y descartar el resto.
- Por ejemplo, hacer subsampling a una imagen por 2, reduce a la mitad su ancho y su alto (1/4 del total de pixeles).

Pirámides de imágenes



By Cmglee - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=42549151

Explicación

- Las imágenes se acumulan a partir de imágenes de alta resolución en la parte inferior e imágenes de baja resolución en la parte superior, formando una pirámide.
- A cada imagen en la pirámide se le llama “octava” (octave).

Tipos de pirámides

- Depende del algoritmo utilizado para suavizar:
- Pirámides de caja.
- Pirámides medianas.
- Pirámides gaussianas.
- Pirámides bilaterales.
- Pirámides laplacianas.
- Pirámides subsampled (sin ningún algoritmo de suavizado).
- Se puede usar cualquier algoritmo de interpolación: vecino más cercano, bilineal, bicúbico, etc.

Tipos de pirámides

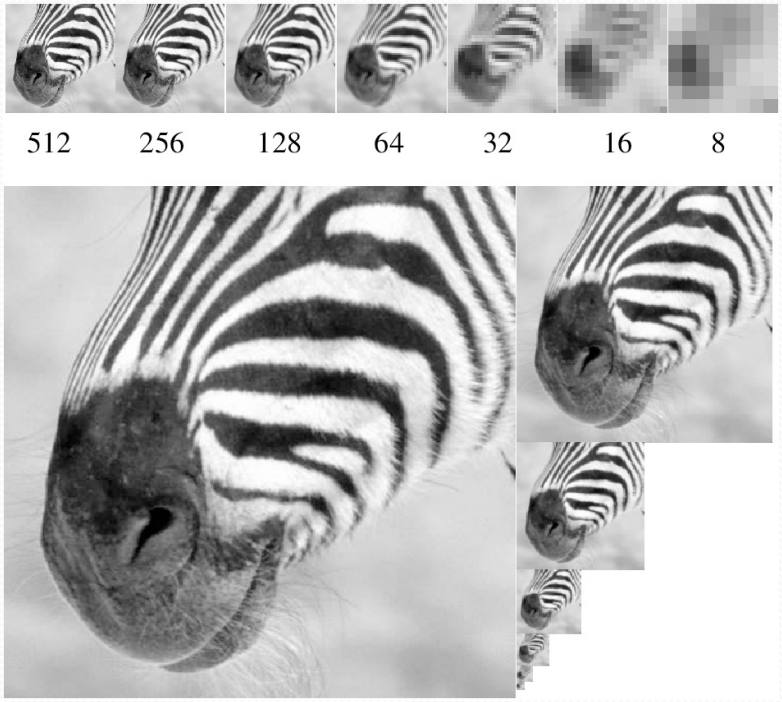
- Dos tipos importantes:

1. Pirámides gaussianas.
2. Pirámides laplacianas.

Pirámide gaussiana

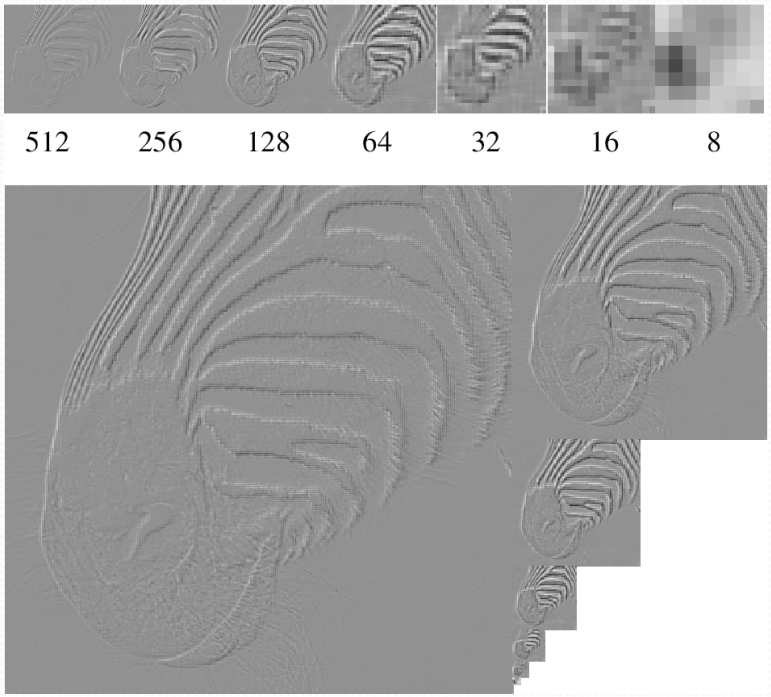
- Aplica repetidamente funciones gaussianas y reduce la resolución de una imagen hasta cumplir algún criterio de detención.
- Por default, la resolución se reduce eliminando cada segundo renglón y columna.
- Un criterio de terminación puede ser un tamaño mínimo de imagen.
- Las funciones cv2.pyrDown() y cv2.pyrUp() permiten subir o bajar de nivel en una pirámide.

Pirámide gaussiana



Fuente: Forsyth, Derek Hoiem (UIUC)

Pirámide laplaciana



Fuente: Forsyth, Derek Hoiem (UIUC)

Gauss y Laplace

- El gaussiano y el laplaciano de una imagen están relacionados.
- $L(I) = \approx G(x, y, \sigma_1) * I - G(x, y, \sigma_2) * I$
- Donde $\sigma_2 > \sigma_1$.
- A esto se le llama diferencia de gaussianos (DoG).

Multi-template matching con pirámides

- Construir la pirámide de la imagen target.
- En cada nivel, correr cv2.matchTemplate() con el template original (sin escalarlo).
- Si hay un match en un nivel reducido, significa que en la imagen original hay un objeto más grande que el template.
- Al final, convertir las coordenadas detectadas en la pirámide a las coordenadas de la imagen original.

Otras opciones

- Si el template y los objetos en la imagen tienen tamaños distintos:

1. Escalado del template:
   - El template se escala, a por ejemplo, al 80%, 100%, 120%, … del tamaño original.
   - Luego se hace template matching normal en cada versión.
   - Así se puede detectar si el objeto aparece más chico o más grande en la imagen target.

Otras opciones

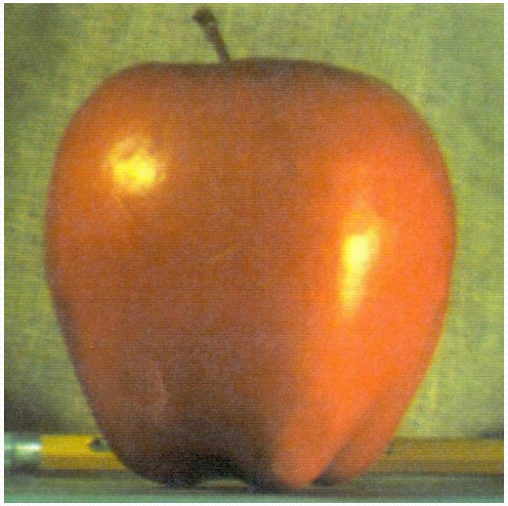
2. Escalado de la imagen target:
   - En vez de cambiar el template, cambiar la imagen target.
   - Se reduce o se amplia la imagen completa.
   - Correr template matching con el template original en cada escala.
   - Esto es útil si se tiene muchos templates pero una sola imagen grande.

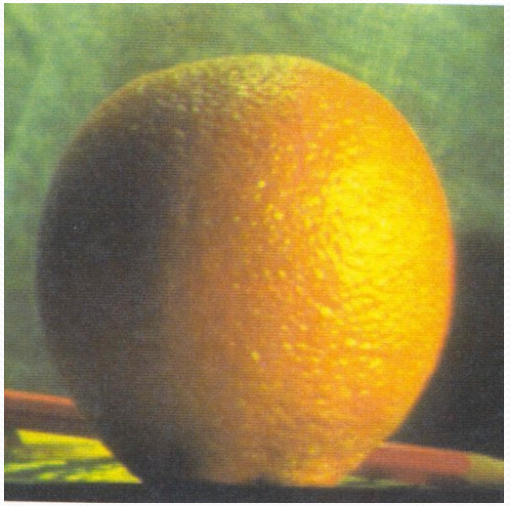
Aplicaciones de la pirámides

- Pegado de imágenes.
- Registro de imágenes.

Pegado de imágenes

- Pegar la parte izquierda de la imagen de la izquierda con la parte derecha de la imagen de la derecha:

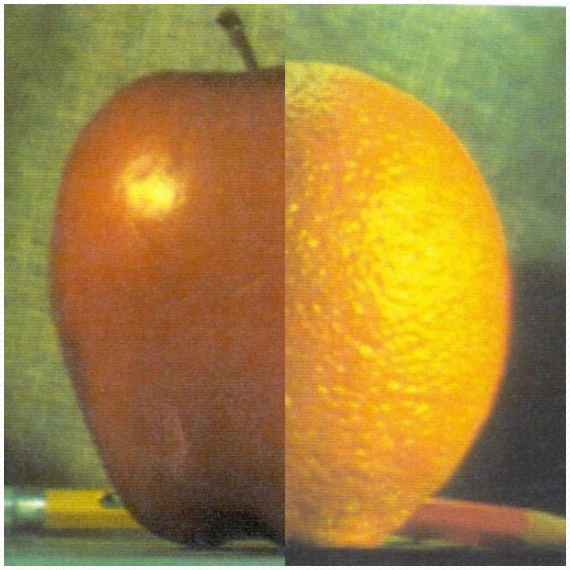




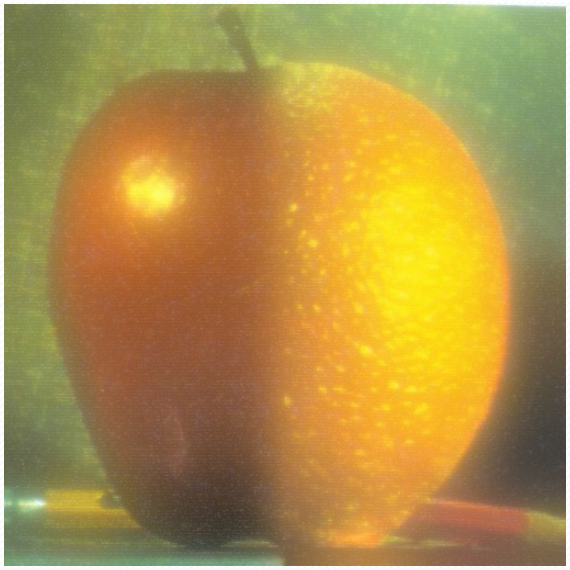
Algoritmo

1. Cargar las dos imágenes.
2. Calcular las pirámides gaussianas (en este ejemplo, el número de niveles es 6).
3. Calcular las pirámides laplacianas.
4. Unir la mitad izquierda de la manzana y la mitad derecha de la naranja en cada nivel de las pirámides laplacianas.
5. Finalmente, reconstruir la imagen original.

Resultado



Pegado directo

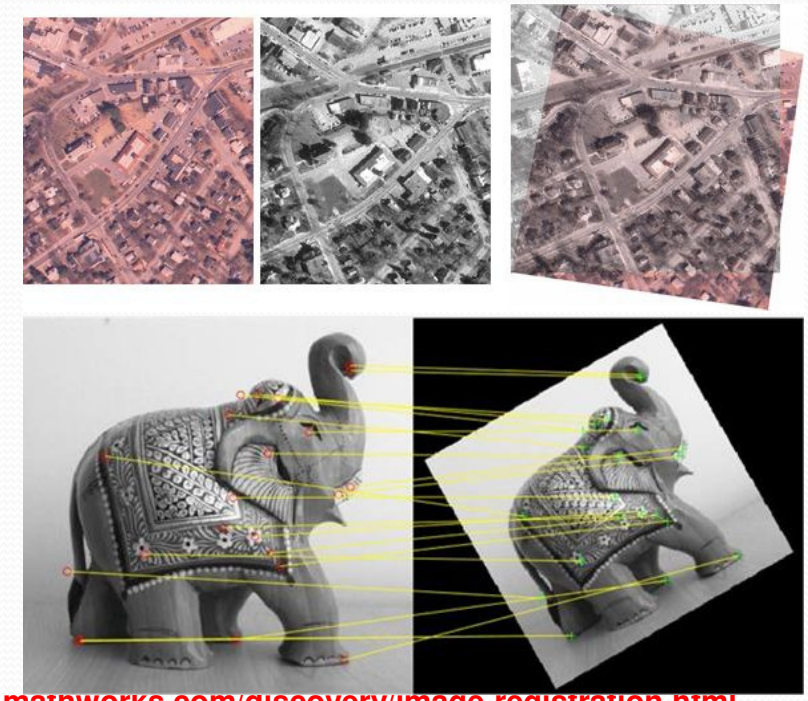


Pegado con pirámides

Registro de imágenes

- Es el proceso de transformar diferentes imágenes en un mismo sistema de coordenadas.
- Pueden ser imágenes de la misma escena tomadas desde distintos puntos de vista, condiciones de iluminación, etc.
- El registro es necesario para poder comparar o integrar las diferentes imágenes.

Ejemplos

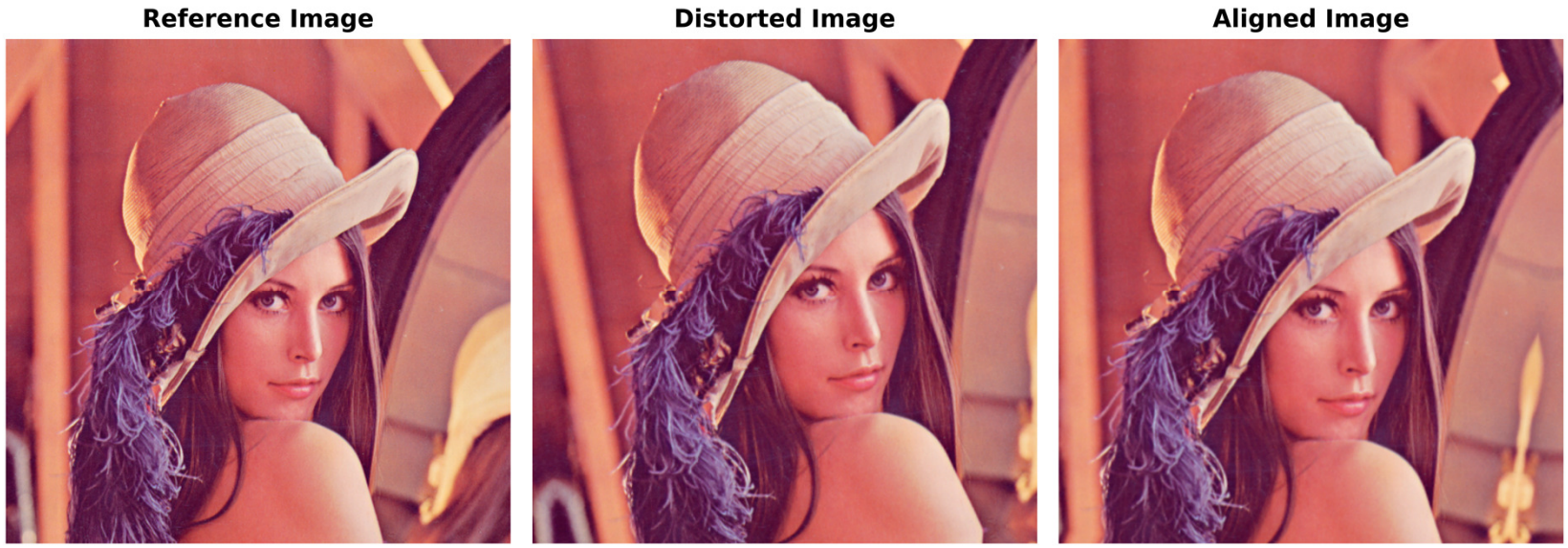


Fuente: https://www.mathworks.com/discovery/image-registration.html

Algoritmo

1. Construir pirámides para las dos imágenes.
2. En el nivel más pequeño, buscar la mejor alineación entre ambas usando correlación.
3. Bajar un nivel en la pirámide.
4. Usar la alineación obtenida arriba como punto de partida y refinar con más detalle.
5. Esto se repite hasta llegar al nivel original.

Ejemplo



Limitantes

- Para que el registro con correlación funcione, ambas imágenes deben ser:

1. La misma escena/objeto.
2. Con solo pequeñas diferencias de posición, escala o rotación (unos pocos grados).
3. Mismo tamaño (o redimensionadas al mismo tamaño).
4. Sin grandes cambios de perspectiva.

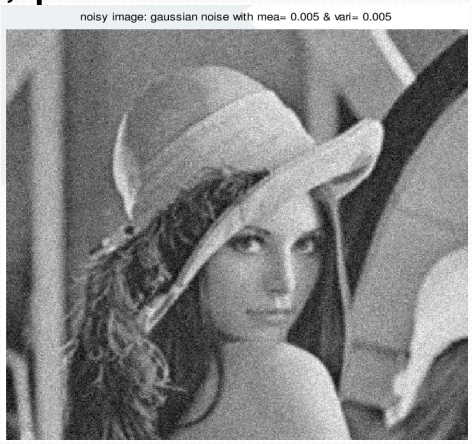
Caso general

- Registro basado en features (SIFT, ORB, AKAZE, etc.)

1. Detectar keypoints y descriptores en ambas imágenes.
2. Empatarlos (match).
3. Estimar una homografía.
4. Utilizar esto como pre-alineación para el refinamiento por correlación.

Denoising

- **Image Denoising**. Consiste en eliminar el ruido de una imagen ruidosa para restaurar una imagen.
- **Ruido**. Modificaciones no deseadas y, en general, desconocidas que una imagen puede sufrir durante la captura, almacenamiento, transmisión, procesamiento o conversión.



Fuente: https://www.researchgate.net/figure/Noisy-image-Gaussian-noise-with-mean-and-variance-0005_fig2_252066070

Denoising con OpenCV

- Filtro de caja: cv2.blur().
- Filtro gaussiano: cv2.GaussianBlur().
- Filtro de mediana: cv2.medianBlur().
- Filtro bilateral: cv2.bilateralFilter().

Comparación de filtros

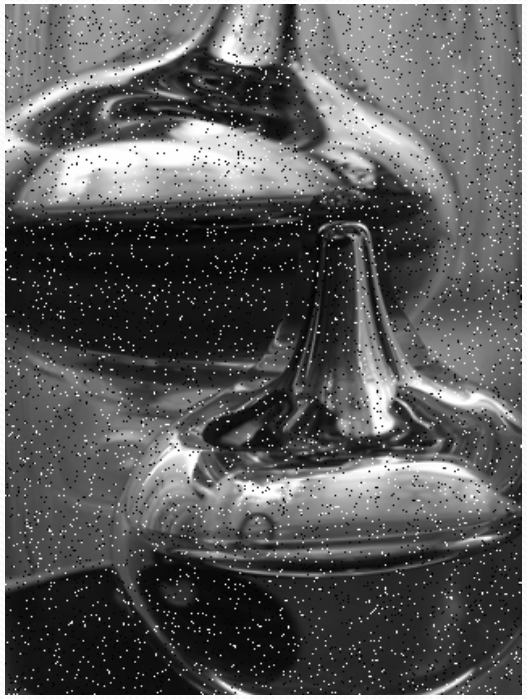


Imagen original

Comparación de filtros

box, Gaussian

median, bilateral

Filtro de mediana

- El filtro de mediana reemplaza cada pixel por la mediana de los pixeles en la región del filtro R, es decir,
- $I'(u,v) = \text{median}_{i,j \in R}\{I(u+i, v+j)\}$

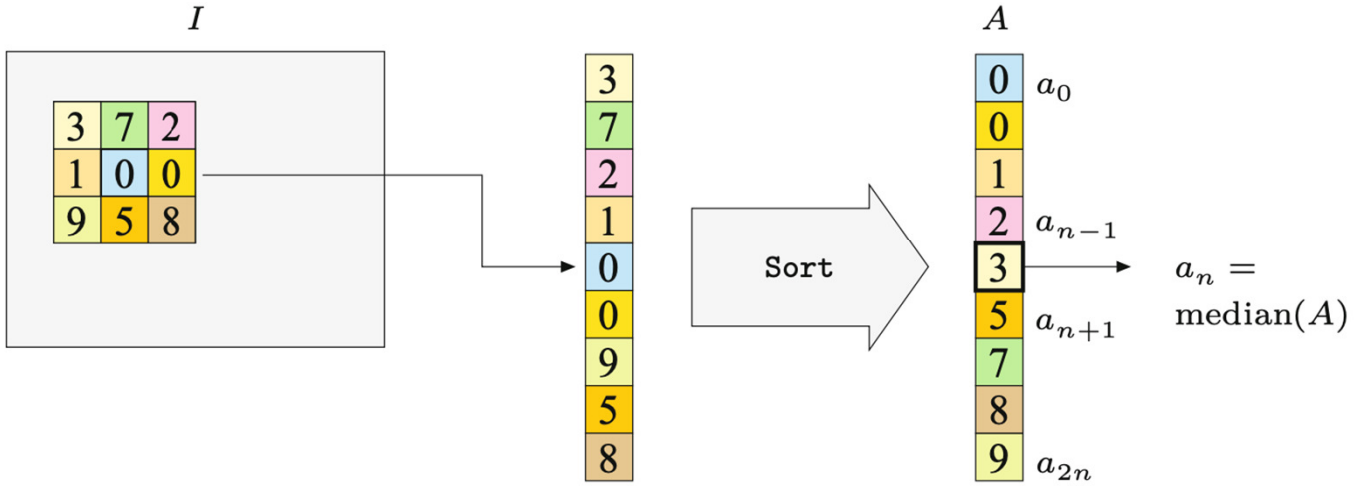
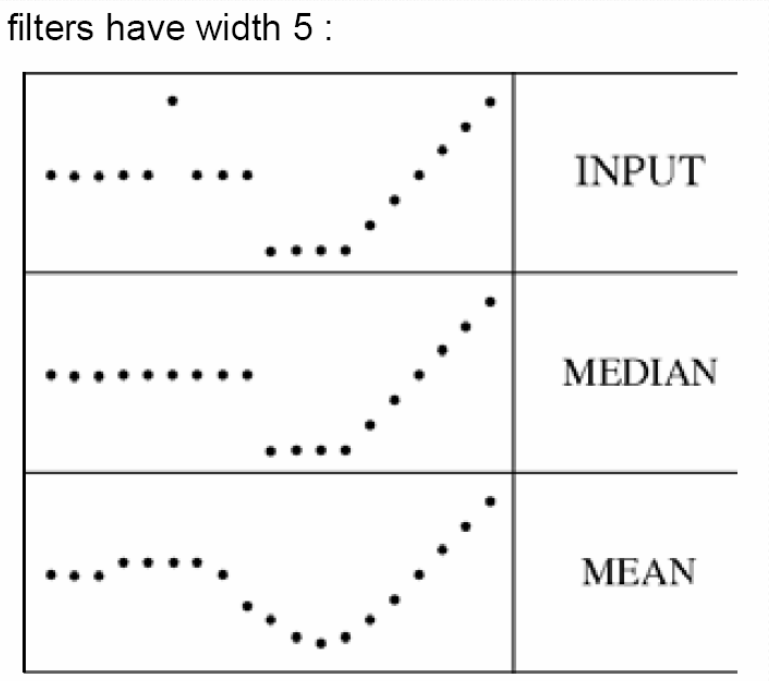


Fig. 5.17 Calculation of the median. The nine pixel values collected from the 3 × 3 image region are arranged as a vector that is subsequently sorted (A). The center value of A is taken as the median.

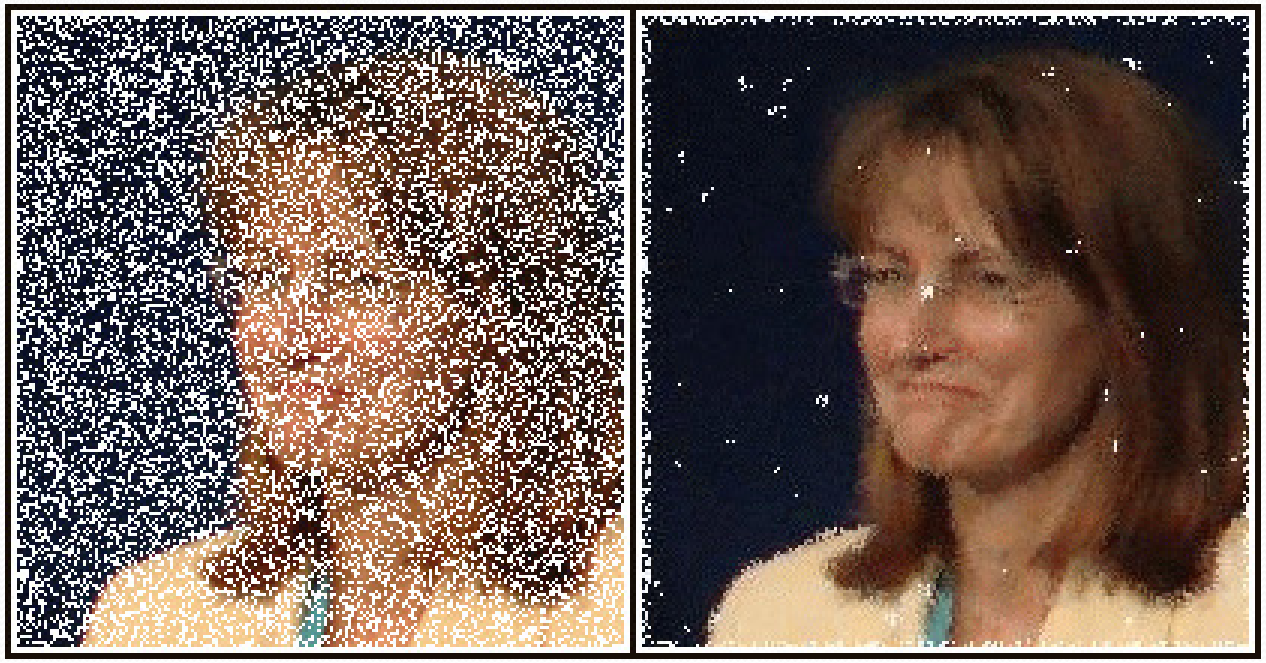
Filtro de mediana

- Ventaja del filtro de mediana: robustez ante valores atípicos (outliers).



Fuente: Derek Hoiem (UIUC)

Filtro de mediana



Fuente: By The original uploader was Anton at German Wikipedia. - Originally from de.wikipedia; description page is/was here., CC BY 2.5, https://commons.wikimedia.org/w/index.php?curid=2544834

Filtro gaussiano vs mediana

Gaussian

Median

Fuente: Derek Hoiem (UIUC)

Conclusiones

- El filtro de mediana está diseñado para el ruido sal y pimienta.
- Funciona bien en estos casos:
- Remover ruido sal y pimienta.
- Imágenes binarias o casi binarias (texto, documentos escaneados, imágenes con umbral).
- No funciona bien en estos casos:
- El ruido es gaussiano.
- La imagen contiene texturas finas: la mediana puede distorsionarlas.
- Se requiere un kernel grande: puede eliminar pequeños detalles y hacer que la imagen con bloques.

Filtro bilateral

- El filtro bilateral reduce el ruido y conserva los bordes.
- Combina dos pesos al calcular el promedio de un pixel con sus vecinos:

1. Espacial: da más importancia a los pixeles cercanos (como un filtro gaussiano).
2. Intensidad: da más importancia a los pixeles con valores de intensidad similares.

Filtro bilateral

cv2.bilateralFilter(src, d, sigmaColor, sigmaSpace)

- Argumentos

1. src – imagen de entrada (8 bits o punto flotante).
2. d – tamaño en pixeles del kernel.
   - Si d > 0, es el tamaño efectivo.
   - Si d = 0, OpenCV lo calcula automáticamente a partir de sigmaSpace.
   - Un valor de d mayor significa que se consideran más pixeles vecinos para el suavizado.

Filtro bilateral

3. sigmaColor
   - Maneja la diferencia de color/intensidad.
   - Un valor grande significa que los pixeles con intensidades muy diferentes se mezclarán (suavizado más intenso en los bordes).
   - Un valor pequeño significa que solo los pixeles de color similar se influyen entre sí (los bordes se conservan mejor).
   - Controla cómo se permite que se mezclen los colores diferentes.

Filtro bilateral

4. sigmaSpace
   - Maneja la distancia.
   - Un valor grande significa que los pixeles más alejados del píxel central pueden influir en el resultado, siempre que sus colores sean similares.
   - Un valor pequeño significa que solo se consideran los pixeles cercanos.
   - Controla el alcance del efecto del color de un pixel en el espacio.

Cómo escoger los parámetros

- No existe una fórmula universal.
- Depende de la imagen y del efecto deseado.
- Algunas consideraciones prácticas:
- d (tamaño de la vecindad)
- Normalmente se elige entre 5 y 15.
- Muy pequeño: poco efecto.
- Muy grande: caro computacionalmente.

Cómo escoger los parámetros

- sigmaColor (filtro de intensidad)
- Valores pequeños (10 a 50): conserva los detalles, menor suavizado.
- Valores altos (75 a 150+): suavizado más intenso, puede hacer borrosos los bordes.
- Regla general: probar valores en torno al rango de intensidad del ruido.

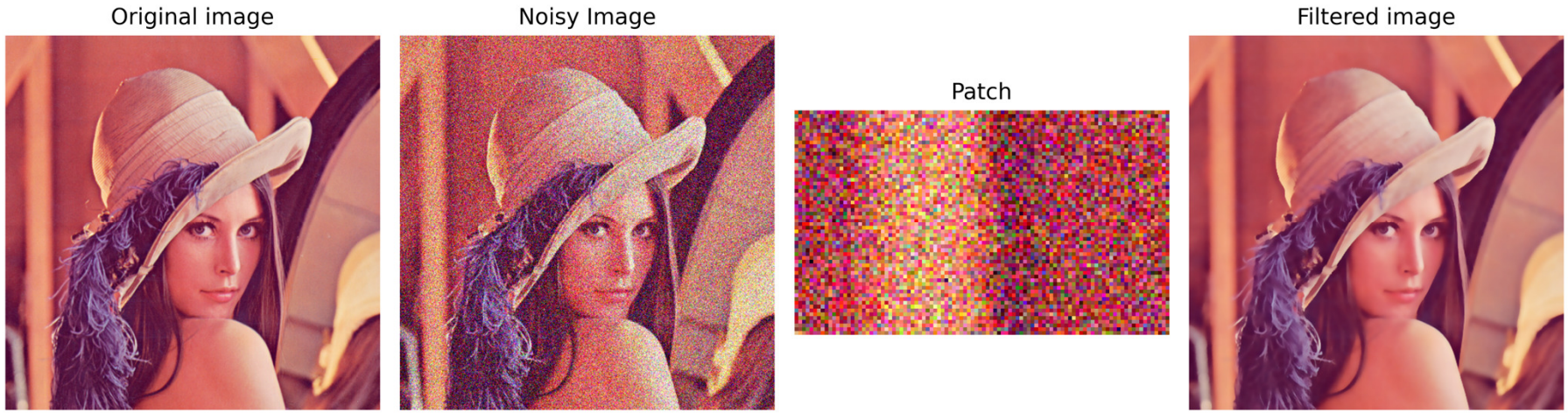
Cómo escoger los parámetros

- sigmaSpace (filtro espacial)
- Valores pequeños (5-15): solo los pixeles muy cercanos influyen en el suavizado.
- Valores altos (30-75+): las áreas grandes se suavizan.
- Regla general: aumentar el valor si se desea suavizar regiones planas grandes.

En resumen

- Si los bordes se ven borrosos: reducir sigmaColor.
- Si el ruido no se reduce lo suficiente: aumentar sigmaColor o sigmaSpace.
- Si el procesamiento es demasiado lento: reducir d.

Filtro bilateral



Filtro bilateral

Imagen original

Imagen filtrada

Conclusiones

- El filtro bilateral está diseñado para una reducción de ruido gaussiano, conservando los bordes.
- Funciona bien en estos casos:
- Fotografías con ruido gaussiano suave o uniforme.
- Reducción de ruido de fotos a color: suaviza las regiones planas (como cielos, paredes, piel) sin difuminar los límites de los objetos.
- Efectos de dibujos animados/estilización: mantiene los bordes nítidos y aplana las texturas.
- Imágenes médicas donde las variaciones sutiles de intensidad son importantes, pero es necesario conservar los bordes.

Conclusiones

- No funciona bien en estos casos:
- El ruido es sal y pimienta: no puede rechazar valores atípicos fuertes.
- El ruido es muy intenso: tiende a difuminar los bordes junto con el ruido.
- La velocidad es crucial: el filtro bilateral es más lento que el filtro de mediana.

Resumen de filtros

- Filtrado en dominio espacial.
  - Deslizar el filtro sobre la imagen y tomar el producto punto en cada posición.
  - Recordar la linealidad (para filtros lineales).
- Filtros lineales para procesamiento básico.
  - Filtro laplaciano (pasa altas).
  - Filtro gaussiano (pasa bajas).

Resumen de filtros

- Filtrado en el dominio de la frecuencia.
  - Puede ser más rápido que filtrar en el dominio espacial (para filtros grandes).
  - Algoritmo:
    1. Aplicar DFT 2D a la imagen y al filtro.
    2. Multiplicación puntual.
    3. Convertir el resultado al dominio espacial con DFT 2D inversa.

Resumen de filtros

- Aplicaciones de filtros.
  - Template matching (SSD o cross-correlation normalizado).
    - SSD se puede hacer con filtros lineales, es sensible a la intensidad general.
  - Pirámide gaussiana.
    - Búsqueda de grueso a fino, detección de múltiples escalas.
  - Pirámide laplaciana.
    - Se puede utilizar para pegar imágenes.
    - Representación de imagen más compacta.

Resumen de filtros

- Aplicaciones de filtros.
  - Subsampling.
    - Se aplica un filtro pasa bajas y luego se eliminan algunos renglones y columnas.
  - Reducir el ruido (importante por motivos estéticos y para procesamientos posteriores como la detección de bordes).
    - Filtro gaussiano, filtro de mediana, filtro bilateral.