Aplicaciones del filtrado

Temario

- Template matching.
- Alineación gruesa a fina.
- Eliminación de ruido (denoising).
- Compresión.

Template matching

- **Template Matching**. Método para buscar y encontrar la ubicación de una plantilla (template) en una imagen.
- Ejemplo: buscar  en esta imagen:
- Principal desafío: ¿Cuál es una buena medida de similitud o distancia entre dos parches?

Template matching

- Métodos en OpenCV:
- Correlación (correlation).
- Correlación cruzada (cross-correlation).
- Suma de diferencias de cuadrados (SSD – Sum Square Difference).
- Cada método viene con y sin normalización.

Métodos de comparación

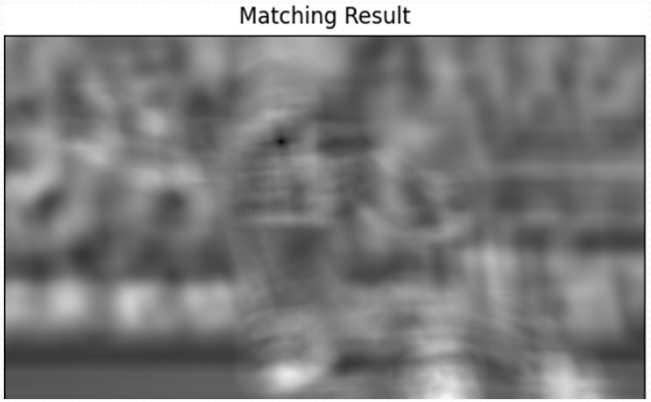
- SSD: $R(x,y) = \sum_{x',y'}\left(T(x',y') - I(x+x',y+y')\right)^2$
- Python: cv2.matchTemplate(im, template, cv2.TM_SQDIFF)
- Correlación cruzada normalizada:
- $R(x,y) = \frac{\sum_{x',y'}\left(T(x',y')\cdot I(x+x',y+y')\right)}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}}$
- Python: cv2.matchTemplate(im, template, cv2.TM_CCORR_NORMED)

SSD

- Original

SSD

Detectado

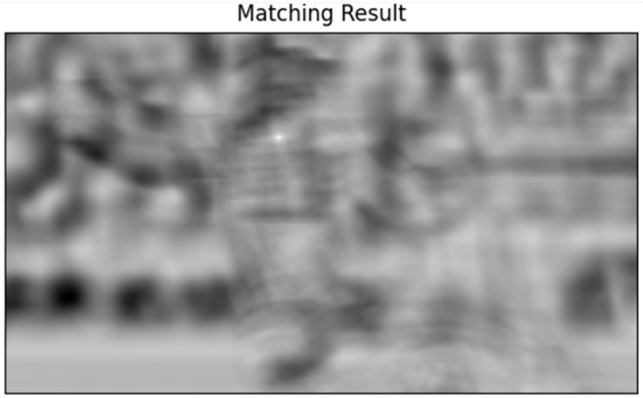


Correlación cruzada normalizada

- Original

CCORR_NORM

Detectado



¿Cuál es el mejor método?

- Depende:
- SSD: rápido, sensible a la intensidad general.
- 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 con OpenCV

- Algoritmo:

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

Supresión no máxima

- **Supresión no máxima (NMS)**. Es una técnica utilizada en numerosas tareas de visión por computadora.
- Permite seleccionar una entidad entre muchas entidades superpuestas.
- Se pueden elegir los criterios de selección para llegar a los resultados deseados.
- Un criterio común es la intersección sobre unión (intersection over union – IoU).

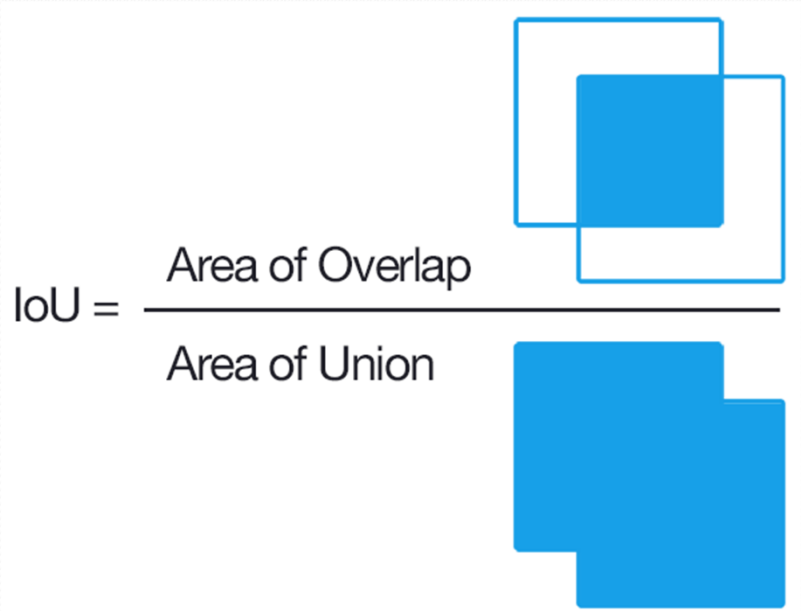
Ejemplo de NMS

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

Intersección sobre unión

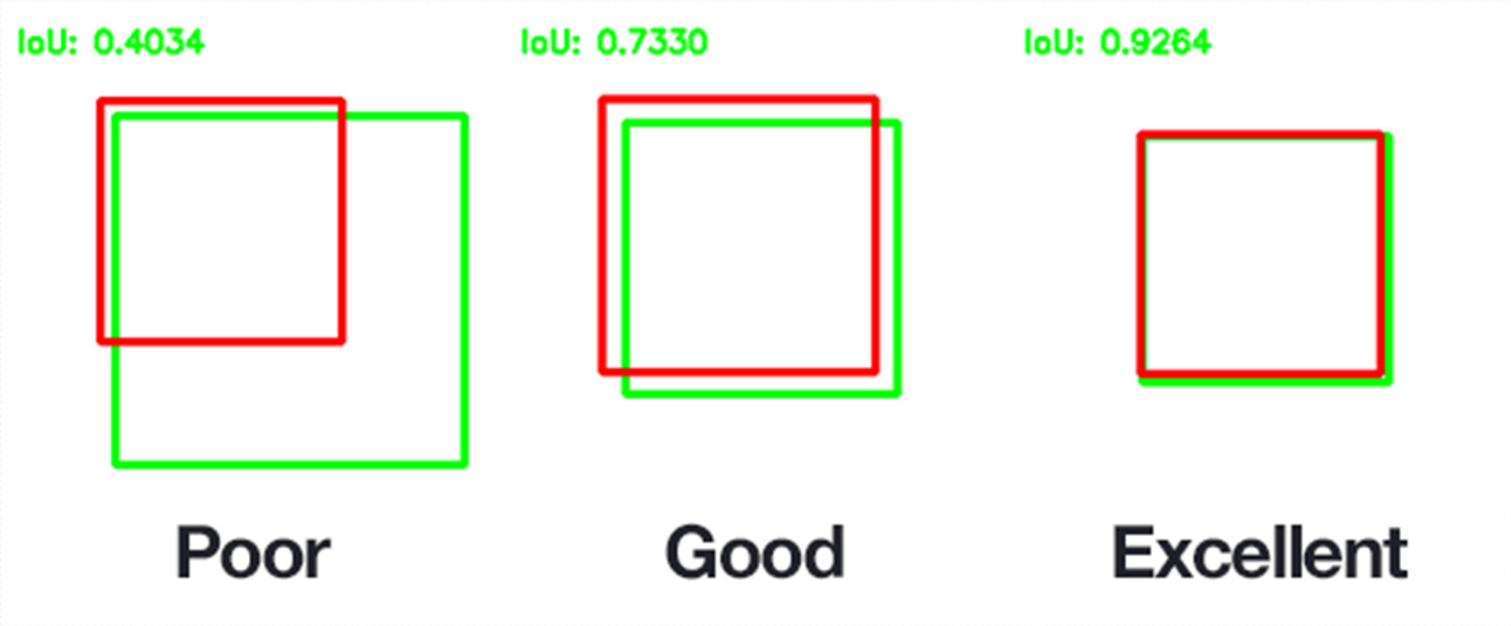
- **Intersección sobre unión (IoU)**. Es una estadística utilizada para medir la similitud de conjuntos de muestras.
- Consiste en tomar la relación entre la intersección y la unión de dos conjuntos.

Cálculo de 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

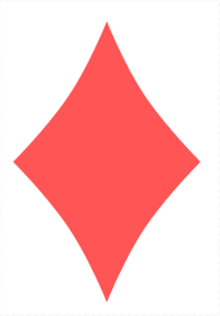
- **Paso 1:** Seleccionar la predicción S con la puntuación de confianza más alta, eliminarla de P y agregarla a la lista keep. (Inicialmente, keep está vacía).
- **Paso 2:** Comparar esta predicción S con todas las predicciones en P. Calcular el IoU de esta predicción S con todas las demás predicciones 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.

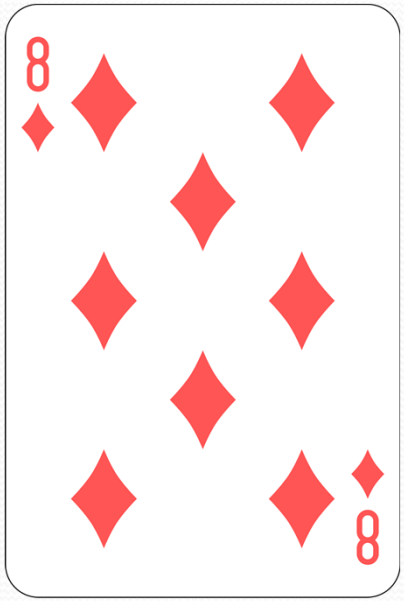
Código

- En el directorio image_processing del código del curso:
- iou.py – implementación de IoU.
- object_detection.py – implementación de NMS.

Ejemplo

- Buscar todos los diamantes dentro de la carta.

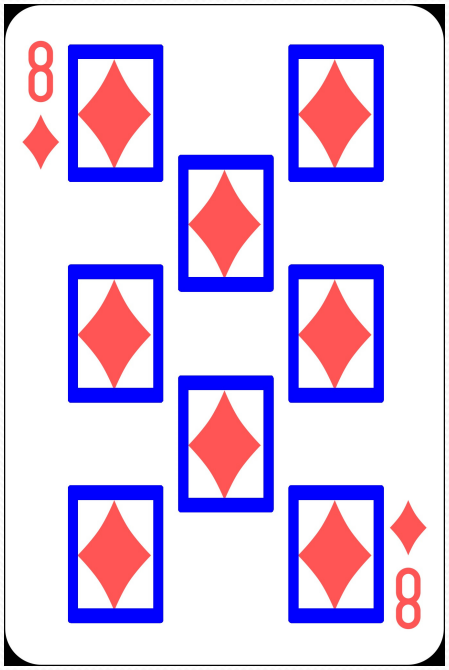




Ejemplo

- Código en /image_processing/template_matching_multiple.py.

Resultado intermedio

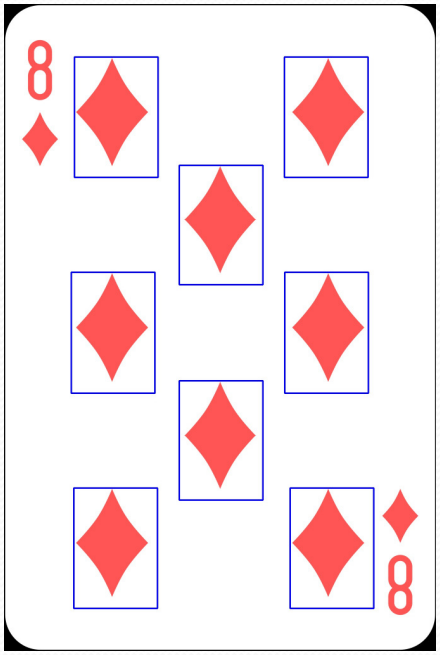


- Salida: **[INFO] 8911 matched locations *before* NMS**

Explicación

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

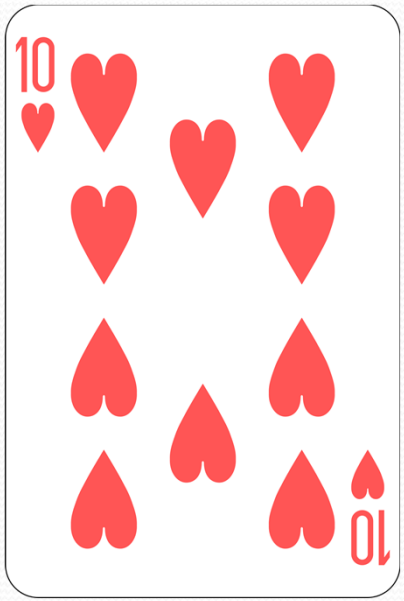
Después de NMS



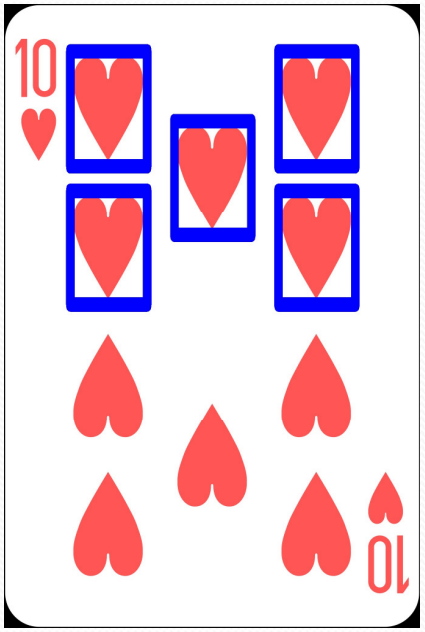
- Salida: **[INFO] 8 matched locations *after* NMS**

Otro ejemplo

- Detectar los corazones en la carta.

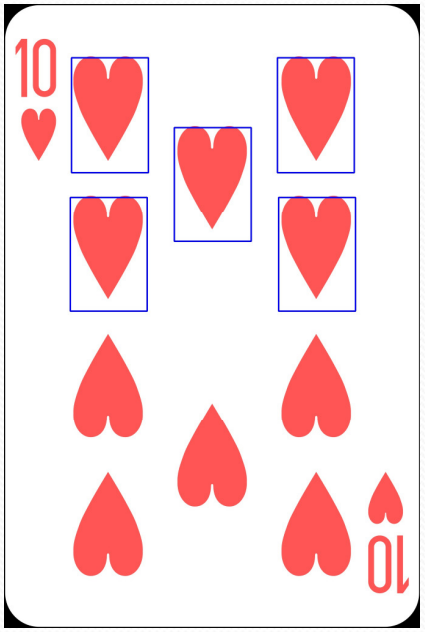


Resultado intermedio



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

Resultado final



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

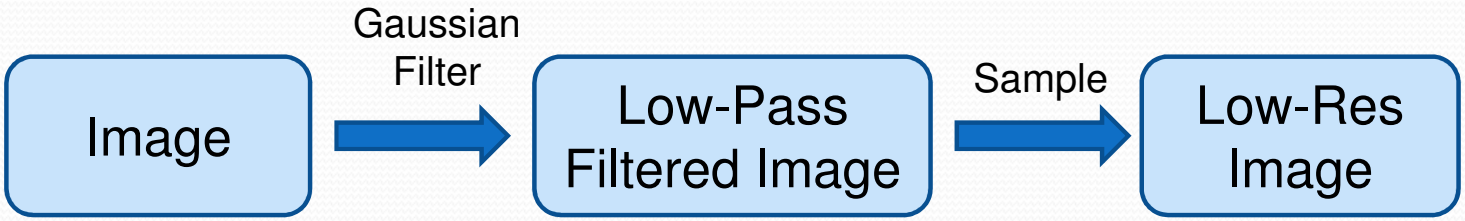
Explicación

- 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.**

Sampling

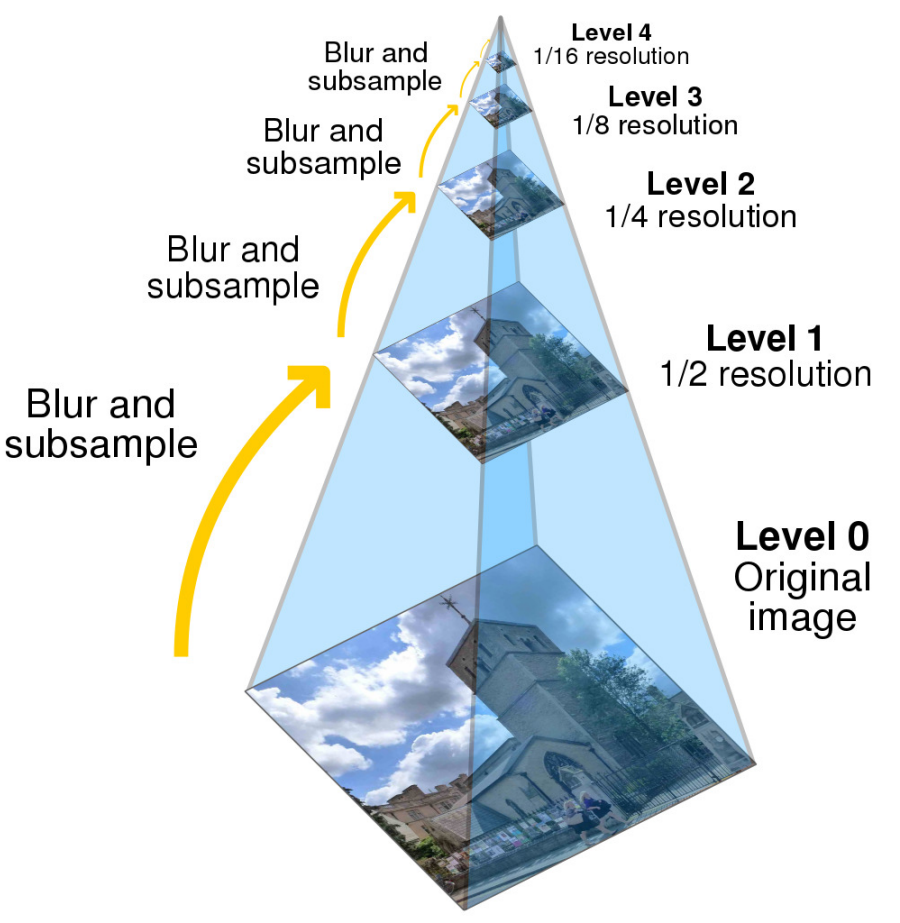


Fuente: Derek Hoiem (UIUC)

Definición

- **Pirámide de imágenes**. Es una representación multiescala en la que una imagen está sujeta repetidamente a un proceso de suavizado y subsampling.

Definición



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 submuestreadas (si no se aplica ningún algoritmo).
- Se puede usar cualquier algoritmo de interpolación: vecino más cercano, bilineal, bicúbico, etc.

Tipos de pirámides

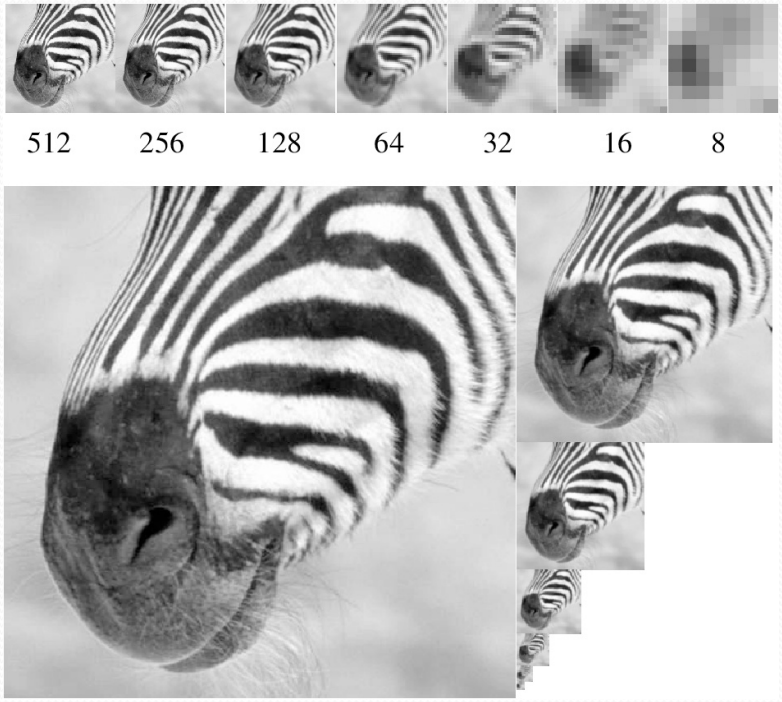
- En este curso se verán dos tipos:

1. Pirámides gaussianas.
2. Pirámides laplacianas.

Pirámide gaussiana

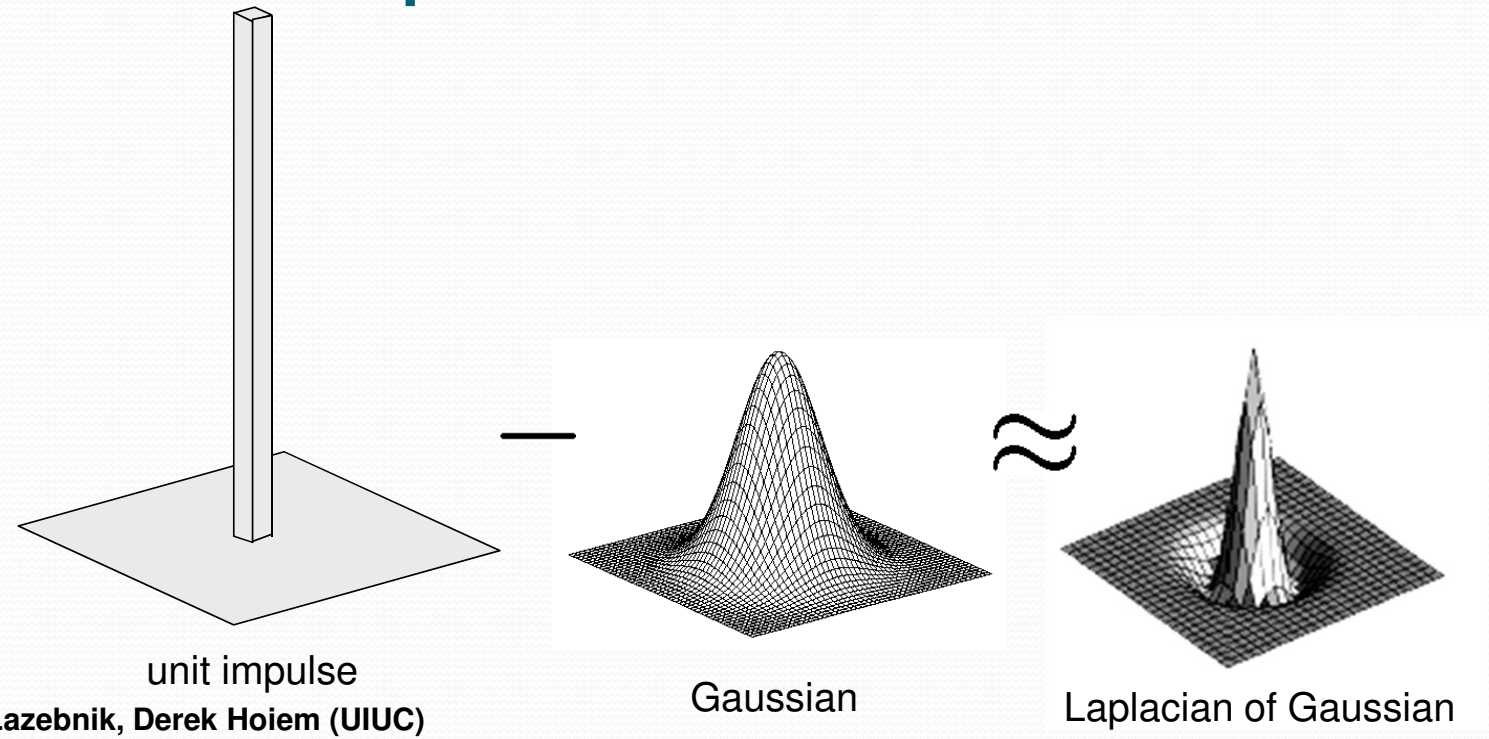
- La pirámide gaussiana aplica repetidamente funciones gaussianas y reduce la resolución de una imagen hasta cumplir algún criterio de detención.
- Por default, la resolución se reduce eliminando cada segundo renglón y columna.
- Un criterio de terminación puede ser un tamaño mínimo de imagen.
- OpenCV proporciona las funciones `pyrDown` y `pyrUp` para subir o bajar de nivel en una pirámide.

Pirámide gaussiana



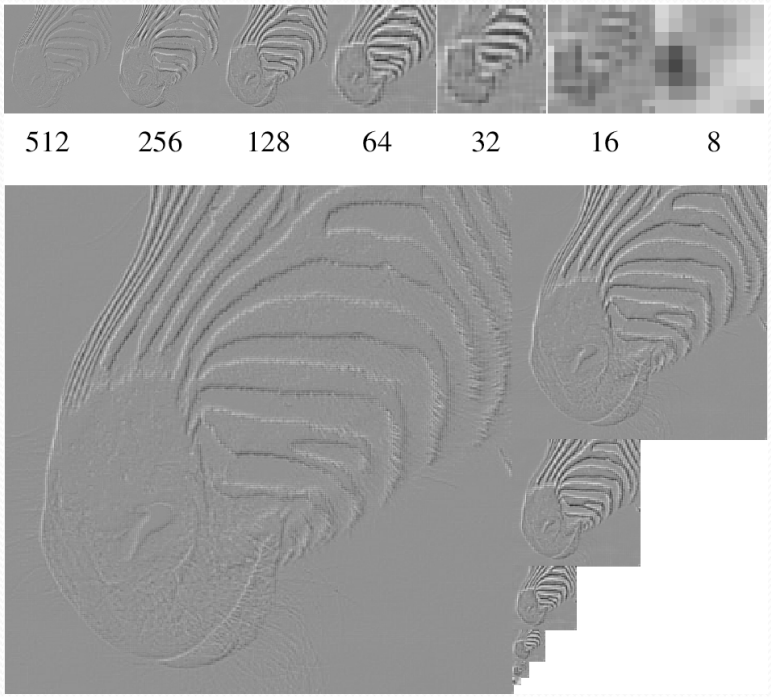
Fuente: Forsyth, Derek Hoiem (UIUC)

Pirámide laplaciana



Fuente: Lazebnik, Derek Hoiem (UIUC)

Pirámide laplaciana

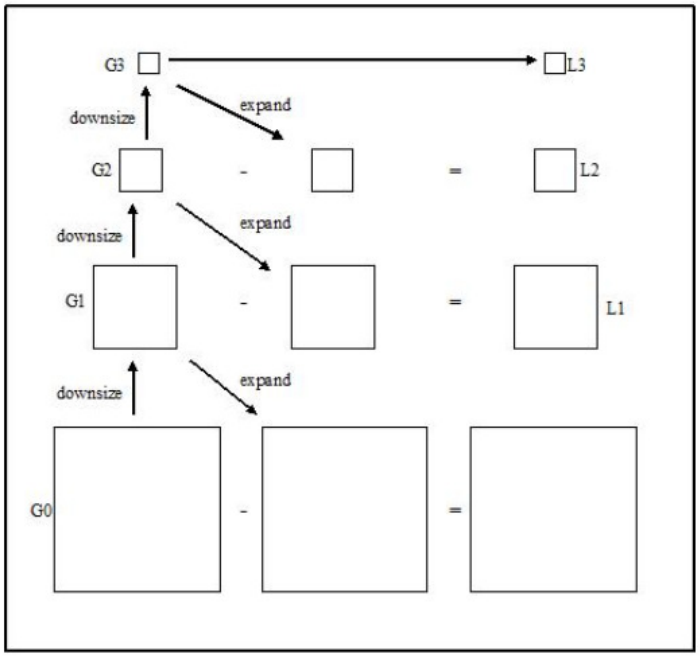


Fuente: Forsyth, Derek Hoiem (UIUC)

Pirámide laplaciana

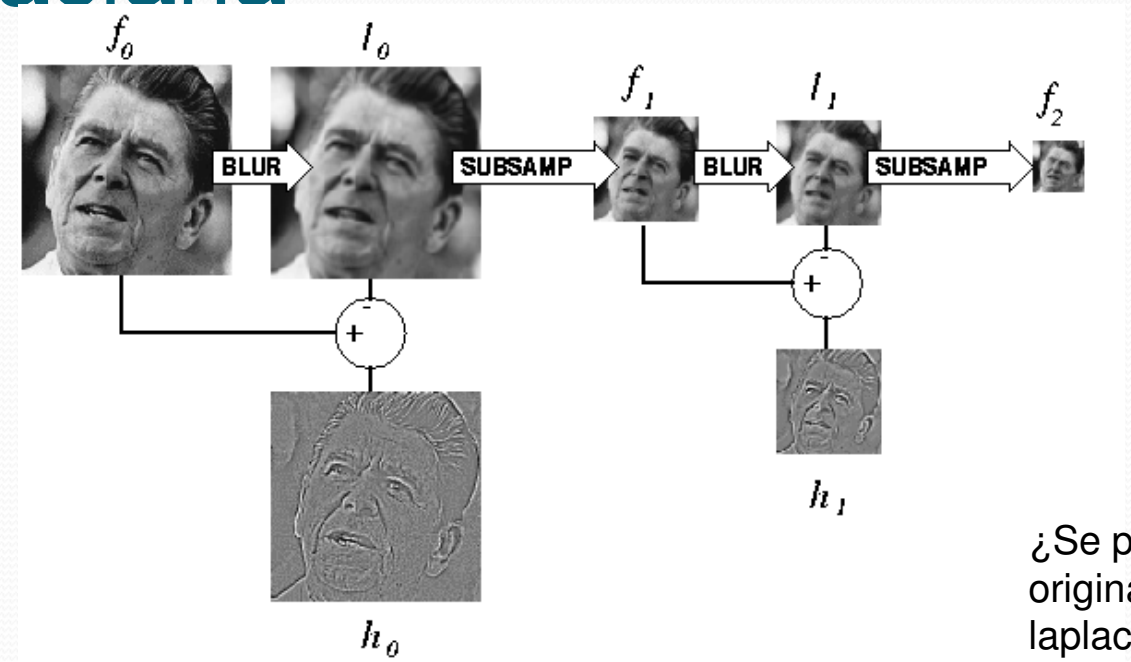
- El laplaciano se puede aproximar utilizando la diferencia de funciones Gaussianas.
- La pirámide laplaciana se obtiene restando los niveles de la pirámide gaussiana.
- El laplaciano de un nivel se obtiene restando ese nivel en la pirámide gaussiana y la versión ampliada de su nivel superior.

Pirámide laplaciana



Fuente: https://theailearner.com/2019/08/19/image-pyramids/

Calculando una pirámide gaussiana / laplaciana



¿Se puede reconstruir el original a partir de la pirámide laplaciana?

Fuente: Derek Hoiem (UIUC)

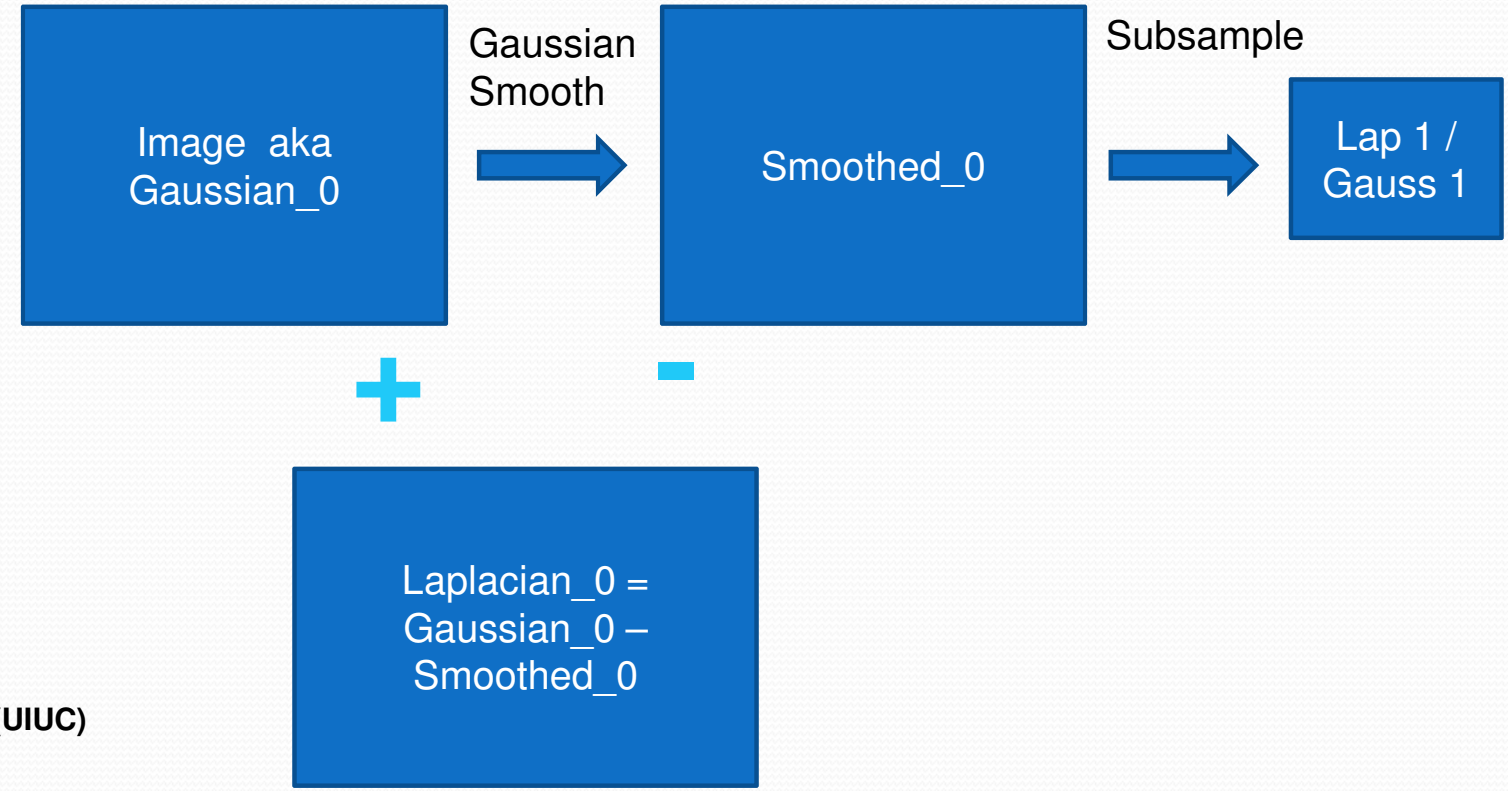
Respuesta

- La imagen reconstruida es la imagen original.
- En cada nivel, donde la pirámide gaussiana (filtro de paso bajo) elimina las frecuencias altas, la pirámide laplaciana almacena estas frecuencias altas.
- Una vez que se tiene una imagen borrosa en cualquier nivel de la pirámide gaussiana y la pirámide laplaciana correspondiente, se puede reconstruir la imagen original mediante la adición sucesiva de componentes de mayor frecuencia en diferentes niveles.

Respuesta

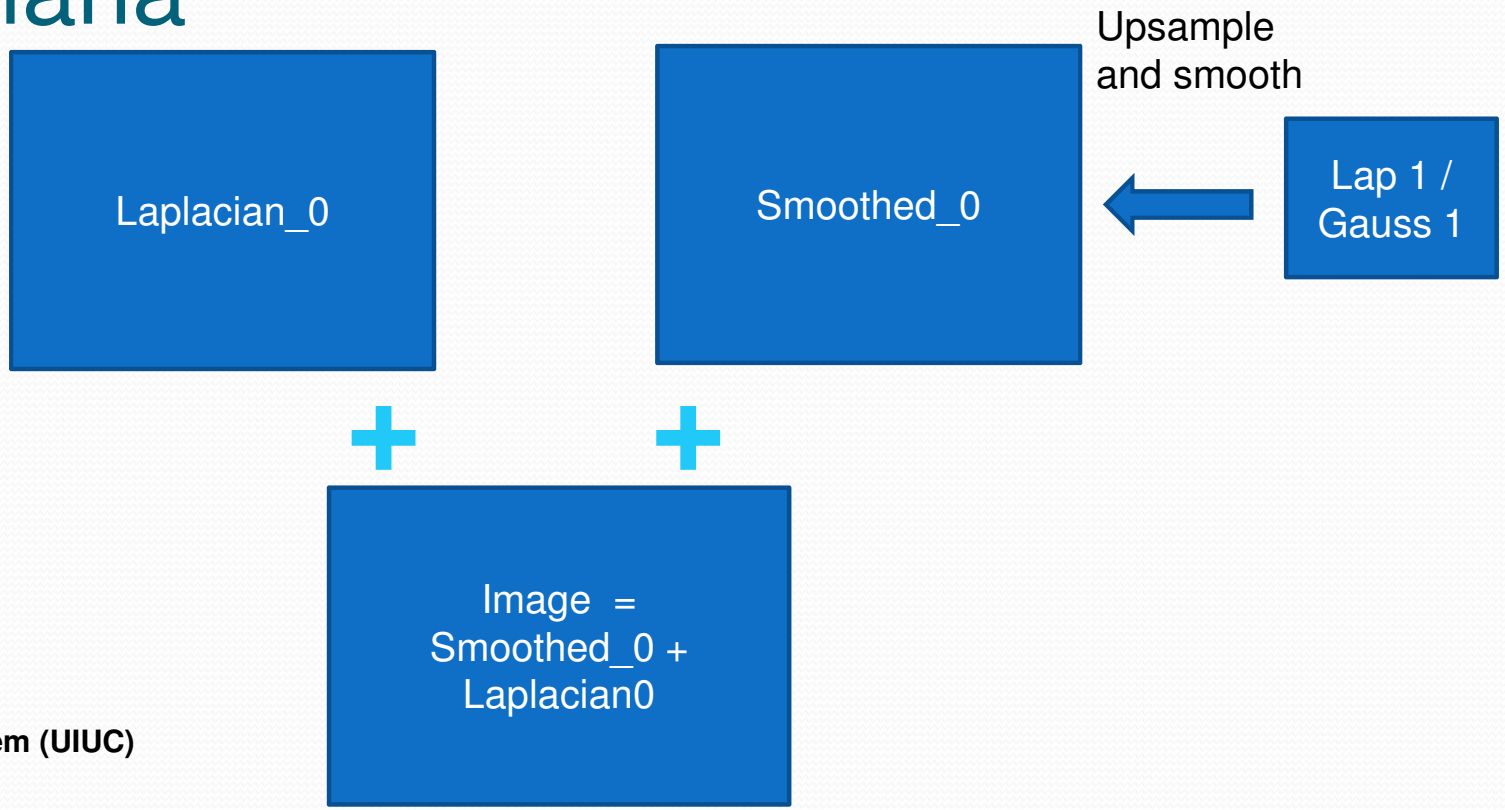
- Debido a esta propiedad de las pirámides laplacianas, se utilizan para la **compresión de imágenes**.
- Ver ejemplo completo con código en Python en: https://notebook.community/darshanbagul/ComputerVision/LaplacianPyramid/LaplacianPyramid

Crear una pirámide Laplaciana de dos niveles



Fuente: Derek Hoiem (UIUC)

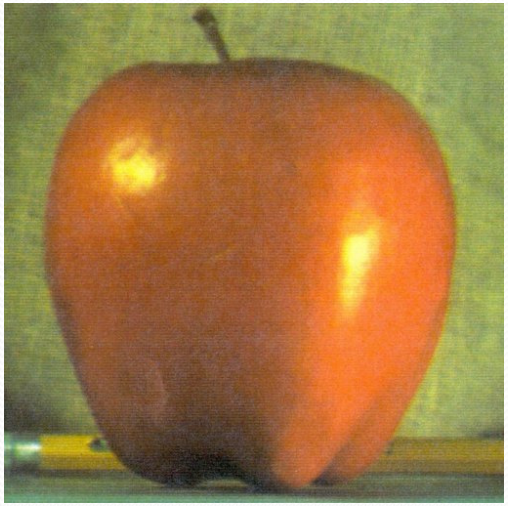
Reconstruir la imagen desde la pirámide Laplaciana

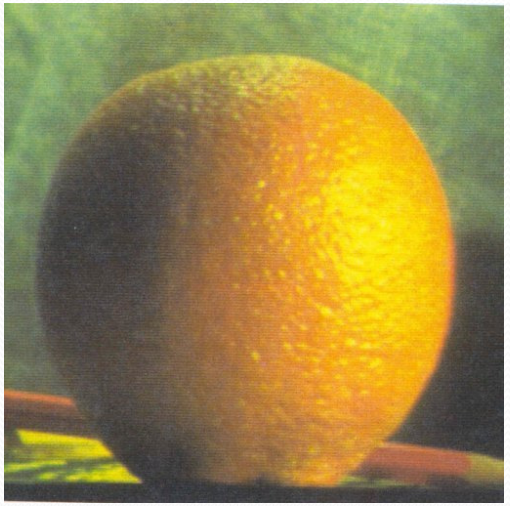


Fuente: Derek Hoiem (UIUC)

Ejemplo: pegado de imágenes

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



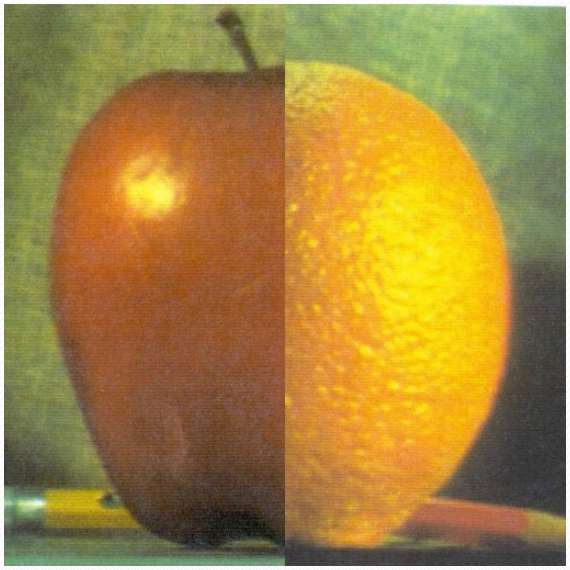


Algoritmo

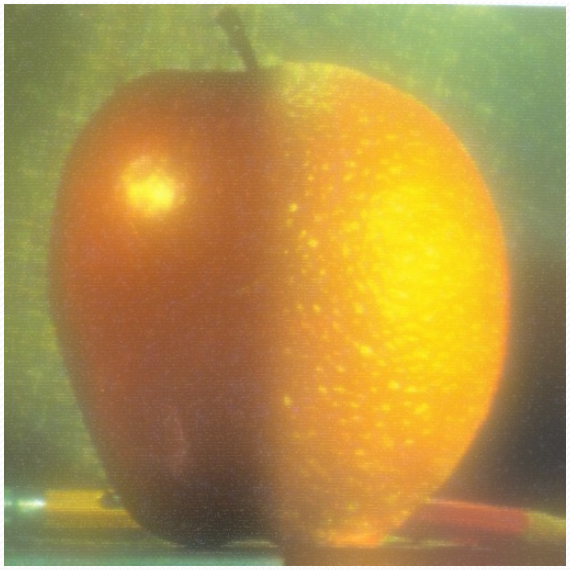
- Ver https://docs.opencv.org/4.x/dc/dff/tutorial_py_pyramids.html

1. Cargar las dos imágenes.
2. Encontrar las pirámides gaussianas para las imágenes (en este ejemplo, el número de niveles es 6)
3. Con las pirámides gaussianas, encontrar 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, a partir de esta imagen conjunta de las pirámides, reconstruir la imagen original.

Resultado



Pegado directo



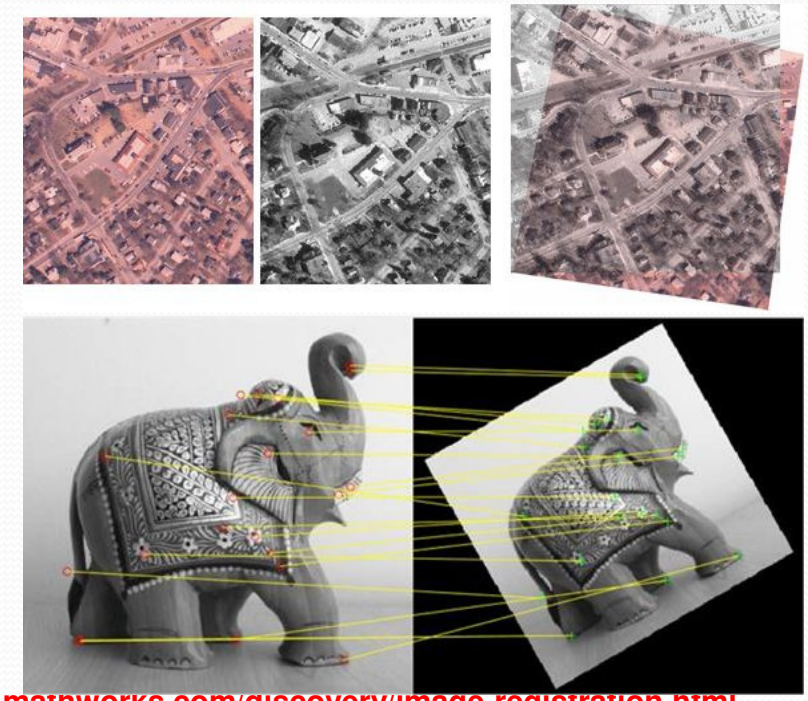
Pegado gaussiano / laplaciano

- Ver /image_processing/pyramid_blend.py en el código del curso.

Registro de imágenes

- **Registro de imágenes**. Es el proceso de transformar diferentes conjuntos de 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.

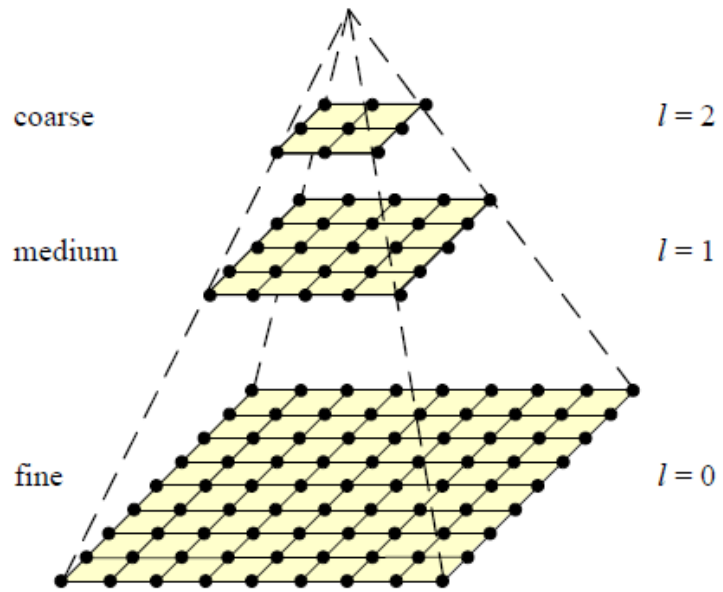
Ejemplos



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

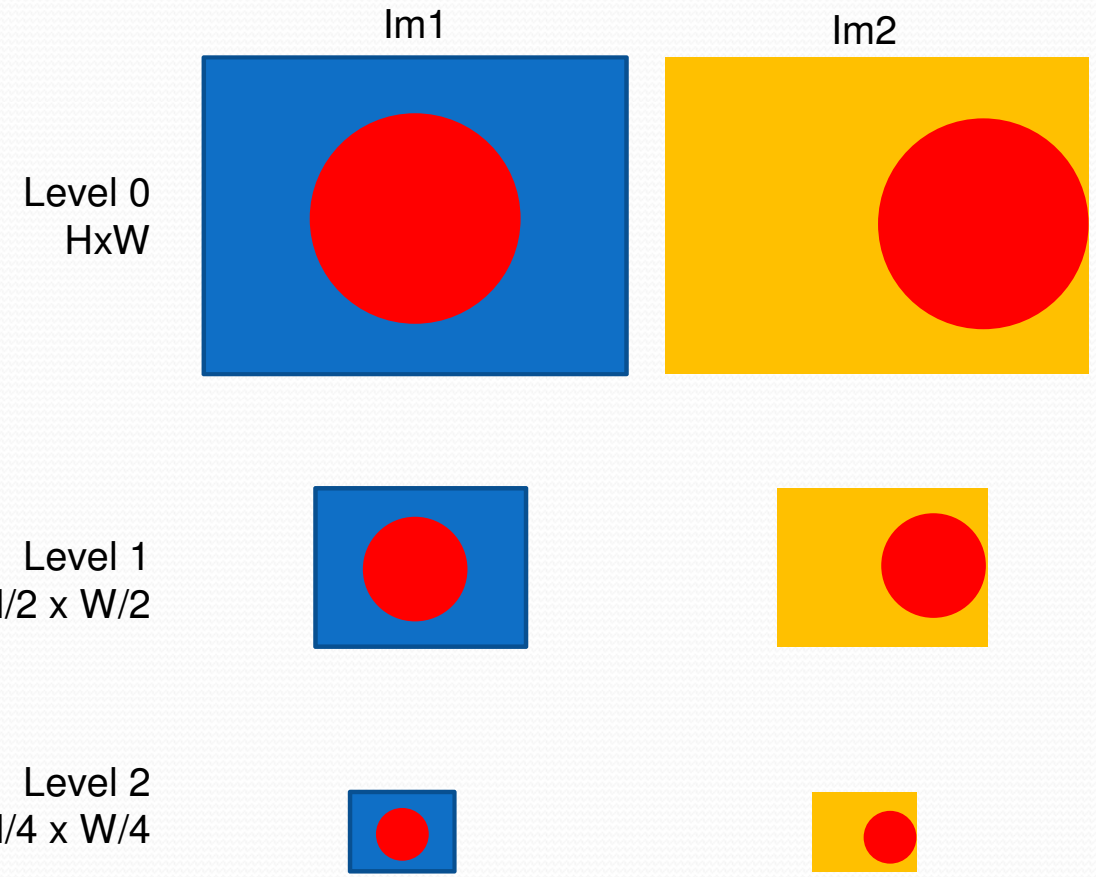
Registro de imágenes grueso a fino

1. Calcular la pirámide gaussiana.
2. Alinear con la pirámide gruesa.
   - Encontrar la posición mínima de SSD.
3. Alinear sucesivamente con las pirámides más finas.
   - Buscar un rango pequeño (por ejemplo, 5x5) centrado alrededor de la posición determinada en la escala más gruesa.



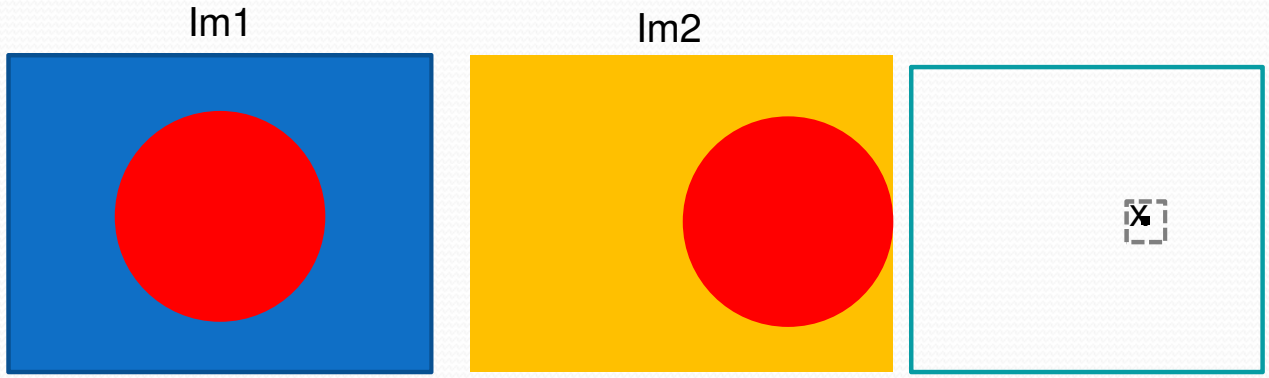
Fuente: Derek Hoiem (UIUC)

Ejemplo



Fuente: Derek Hoiem (UIUC)

Ejemplo



$$tx_0, ty_0 = \underset{tx \in 2 \cdot tx_1 + \{-s..s\}, ty = 2 \cdot ty_1 + \{-s..s\}}{\operatorname{argmin}} \operatorname{SSD}(im1_1, \operatorname{translate}(im2_1, tx, ty))$$

$$tx_1, ty_1 = \underset{tx \in 2 \cdot tx_2 + \{-s..s\}, ty = 2 \cdot ty_2 + \{-s..s\}}{\operatorname{argmin}} \operatorname{SSD}(im1_1, \operatorname{translate}(im2_1, tx, ty))$$

$$tx_2, ty_2 = \underset{tx \in \left\{-\frac{W}{8}..\frac{W}{8}\right\}, ty \in \{-\frac{H}{8}..\frac{H}{8}\}}{\operatorname{argmin}} \operatorname{SSD}(im1_2, \operatorname{translate}(im2_2, tx, ty))$$

Fuente: Derek Hoiem (UIUC)

Compresión de imágenes

- Compresión con pérdida (lossy). Ejemplo JPG.
- Compresión sin pérdida (lossless). Ejemplo PNG.

Ejemplo JPEG

1. Los colores en la imagen se convierte de RGB a Y′CbCr, que consta de un componente luma (Y'), que representa el brillo, y dos componentes croma (Cb y Cr), que representan el color.
2. La resolución de los datos cromáticos se reduce, normalmente en un factor de 2 o 3.
3. La imagen se divide en bloques de 8 × 8 pixeles y, para cada bloque, cada uno de los datos Y', Cb y Cr se somete a la transformada de coseno discreto (DCT).

Ejemplo JPEG

4. Las amplitudes de los componentes de frecuencia se cuantifican.
5. Los datos resultantes para todos los bloques de 8×8 se comprimen aún más con un algoritmo sin pérdidas, una variante de la codificación Huffman.

- El proceso de decodificación invierte estos pasos, excepto la cuantificación porque es irreversible.
- Ver detalles en https://en.wikipedia.org/wiki/JPEG#JPEG_codec_example

Ejemplo PNG

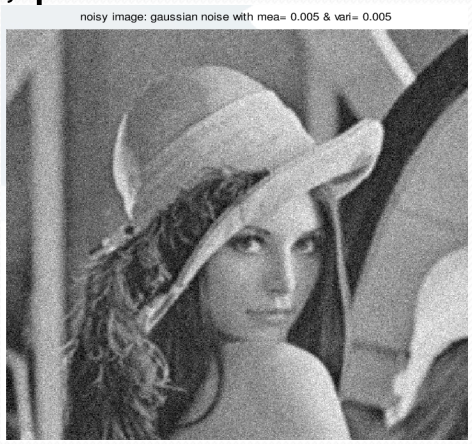
- PNG utiliza un proceso de compresión de 2 etapas:

1. Pre-compresión (filtrado). El filtro predice el valor de cada pixel basándose en los valores de los pixeles vecinos anteriores y resta el color previsto del pixel del valor real
2. Compresión. Utiliza DEFLATE, un algoritmo de compresión de datos sin pérdidas no patentado.

- Ver detalles en https://en.wikipedia.org/wiki/PNG#Compression

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 filtro de caja

- Python: cv2.blur().

Denoising con filtro gaussiano

- Python: cv2.GaussianBlur().

Denoising con filtro de mediana

- Python: cv2.medianBlur().

Filtro de mediana

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

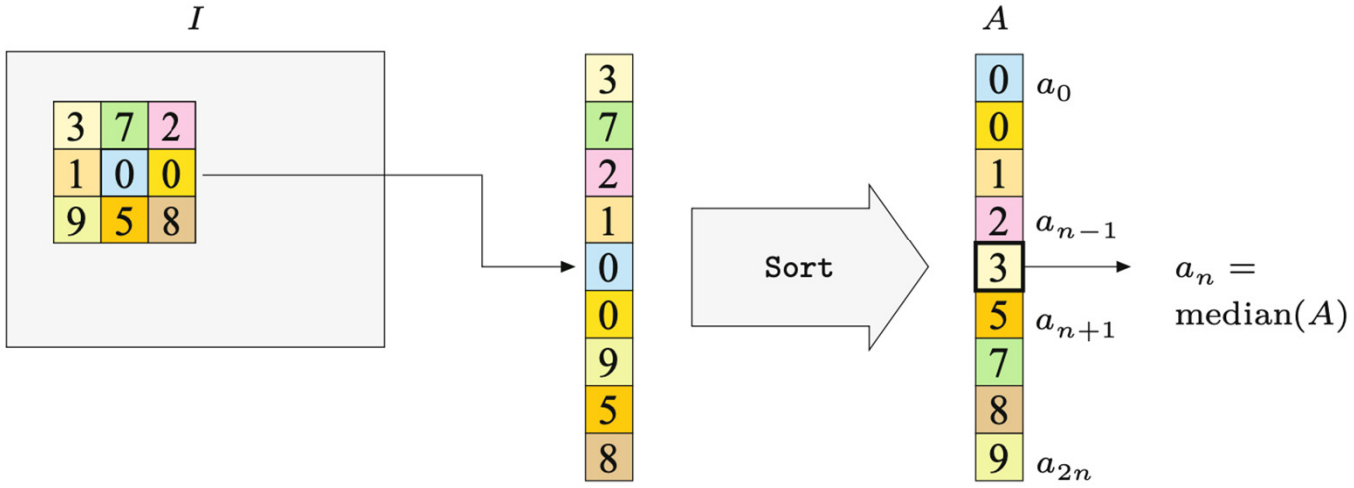
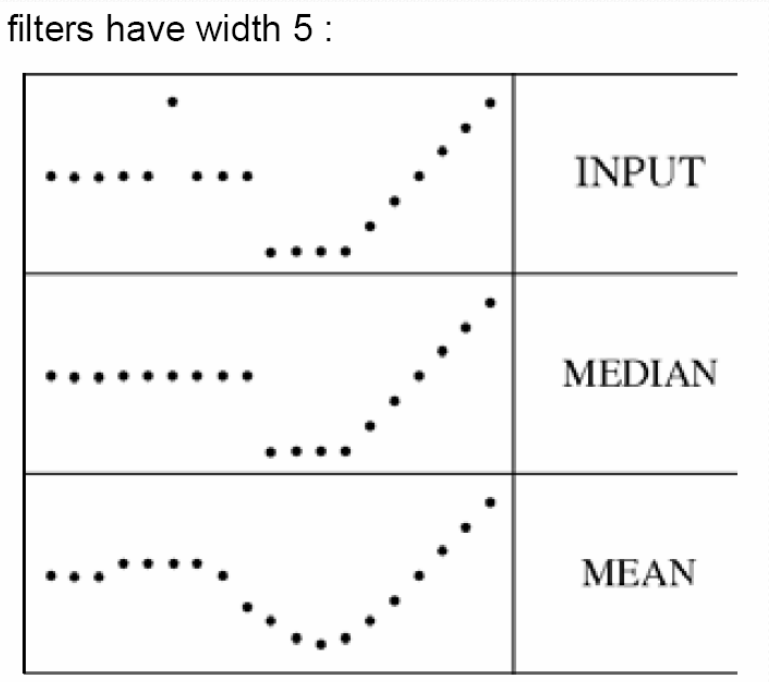


Fig. 5.17 Calculation of the median. The nine pixel values collected from the 3 × 3 image region are arranged as a vector that is subsequently sorted (A). The center value of A is taken as the median.

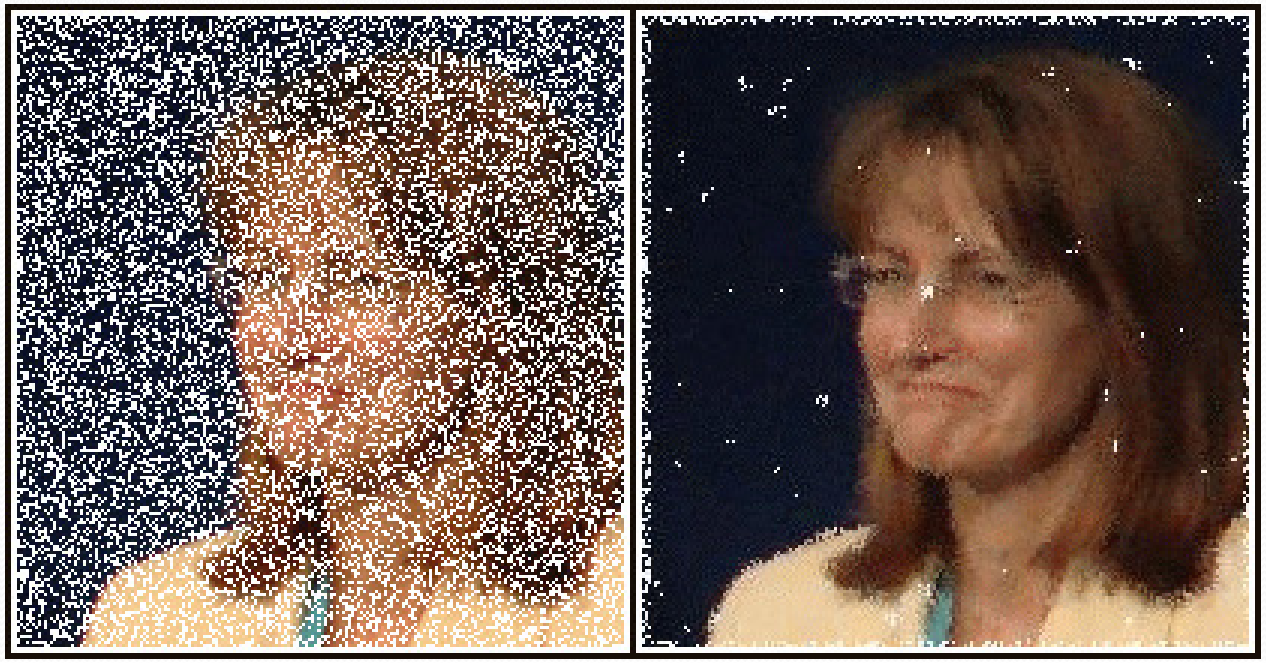
Filtro de mediana

- 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

Filtrado gaussiano vs mediana

Gaussian

Median

Fuente: Derek Hoiem (UIUC)

Otros filtros

- Mediana ponderada (los pixeles más alejados del centro cuentan menos).
- Media recortada (promedio, ignorando algunos de los pixeles más brillantes y más oscuros).
- Bilateral (peso por distancia espacial y diferencia de intensidad).
- Python: cv2.bilateralFilter(size, sigma_color, signal_spatial)

Filtro bilateral

Original

Filtro bilateral

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).
  - Puede ayudar a comprender el efecto del filtro.
  - Algoritmo:
    1. Convertir imagen y filtro a FFT.
    2. Multiplicación puntual de los FFTs.
    3. Convertir el resultado al dominio espacial con FFT 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.
  - Reducción de resolución (downsampling).
    - Se aplica un filtro pasa bajas y luego se hace la reducción.
  - Compresión.
    - JPEG (con pérdida, tamaño menor), PNG (sin pérdida, tamaño mayor).
  - Reducir el ruido (importante por motivos estéticos y para procesamientos posteriores como la detección de bordes).
    - Filtro gaussiano, filtro mediano, filtro bilateral.