

Práctica 1

Fusión de imágenes con Pirámides



OBJETIVO



- Fusión de imágenes mediante pirámides

Imagen A



Imagen B

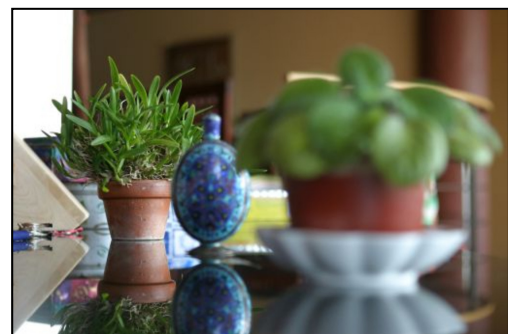


Imagen fusionada

- Descargue los ficheros de código Python disponibles en Moodle (fichero `p1_fusionPiramides_code.zip`)
- Complete las tareas utilizando los ficheros `*.py` contenidos en el ZIP
- La entrega se **realiza por parejas** constará de los siguientes ficheros:
 - Ficheros código Python de las tareas realizadas (hasta 4 ficheros)
 - Memoria en PDF (1 fichero) + ficheros Python adicionales






Entrega PRC1

Volver a: PRÁCTICAS

Archivos enviados

Tamaño máximo para nuevos archivos: 10MB, número máximo de archivos adjuntos: 7

No entregar los
ficheros Python
en ZIP!!!

Nombre	Última modificación	Tamaño	Tipo
 p1_memoria.pdf	20/09/2019 10:15	969.2KB	documento PDF
 p1_tarea1.py	20/09/2019 10:15	8.7KB	Archivo
 p1_tarea2.py	20/09/2019 10:15	9.8KB	Archivo
 p1_tarea3.py	20/09/2019 10:15	11.9KB	Archivo
 p1_tarea4.py	20/09/2019 10:15	8.1KB	Archivo

+ficheros preguntas memoria (hasta dos)

TSV – P

3

- **Tarea 0** – Descargar material e incluir nombre en todos los ficheros Python `*.py` y memoria
- Tareas programación:
 - **Tarea 1** - Implementar funciones básicas `reduce` y `expand`
 - **Tarea 2** - Implementar construcción pirámides Gaussiana y Laplaciana
 - **Tarea 3** – Implementar fusión y reconstrucción pirámides Laplacianas
 - **Tarea 4** – Implementar la fusión de dos imágenes en escala de grises
- Tareas razonamiento:
 - **Tarea 5** - Responder a las preguntas en memoria

No es necesario entregar todas las tareas

• Tarea 1 - Implementar funciones básicas `reduce` y `expand`

– Disponibles en el fichero `p1_tarea1.py`

– **Reduce(imagen):**

- Debe realizar las siguientes operaciones:
- 1. Crear un kernel de suavizado con la función `generar_kernel_suavizado(a)` disponible en el fichero `p1_utils.py` con `a = 0.4`
- 2. Convolucionar la imagen con este kernel utilizando la función `scipy.signal.convolve2d(imagen, kernel, 'same')`
- 3. Muestrear por 2 el resultado de la convolución (i.e. coger una de cada dos muestras en ambas direcciones) empezando por la primera posición del array, es decir en las posiciones impares (en Python la posición primera, tercera, quinta,... corresponden con índices pares 0, 2, 4,...)

– **Expand(imagen):**

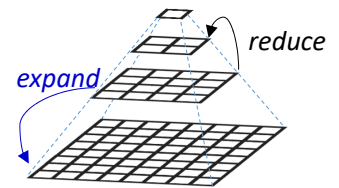
- Debe realizar las siguientes operaciones:
- 1. Crear una "imagen expandida" de dimensión doble comparada con la imagen de entrada
- 2. Copiar el contenido de imagen de entrada en la "imagen expandida" en las posiciones impares (índice par) de la imagen expandida
- 3. Crear un kernel de suavizado con `a = 0.4`
- 4. Convolucionar la imagen expandida con este kernel utilizando la función `scipy.signal.convolve2d(imagen, kernel, 'same')`
- 5. El resultado de 4 se multiplica por 4 para mantener el rango de la imagen de salida

• Tarea 2 - Implementar funciones para construir pirámides

– Disponibles en el fichero `p1_tarea2.py`

– Función **gaus_piramide(imagen, niveles)**

- Recibe una imagen y genera una pirámide
- El primer nivel de la pirámide es la imagen original.
Cada nuevo nivel se obtiene aplicando `reduce` sobre el nivel anterior:
- 1. Crear un kernel de suavizado con `a = 0.4`
- 2. Convolucionar la imagen con este kernel utilizando la función 'convolucion2d' de la tarea 1
- 3. Muestrear por 2 el resultado de la convolución (i.e. coger una de cada dos muestras en ambas direcciones) empezando por la primera posición del array)



– Función **lapl_piramide(gaus_pyr)**

- Esta función recibe una pirámide Gaussiana y calcula la pirámide Laplaciana según definido en teoría utilizando la operación `expand`
- Cada nivel 'k' de la pirámide Laplaciana se obtiene como la resta entre el nivel 'k' de la pirámide Gaussiana y la expansión del nivel 'k+1' de la pirámide Gaussiana
- El número de niveles de la pirámide Laplaciana viene determinado por el número de niveles de la pirámide Gaussiana
- El último elemento (imagen) de la pirámide Laplaciana es idéntico al último elemento (imagen) de la pirámide Gaussiana

• Tarea 3 – Implementar funciones de fusión y reconstrucción

– **fusionar_lapl_pyr**(lapl_pyr_img1, lapl_pyr_img2, gaus_pyr_mask):

- Funciones disponibles en el fichero `p1_tarea3.py`
- Las pirámides han de tener el mismo numero de niveles. La función debe indicar un error en caso de que no sean similares
- Se debe obtener una nueva pirámide Laplaciana con el mismo numero de 'niveles' que las pirámides Laplacianas
- Cada nivel de la nueva pirámide se corresponde con una suma ponderada con la mascara de los niveles de las pirámides de entrada

$$L_F^i = L_A^i * G_M^i + L_B^i (1 - G_M^i)$$

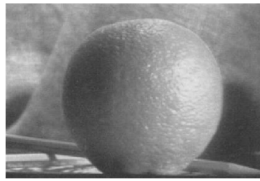


Imagen A

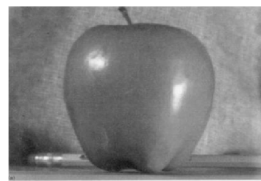
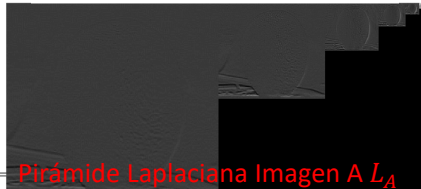


Imagen B



Máscara



Pirámide Laplaciana Imagen A L_A



Pirámide Laplaciana Imagen B L_B



Pirámide Gaussiana máscara G_M

TSV – Práctica 1: Fusión de imágenes mediante pirámides

7

• Tarea 3 – Implementar funciones de fusión y reconstrucción

– **reconstruir_lapl_pyr**(lapl_pyr)

- Esta función reconstruye la imagen dada una pirámide Laplaciana.
- Debe realizar las siguientes operaciones:
 1. Expandir el nivel 'k' de la pirámide
 2. Sumar la imagen expandida con el nivel 'k-1'
 3. Repetir la operación de manera iterativa
- Utilizar las funciones `reduce` y `expand` implementadas en la tarea 1
- Para más información, consultar comentarios en fichero `p1_tarea3.py`

• Tarea 4 – Implementar la fusión de dos imágenes en gris

– Funciones disponibles en el fichero `p1_tarea4.py`

– `run_fusion(imgA, imgB, mask, niveles)`

- Esta función implementa la fusión de dos imágenes calculando las pirámides Laplacianas de las imágenes de entrada y la pirámide Gaussiana de una máscara.
- Debe realizar las siguientes operaciones:
 1. Verificar que las imágenes son matrices bidimensionales (i.e. escala de grises) pues esta función no procesa imágenes RGB. La función debe indicar un error en este caso
 2. Convertir las imágenes y máscara a tipo `float`
 3. Normalizar todas las imágenes/máscara tipo `float` en el rango `[0,1]`
 4. Calcular las pirámides Gaussianas de las imágenes calculadas en el apartado anterior utilizando la función `"gaus_piramide"`
 5. Calcular las pirámides Laplacianas de las imágenes utilizando la función `"lapl_piramide"`
 6. Fusionar las pirámides Laplacianas de las imágenes y la Gaussiana de la máscara con la función `"fusionar_lapl_pyr"`
 7. Reconstruir la pirámide resultante para obtener una imagen con la función `"reconstruir_lapl_pyr"`
 8. Tras la reconstrucción, algunos valores pueden estar fuera de rango (`<0` o `>1`). En caso positivo, recortar a '0' o '1' según corresponda.
 9. El tipo de imagen de salida debe ser `float`

9

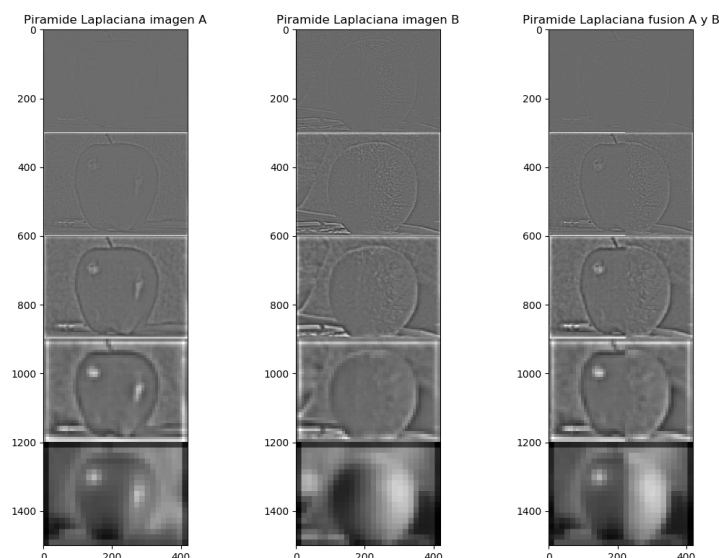
• Tarea 4 – Implementar la fusión de dos imágenes en gris

– Se proporcionan dos métodos en `p1_utils.py` para visualizar

– `visualizar_lapl_piramide(lapl_pyr)`

– `visualizar_gaus_piramide(gauss_pyr)`

– Devuelven una imagen que puede mostrarse con `matplotlib`



• Tarea 5 - Memoria

–Responda a las preguntas del fichero “preguntas_P1.docx”

- No necesita realizar funciones nuevas, el trabajo se basa en realizar experimentos con la funcionalidad que ha generado
- Puede necesitar repasar algunos conceptos teóricos
- Utilice figuras y tablas para resumir experimentos

–Ficheros a generar

- Fichero “preguntas_P1.pdf” en formato PDF con las respuestas
- Genere tantos ficheros Python como sea necesario siguiendo el formato `p1_pregunta_XX.py` donde XX es el número de pregunta

–Criterios generales de evaluación: texto conciso y claro; numeración de diagramas, figuras y tablas; referencias a figuras y tablas en texto; referencias a fuentes externas (si procede); errores ortográficos.

• ¿Cómo comprobar si la implementación es correcta?

–Se proporciona funciones que se pueden ejecutar cuando se desee

- `test_p1_tarea1()` en el fichero `p1_tarea1.py`
- `test_p1_tarea2()` en el fichero `p1_tarea2.py`
- `test_p1_tarea3()` en el fichero `p1_tarea3.py`
- `test_p1_tarea4()` en el fichero `p1_tarea4.py`

```
if __name__ == "__main__":  
    print("Practica 1 - Tarea 1 - Test autoevaluación\n")  
    print("Tests completados = " + str(test_p1_tarea1()))
```

- ¿Cómo comprobar si la implementación es correcta?
 - Para ejecutar la función de test de cada tarea, simplemente ejecutar el fichero Python correspondiente

test_p1_tarea1()

```
Realizando tests para las funciones de la tarea 1
Las funciones seran correctas si los resultados obtenidos
tienen una tolerancia de 2 decimales con respecto al resultado correcto.

* Evaluando la función "reduce"
Test imagen dimensiones(6, 8)...OK
Test imagen dimensiones(3, 4)...OK
Test imagen dimensiones(2, 2)...OK
Test imagen dimensiones(5, 7)...OK
Test imagen dimensiones(3, 4)...OK
Test imagen dimensiones(2, 2)...OK
Función "reduce" correcta!

* Evaluando la función "expand"
Test imagen dimensiones(1, 1)...OK
Test imagen dimensiones(2, 2)...OK
Test imagen dimensiones(3, 5)...OK
Función "expand" correcta!
Tests completados = True
PS E:\OneDrive - UAM\docencia\grado.gitst.tsv\practicas\code\practicasTSV>
```

- **Funciones permitidas:**
 - Operaciones básicas con Numpy (e.g., suma `numpy.add()` or simplemente `+`, multiplicación elemento-a-elemento `numpy.multiply()` o simplemente `*`, sumatorio `numpy.sum()`, inversión `numpy.flip`, recorte de rango `numpy.clip()`, padding `numpy.pad()`, rotación `numpy.rot90()`, generación de imágenes RGB a partir de varios arrays `numpy.stack()`, etc.
- **Funciones prohibidas:**
 - Funciones del paquete Python-opencv (`cv2.pyrUp`, `cv2.pyrDown` y `cv2.merge`)
 - No se pueden utilizar las funciones `img_as_float64`, `img_as_float32`, `img_as_float` para normalización de imágenes. Se debe realizar la operación manualmente.
 - Cualquier método no incluido en los paquetes definidos en el entorno virtual de la asignatura *PracticasTSV_env*

Preguntar al profesor ante cualquier duda

- 3 sesiones de prácticas (14 horas)
 - 6h presenciales (1x obligatoria + 2x opcionales)
 - 8h no presenciales

TAREA	Horas presenciales	Horas no presenciales	Horas TOTAL	Sesión de prácticas
Intro Prácticas + Explicación PRC1	1.5 h	0 h	1.5 h	1
Tarea 1	0.5 h	2 h	2.5 h	1
Tarea 2	2 h	1 h	3 h	2
Tarea 3	2 h	0 h	2 h	3
Tarea 4	0 h	1 h	1 h	-
Memoria	0 h	4 h	4 h	-
TOTAL	6h	8h	14h	

- Evaluación de la práctica sobre 10 puntos

TAREA	Max nota	Criterio evaluado (ver rúbrica en Moodle)
Tarea 1	2.5	Código: Ejecución (60%) Código: Diseño(40%)
Tarea 2	2	
Tarea 3	2	
Tarea 4	1	
Memoria	2.5	Memoria: Claridad, exactitud de la respuesta y experimentos realizados (100%)
TOTAL	10	

- Penalizaciones:
 - Por entrega de ficheros no acorde a las especificaciones: -0.5 puntos
 - Por uso de funciones prohibidas: -50% tarea
 - Por entrega tardía (tras considerar los 4 días disponibles para cada pareja):
 - -25% (un día), -50% (dos días), -75% (tres días), -100%(>= días)

- Wikipedia [https://en.wikipedia.org/wiki/Pyramid_\(image_processing\)](https://en.wikipedia.org/wiki/Pyramid_(image_processing))
- **Artículos originales de pirámides:**
 - P. J. Burt and E. H. Adelson, "The Laplacian Pyramid as a Compact Image Code," *IEEE Trans. on Communications*, Vol. COM-31, no. 4, April 1983, pp. 532-540.
 - E. H. Adelson, C. H. Anderson, J. R. Bergen, P. J. Burt and J. M. Ogden, "Pyramid methods in image processing", *RCA Engineer*, 29-6, Nov/Dec 1984
- **Artículo de aplicación para fusión de pirámides:**
 - P. J. Burt and E. H. Adelson, "A Multiresolution Spline With Application to Image Mosaics," *ACM Transactions on Graphics*, Vol. 2. No. 4, October 1983, Pages 217-236.
- Artículos disponibles en Moodle para consultar