**Visión por Computador**

Proyecto Final:

Implementación del Algoritmo de Seam Carving

Curso 2019-2020   
Cuarto Curso del Grado en Ingeniería Informática

Paula Iglesias Ahualli 75574877M  
[piglesias@correo.ugr.es](mailto:piglesias@correo.ugr.es)

Laura Rabadán Ortega 79088745W  
[laurarabadan@correo.ugr.es](mailto:laurarabadan@correo.ugr.es)

Contenido

[1. Detección de puntos Harris 3](#_Toc27953630)

[2. Detección y extracción de descriptores AKAZE 10](#_Toc27953631)

[3. Generación de mosaicos (N=2) 14](#_Toc27953632)

[4. Generación de mosaicos (N=10) 17](#_Toc27953633)

[5. Referencias 18](#_Toc27953634)

# Introducción

El paper en el que basamos nuestro proyecto, *Seam Carving for Content-Aware Image Resizing*,publicado en 2007 en SIGGRAPH afronta el problema de modificar el tamaño de una imagen sin afectar al contenido. Cuando la imagen se quiere reducir, se intenta no modificar información que se considera relevante por deformación de la imagen. De la misma manera, al aumentar el tamaño, se quiere evitar *estirar* los objetos, deformarlos al buscar mayores tamaños. Con *Seam Carving* se intenta dar una solución a estos problemas.

Los autores, Avidan y Shamir, proponen el concepto de *seam* (“costura”), en el que se basa la solución. Consiste en un camino óptimo de píxeles conectados de manera vertical, de abajo hacia arriba de la imagen, u horizontal, de derecha a izquierda, con menor energía de la imagen, es decir, los píxeles de menor relevancia. Se eliminarán o añadirán los *seams* necesarios para conseguir el tamaño deseado.

# Funciones de Energía

Para conseguir el camino de píxeles de menor relevancia que se mencionó en el apartado anterior, se necesita calcular la energía de la imagen, para seleccionar la *seam* de menor energía.

La primera función para este propósito que se propone en el paper estudiado, *e1*, consiste en calcular la suma de la derivada de la imagen en *x* y en *y*.



A la hora de implementarla, se utiliza la función de *Sobel* de OpenCV, calculando, por un lado, la derivada de la imagen en x y por otro lado, la derivada en y.

**def** simpleEnergy (image):  
 image = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)  
 image = image.astype(np.float)  
  
 x = np.abs(cv2.Sobel(image, cv2.CV\_64F, 1, 0))  
 y = np.abs(cv2.Sobel(image, cv2.CV\_64F, 0, 1))

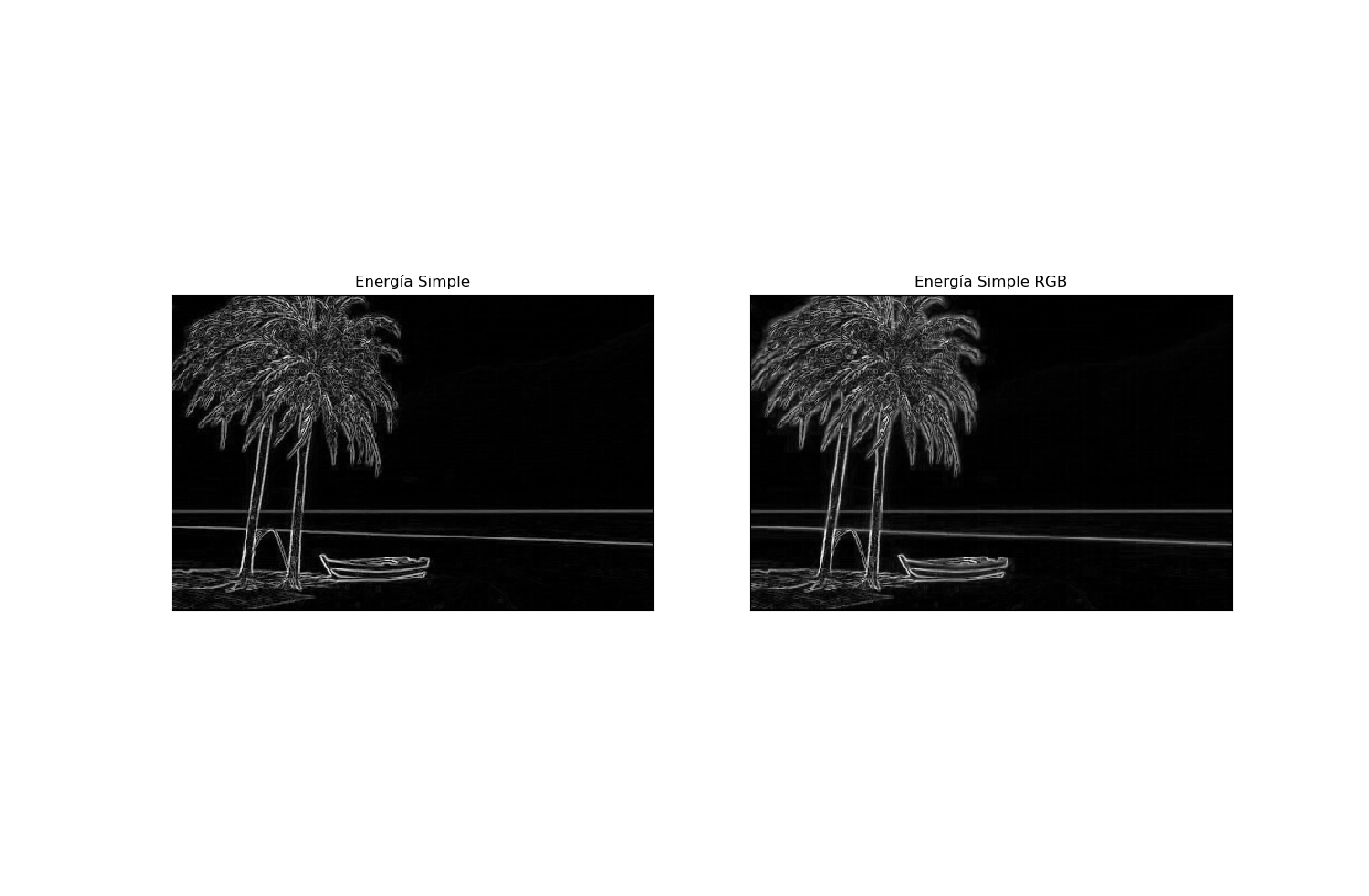
**return** x + y

Como esta función trabaja con la imagen en formato monobanda, implementamos una segunda función de energía, similar a la anterior, pero trabajando con la imagen en formato tribanda. Es similar a la anterior, pero calculando la suma de las derivadas para cada banda. El resultado es la suma de todas las derivadas conseguidas.

**def** simpleEnergyRGB(image):  
 b, g, r = cv2.split(image)  
  
 b\_energy = np.absolute(cv2.Sobel(b, cv2.CV\_64F, 1, 0, ksize=3)) +   
 np.absolute(cv2.Sobel(b, cv2.CV\_64F, 0, 1, ksize=3))  
 g\_energy = np.absolute(cv2.Sobel(g, cv2.CV\_64F, 1, 0, ksize=3)) +   
 np.absolute(cv2.Sobel(g, cv2.CV\_64F, 0, 1, ksize=3))  
 r\_energy = np.absolute(cv2.Sobel(r, cv2.CV\_64F, 1, 0, ksize=3)) +   
 np.absolute(cv2.Sobel(r, cv2.CV\_64F, 0, 1, ksize=3))

**return** b\_energy + g\_energy + r\_energy

A continuación se muestra un ejemplo de la salida de cada función de energía:



# Modificar una Seam

La técnica comentada puede utilizarse para dos tareas básicas: reducir el tamaño de una imagen eliminando *seams*, y aumentarlo, duplicando los *seams*.

## Eliminar una seam

Para disminuir el tamaño de una imagen, se deben eliminar *seams* de la imagen, seleccionándolos previamente. Para ello, se utiliza la energía de la imagen y se calcula la matriz de energía acumulativa.

La matriz de energía acumulativa, *M*, es esencial para este proceso. La idea es crear todos los caminos de menor energía acumulada y, a partir de ellos, elegir aquel que menos energía tenga. Para rellenar esta matriz, se empieza desde la segunda fila de la matriz de energía y se rellena la primera fila de *M* con 0. Desde este momento, cada píxel tendrá la energía correspondiente a esta función:



Dónde:

* es el píxel que estamos calculando.
* es la energía del píxel.
* es el píxel superior izquierdo al que estamos calculando.
* es el píxel superior al que estamos calculando.
* es el píxel superior derecho al que estamos calculando

La idea es, de los posibles píxeles superiores a los que se puede ir, elegir el que mejor energía tenga. Como se mira en la matriz de energía acumulativa, esos píxeles posibles habrán elegido el padre con menor energía, por lo que vamos consiguiendo un camino de energía mínimo.

Una vez tenemos esta matriz rellena, buscamos en su última fila el menor valor. Desde este valor se buscará, de manera inversa a como hemos rellenado , la *seam* que supone un menor coste de energía. Se guardan los índices que forman parte de la *seam* óptima para eliminarlos.

Este proceso es sencillo: simplemente, a partir del índice guardado, desplazamos toda la fila a la izquierda. Si el píxel es el último de su fila, no se hace nada. Hay que tener en cuenta que solo se va a eliminar una *seam* de cada fila, por lo que sobrará la última columna de la imagen, que se elimina.

Si se quiere eliminar una *seam* horizontal, se utiliza el mismo procedimiento. Para evitar rehacer la función, utilizamos los métodos anteriores pero rotando la imagen 90 grados, de manera que las filas pasan a ser las columnas y viceversa.

Cómo al eliminar una única *seam*, ya sea vertical u horizontal, la diferencia con respecto a la original no se aprecia, mostramos el resultado pintando las *seams* seleccionadas.

## Añadir una seam

Para añadir una *seam* el procedimiento es similar al anterior. Debemos buscar, de la misma manera que antes, la *seam* que consiga un camino de menor energía, ayudándonos de la matriz .

Esta vez, tenemos que duplicar la seam seleccionada con el valor promedio de sus vecinos de la izquierda y derecha. Cómo se quiere duplicar el píxel, para que quede acorde a la imagen, hay que modificar también el píxel que pertenece a la *seam*, para que no se note un salto brusco en la imagen.

Por tanto, suponemos que tenemos un píxel que pertenece a la *seam*. En ese punto tenemos que añadir dos nuevos píxeles: el primero haciendo el promedio entre el píxel original y su vecino de la izquierda, y el segundo con el promedio entre el píxel original y su vecino de la derecha. De esta manera, los dos píxeles guardan relación con el original, ya que uno de los valores que le influye es el del propio píxel original, pero cada uno está más relacionado con el píxel vecino. En el caso de los píxeles que son bordes, el del borde será el original y al otro se le hará el promedio.

Para hacer el promedio de dos píxeles no basta con sumar los dos valores y dividirlos entre dos, como se hace normalmente. En este caso, hay que multiplicar cada valor por si mismo antes de sumarlos y dividirlos entre dos. A este resultado le hacemos la raíz cuadrada y conseguimos el valor promedio de los dos píxeles.

Al igual que en el caso anterior, para añadir una *seam* horizontal el procedimiento es el mismo.

Al añadir una *seam* pasa lo mismo que al eliminarla, apenas se nota diferencia con respecto a la original. Por esta razón, resaltamos la *seam* seleccionada en rojo.

# Modificar Varias Seams

Cuando queremos modificar el tamaño de una imagen, lo normal es que tengamos que añadir o eliminar más de una *seam*, ya sea horizontal o vertical. Para el caso de eliminar *seams*, basta con repetir de manera iterativa el proceso anterior tantas veces como filas o columnas quieran añadirse o eliminarse.

## Eliminar varias seams

El problema surge cuando se quieren modificar las dos dimensiones de la imagen . En este caso, los autores plantean la pregunta de en qué orden eliminar las *seams*. Como solución, proponen utilizar una matriz, , de dimensiones , siendo y , en la que se guarde, en cada casilla, el mínimo coste para conseguir ese tamaño. Es decir, si estamos en , el mínimo coste posible para reducir 1 fila y cuatro columnas de la imagen original será el valor que contenga.

Para rellenar la tabla, empezamos con a , ya que el coste mínimo para conseguir el tamaño actual es nulo. El resto de las casillas se rellena calculando:



Es decir, el mínimo coste entre la energía de la casilla a la izquierda más el coste de quitarle una *seam* vertical a la imagen actual y la energía de la casilla de encima más el coste de quitarle una *seam* horizontal a la imagen actual. En una matriz auxiliar con las mismas dimensiones que vamos guardando la opción que se eligió en cada momento para cada casilla de .

Tras iteraciones, tendremos un mapa de menores energías para cada tamaño posible, eliminando un máximo de *r* filas y *c* columnas. La casilla contendrá el mínimo coste de energía posible para conseguir el tamaño deseado.

A partir de los valores de la matriz , seleccionamos el orden en el que se va a aplicar la reducción de *seams*. Empezando en vamos eligiendo el vecino, izquierdo o superior (exclusivamente), con menor energía. Si hemos elegido el vecino de la izquierda, guardamos como opción *seam* horizontal; si, por el contrario, elegimos el vecino superior, guardamos *seam* vertical. Repetimos este proceso hasta llegar a .

A la hora de implementarlo, nos dimos cuenta de que cada píxel acumula el menor coste de los dos píxeles entre los que se tendrá que elegir a la hora de buscar el orden. Por tanto, si en vez de la matriz utilizamos la matriz en la que guardamos la información del tipo de *seam* eliminada en cada momento. De esta manera, si es *seam* horizontal, le restamos 1 a , ya que hemos eliminado una fila de la imagen; si, por otro lado, es *seam* vertical, le restamos 1 a , ya que hemos eliminado una columna.

Una vez tenemos seleccionado el orden en el que vamos a ir eliminando *seams*, resultado de la función anterior, solo debe irse eliminando *seams* horizontales y verticales, según se indique, con el mismo proceso con el que se eliminaba una sola.

## Añadir varias seams

Si en vez de reducir la imagen queremos aumentarla, no basta con utilizar un bucle que vaya añadiendo *seams*, ya que todas tenderían a añadirse en el mismo lugar que es donde estaría la menor energía. Para evitar esto, que produciría una deformación en la imagen, se deben coger las *k* mejores *seams* para eliminar. De esta manera el aumento estará repartido por la imagen.

Para representar este comportamiento en nuestro programa, se realiza el procedimiento anterior ([eliminar varias *seams*](#_Eliminar_varias_seams)) y guardamos los caminos que se seleccionan para eliminar. Cuando se cambie de *seam* vertical a horizontal o viceversa, aumentamos las *seams* según los caminos seleccionados.

Con este procedimiento se consigue un resultado más correcto al que se conseguiría al añadir *seams* de manera iterativa.

## Añadir y eliminar varias seams

Si en vez de eliminar o aumentar tanto las filas como las columnas queremos aumentar una de las dos y disminuir la otra, dividimos el problema en dos: primero reducimos a imagen en la cantidad que se pide y después la aumentamos. Como una de las operaciones va a trabajar con las filas y la otra con las columnas, no interfieren la una con la otra.

Realizamos primero la operación de eliminar para evitar cálculo innecesario: si aumentamos y después reducimos, eliminamos píxeles que hemos tenido que calcular. Este caso se contempla en la función que se encarga de reducir el tamaño de una imagen. Cuando una de las dimensiones se quiere reducir y la otra ampliar, primero llama a reducir, manteniendo la dimensión que se va a ampliar intacta, y después a aumentar, con el mismo criterio que antes.

**def** contentAwareResizing (img, nn, nm):  
 n, m = img.shape[:2]  
 dif\_n = n – nn  
 dif\_m = m – nm  
  
 **if** dif\_n > 0:  
 **if** dif\_m < 0:  
 img = funcion(img, nn, m, accion[0], energy, draw)  
 img = funcion(img, nn, nm, accion[1], energy, draw)

**else**:  
 img = funcion(img, nn, nm, accion[0], energy, draw)  
  
 **elif** dif\_n < 0:  
 *// Espejo del caso anterior*

**else**:  
 *// Solo se eliminan horizontales o verticales*  
 **return** img

# Caso Práctico: Eliminar un Objeto

# Mejoras

## Forward Energy

## Scale