SEAM CARVING

Una implementación del método para reajustar imágenes en función de su contenido

Informe Final de Introducción al Procesamiento de Imágenes

Armanasco, Matías y Suriano, Ramiro

25 de septiembre 2020

Resumen

En este informe se detalla el trabajo realizado para implementar el método de SeamCarving, escalado inteligente de imágenes.

1. Introducción

El motivo de este trabajo fue encontrar una aplicación práctica de los métodos estudiados en el curso de Introducción al Procesamiento Digital de Señales.

Particularmente nos interesamos en el ajuste dinámico del tamaño de las imágenes y hallamos el paper *Seam Carving for Content-Aware Image Resizing*[1], el cual consiste en re-ajustar una imagen al tamaño deseado mediante un análisis de los posibles caminos de píxeles a eliminar asignándoles peso relativo según su energía.

2. Trabajo Previo

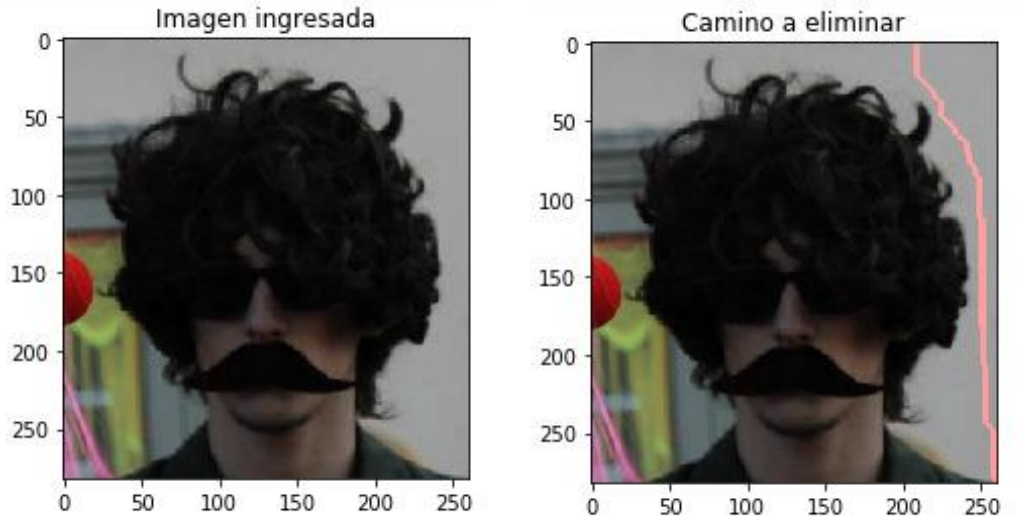
El trabajo realizado en Seam Carving for Content-Aware Image Resizing cubre varios aspectos más allá de los analizados en este trabajo. Los autores desarrollaron un método capaz de eliminar caminos o 'seams' de píxeles con poca energía, es decir, con información poco relevante, de esta manera se protegen áreas de la imagen haciendo que parezca natural el recorte de tamaño. Normalmente cuando se recorta una imagen se dejan partes relevantes fuera de la misma o si se elimina una zona central se nota la unión, este método permite evitar esto y mantiene una forma agradable a la vista luego de ser recortada.

Además, el trabajo realizado por Avidan y Shamir utiliza este mismo principio a la inversa, puede agrandar el tamaño de la imagen creando píxeles nuevos alrededor de los *seams* de baja energía promediando el color de sus vecinos de manera que no se note la introducción de información que no estaba originalmente en la imagen.

También sirvió como base, y es el contexto en el cual se realiza este trabajo final, el conocimiento aprendido durante el cursado de la materia Introducción al Procesamiento Digital de Imágenes. De donde pudimos recolectar métodos y técnicas de procesamiento que nos fueron de gran utilidad para desarrollar este algoritmo.

3. Desarrollo

El principio básico del seam carving consiste en eliminar selectivamente líneas horizontales y/o verticales de una imagen llamadas *'seams'*, las cuales consisten en líneas de píxeles consecutivos que atraviesan la imagen horizontal o verticalmente. Al eliminar primero los caminos o *seams* con menor energía, la imagen resultante conservará las características de la original, obteniéndose una imagen de menor tamaño sin distorsión.



(a) Imagen original (b) Seam de mínima energía

Figura 1: Muestra gráfica de un seam

El desafío entonces es primero encontrar una forma de obtener la energía de cada píxel de la imagen. Para esto se utiliza el operador Sobel, que calcula el gradiente total de la imagen. Este se obtiene convolucionando la luminancia de la imagen con el kernel sobelX para obtener el gradiente horizontal de la misma, y con el kernel sobelY para obtener el gradiente vertical. Luego se calcula el total como el módulo entre las dos imágenes filtradas:

$$magnitudSobel = \sqrt{sobelX^2 + sobelY^2} \tag{1}$$

donde,

$$sobelX = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{2}$$

$$sobelY = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{3}$$

A continuación se debe encontrar el camino de menor energía a lo largo de la imagen. Como los algoritmos para calcular seams funcionan de igual forma para calcular filas o columnas si la imagen de entrada es rotada o traspuesta, el análisis será indistinto.

La búsqueda de *seams* se hace a partir de la energía de la matriz de luminancia de la imagen. En una primera iteración del algoritmo, se consideró armar el *seam* de mínima energía empezando por la fila superior de la imagen, eligiendo el píxel con menor valor, y eligiendo para cada píxel el menor de sus tres vecinos inferiores, y repitiendo el proceso hasta llegar al final de la imagen. El problema con este enfoque es que solamente prioriza los píxeles con menor energía, sin tener en cuenta que en algunas imágenes el camino de menor gradiente se encuentre al atravesar una porción de la imagen que pueda contener gran energía. De igual forma, al elegir siempre los píxeles con menor valor, el *seam* puede quedar "atrapado"en una región con alta energía de la imagen.

Para solucionar este problema, en lugar de elegir entre múltiples píxeles para conectar a un *seam*, para cada pixel se elige entre múltiples *seams*, el de menor energía[(2)]. De esta forma, para cada píxel hay que mirar a los tres *seams* que terminen en los píxeles superiores adyacentes al actual.

Para la fila superior de la imagen, no hay *seams* entre los cuales elegir, por lo que la energía de cada píxel será la energía de cada *seam*. Para las filas sucesivas, cada píxel tendrá conectados tres píxeles superiores, y la energía del *seam* que desemboca en cada uno de ellos será la energía sobre ese píxel (ya que se van acumulando a medida que se recorren las filas). Al llegar a la fila inferior, los valores de energía serán los de los *seams* de menor energía dentro de la imagen, por lo que el que tenga menor valor será el píxel por donde se comience a remover.

Con el punto de inicio calculado, solo resta conocer las ubicaciones del *seam* de menor energía. Como no se saben las coordenadas de éste hasta haber procesado toda la imagen, se debe realizar una matriz auxiliar que incluya "direcciones"que den información sobre los *seams* calculados en el paso anterior. Por ejemplo, si para un píxel de coordenadas (i, j) el camino de menor energía viene desde $(i-1, j)$, es decir el elemento directamente superior, en la matriz *seamMap* la coordenada (i, j) tendrá un 0. A su vez, si el píxel se conectase con un seam superior izquierdo, la coordenada correspondiente en el mapa de seams será de -1, y 1 para una conexión con el píxel superior derecho. De esta forma, al finalizar el cálculo del seam de mínima energía, se obtienen dos variables: la posición en la última fila en donde comienza el seam a remover, y la matriz de Mapa de Energías de la que se recupera el camino del mismo.

```
def seamFinder(luminance, energyMap):

    seamMap = np.zeros(luminance.shape, dtype=int)
    #primer loop de seam finder, recorre fila por fila
    previousRow = energyMap[1]
    for row in range(energyMap.shape[0]-1):
        currentRow = np.zeros(energyMap.shape[1])
        #print(seamMap)

        #segundo loop, recorre elementos de cada fila para calcular energias
        for col in range(energyMap.shape[1]):

            #x_range define si se trata de un caso borde o no
            x_left = max(col - 1, 0)
            x_right = min(col + 1, energyMap.shape[1])

            #calculo de seam minimo encima de cada pixel
            prevPath = previousRow[x_left : x_right]
            minSeam = min(prevPath)
            #guardado de energias acumuladas en la nueva fila, y coordenadas en el
            ↪  mapa
            seamMap[row+1,col] = int(np.where(prevPath == min(prevPath))[0][0] + 1
            ↪  * (x_left - max(x_left, col)))
            currentRow[col] = energyMap[row+1,col]+minSeam
            #print(row+1,col)
            #print(seamMap)

        previousRow = currentRow

    startPoint = np.where(currentRow == min(currentRow))[0][0]

    return startPoint, seamMap
```

Finalmente para remover los *seams* hallados se procede a recorrer desde abajo hacia arriba la matriz de píxeles eliminando primero el que tiene menos energía en la última fila. Luego se sube un nivel para eliminar el píxel de la fila superior pero observando el Mapa de Energías, dependiendo si el Mapa tiene un -1, 0 o 1 el elemento a eliminar en la fila superior es el próximo izquierdo, superior, o derecho respectivamente.

A continuación se muestra la función implementada en Python que resuelve la eliminación de un seam. Esta función es llamada de forma recurrente hasta cumplir con el requerimiento de ancho o alto deseado.

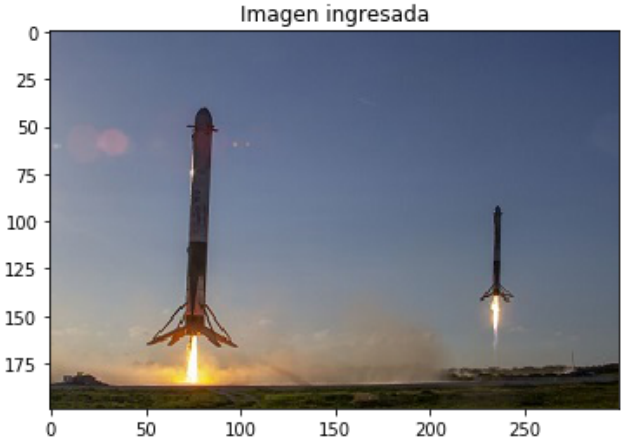
```
def seamremover_revisited(N,roads,imagenOriginal):
    #método para remover un camino de mínima energía almacenado en roads
    alto, ancho = imagenOriginal.shape[0:2]
    imagenAuxiliar = np.zeros([alto,ancho-1,3])
    ult_fila = alto-1
    pixel = N
    seam = np.zeros(imagenOriginal.shape)

    while (ult_fila>-1):
        fila = imagenOriginal[ult_fila,:]   #bottom-up salgo de la ultima fila y
        ↪  voy subiendo
        filaMapa = roads[ult_fila,:]

        pixel = N + filaMapa[N]  #pixel es el que voy a eliminar

        #elimino 'pixel' de las tres capas
        fila = np.concatenate((fila[0:N,:], fila[N+1:ancho,:]))

        seam[ult_fila,N-2:N+2,0] = 1.   #creo la imagen con el seam para mostrar
        ↪  graficamente
        imagenAuxiliar[ult_fila,:] = fila   #la fila sin 'pixel' la guardo en la
        ↪  nueva imagen
        N = int(pixel)              #actualizo N para cuando vuelvo a pasar el
        ↪  loop
        ult_fila = (ult_fila - 1)   #actualizo la ultima fila para cuando vuelvo a
        ↪  pasar el loop

    return imagenAuxiliar, seam #esta imagen entra de nuevo en seamcarver()
```

Aquí se puede ver el trabajo completo realizado en GitHub: enlace a GitHub.

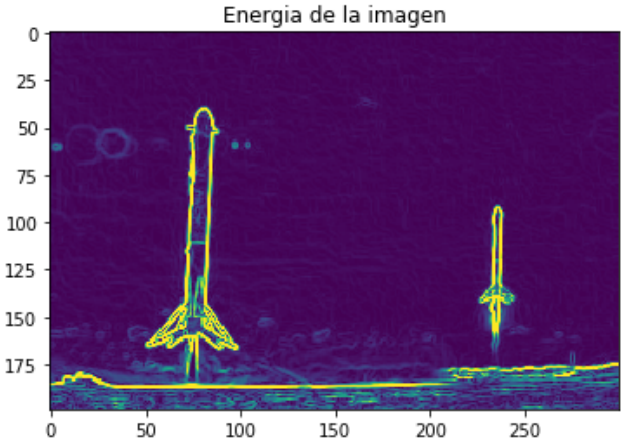
4. Resultados Obtenidos

En esta sección se muestran los resultados obtenidos para distintos tipos de imágenes donde la energía se encuentra distribuida de maneras diferentes y así poder comparar el comportamiento del algoritmo.

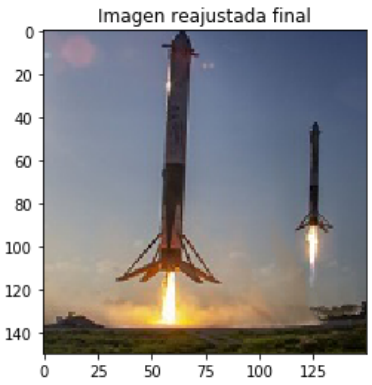
En el primer caso la imagen es de las dos primeras etapas de un cohete Falcon Heavy aterrizando, se puede observar que la mayor cantidad de información se encuentra focalizada en los dos Block5 por lo que resulta relativamente fácil hallar seams fuera de esos dos focos de energía. En la imagen final podemos observar como el humo sigue pareciendo natural sin notarse el recorte de píxeles.



(a) Imagen original [300x199]



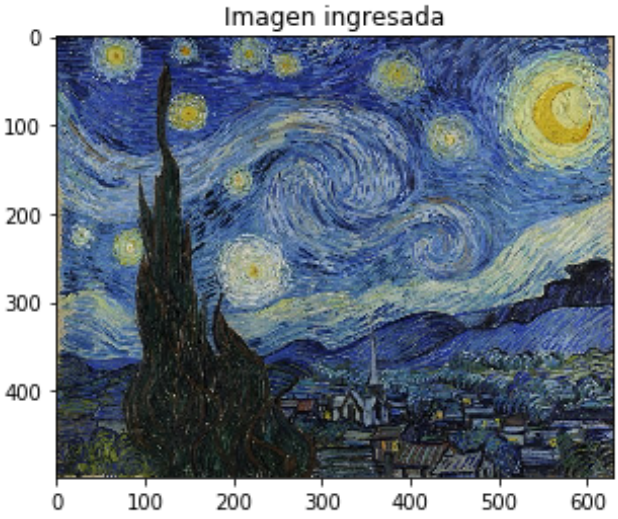
(b) Mapa de energía



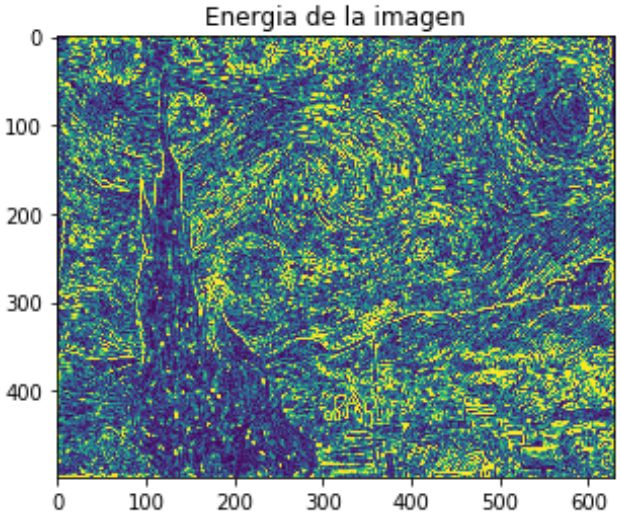
(c) Imagen recortada [150x190]

Figura 2: Dos etapas Block5 de un cohete Falcon Heavy aterrizando

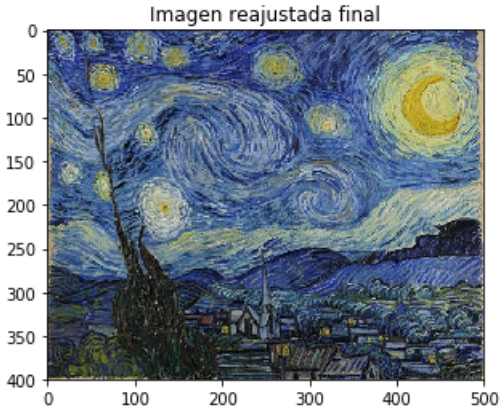
Esta segunda imagen se trata del clásico cuadro de Van Gogh, La Noche Estrellada. En este caso la energía está distribuida por toda la imagen por lo cual encontrar seams de mínima energía se vuelve más complejo, sin embargo puede verse que el algoritmo sortea esta dificultad y mantiene las figuras distinguibles luego de la reducción de píxeles. Puede verse que los "soles" en el cielo y las casas mantienen su estructura.



(a) Imagen original [630x498]



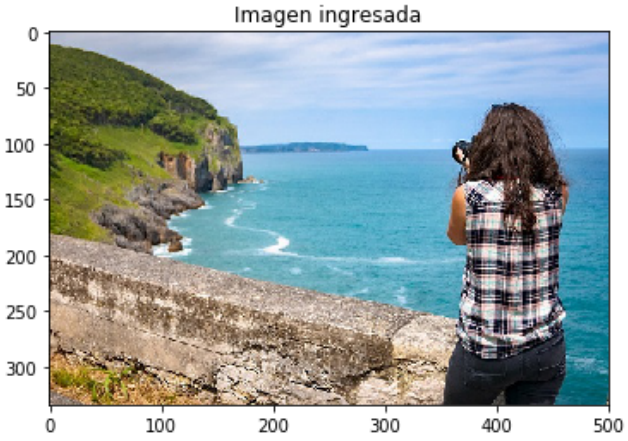
(b) Mapa de energía



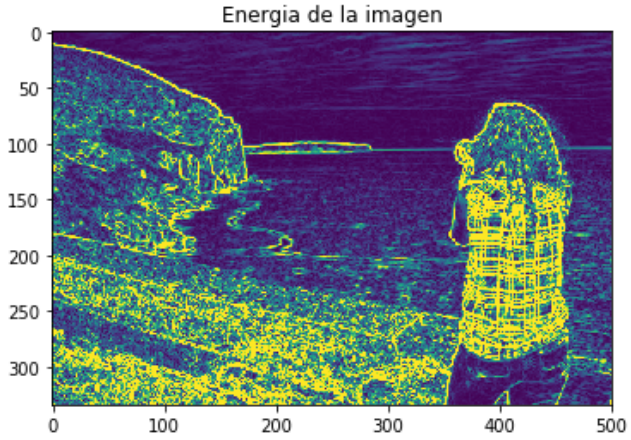
(c) Imagen recortada [500x400]

Figura 3: 'La Noche Estrellada' - Vincent Van Gogh

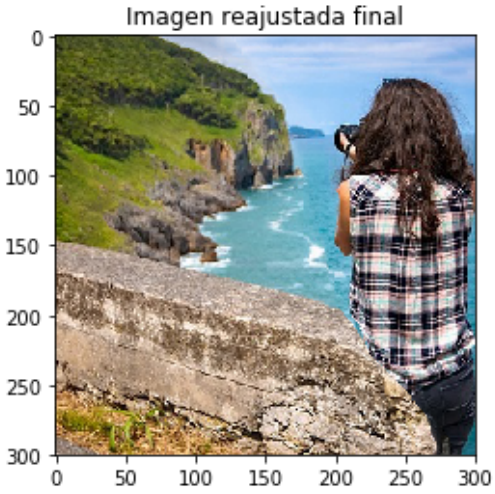
En la tercer imagen elegida se buscó un fondo con un paisaje natural para observar el comportamiento del SeamCarving, puede observarse que la mujer luego del resizing parece estar parada delante de la piedra ya que esa zona tiene mucha energía debido a la rugosidad del muro de piedra y los caminos de menor energía derivaron por el pantalón que al ser de un color uniforme no tiene prácticamente gradientes de cambios. Sin embargo el efecto final parece que la mujer está parada delante de la piedra sin resaltar algún corte poco orgánico a la vista.



(a) Imagen original [500x334]



(b) Mapa de energía



(c) Imagen recortada [300x300]

Figura 4: Una mujer sacando una foto a un paisaje

5. Conclusiones

El resultado fue satisfactorio, se logró cumplir con el objetivo y pudimos probar el comportamiento frente a distintos tipos de dificultades arrojando imágenes donde los objetos relevantes siguen siendo distinguibles y no se producen deformaciones. Por otro lado como limitación del método desarrollado se encuentra el tiempo de ejecución, tardando un promedio unos tres minutos por imagen, dependiendo del tamaño de la misma y de la cantidad de píxeles a eliminar.

Además, el proyecto tiene posibilidades de continuar a futuro agregándose funcionalidades como delimitar una zona para que no pase ningún seam por allí, o eliminar un objeto reemplazándolo por un patrón similar al fondo de la imagen. Todas funcionalidades que se basan en los algoritmos desarrollados aquí.

Referencias

[1] Avidan S, Shamir A. 'Seam Carving for Content-Aware Image Resizing'; Junio 2007. https://perso.crans.org/frenoy/matlab2012/seamcarving.pdf.

[2] Das A. 'Real-world dynamic programming: seam carving'; Mayo 2019. https://avikdas.com/2019/05/14/real-world-dynamic-programming-seam-carving.html.