Visión por Computador

Proyecto final

Curso 2017/2018

7. Implementar la creación de panoramas lineales con proyección rectangular y mezcla de colores usando el algoritmo de Burt-Adelson.

Francisco Javier Caracuel Beltrán

caracuel@correo.ugr.es

4º - Grado en Ingeniería Informática – CCIA – ETSIIT – UGR

Índice

1. Definición del problema a resolver

En la creación de panoramas existe un problema a resolver ajeno al algoritmo que los compone. El problema reside en que el tiempo que transcurre entre imagen e imagen puede provocar cambios en el entorno o, el propio enfoque de la cámara puede variar la iluminación que se obtiene en la imagen. Si se compone el panorama utilizando las imágenes sin tratar, se pueden apreciar cortes no realistas en el momento de unir las distintas imágenes de la composición. Sobre este problema es sobre el que trata el proyecto.

2. Enfoque e implementación elegida

Para construir el panorama se ha seguido un proceso que consta de los siguientes pasos:

El primer paso que se debe realizar para generar el mosaico es calcular el tamaño que tendrá. En este caso se ha establecido el ancho como la suma de todos los anchos de las distintas imágenes que componen el mosaico. La altura será el doble de la altura de la imagen central.

Existe una función crop_image() que elimina todos los bordes negros que se tienen en el mosaico tras añadir todas las imágenes. Como inicialmente no se conoce cuál es el tamaño mínimo que debe tener el contenedor, esta función permite trabajar con un tamaño mucho mayor al que finalmente se devolverá.

Cuando se crean mosaicos o panoramas, la imagen con mayor importancia y sobre la que recae el peso de una correcta visualización es la imagen central. Es sobre esta imagen central sobre la que se van uniendo el resto de imágenes hacia los extremos.

Para comenzar a colocar las imágenes, se calcula la que debe estar en el centro y se genera una homografía que sea capaz de trasladar esta imagen central, al centro del mosaico. Esta homografía se calcula manualmente en la función get_homography_to_center(). La homografía será una matriz 3x3 rellena de 1 en su diagonal, el desplazamiento en píxeles del eje X en la fila 1, columna 3 y el desplazamiento en píxeles del eje Y en la fila 2, columna 3. El resto de posiciones de la matriz tendrán valor 0.

El siguiente paso es calcular todas las homografías necesarias para colocar todas las imágenes en su lugar correspondiente. Este proceso se separa en dos partes:

- En la primera parte, se calculan las homografías desde la imagen central hasta el extremo derecho.
- En la segunda parte, se calculan las homografías desde la imagen central hasta el extremo izquierdo.

Se explica solo el proceso de colocar las imágenes que se encuentran a la izquierda de la principal, siendo el proceso de las imágenes derechas similar.

Al ser la imagen central la que tiene el peso del mosaico, para colocar el resto se debe partir de ella, calculando la homografía entre la imagen inmediatamente a su izquierda y ella (imagen central). Cuando ya se tiene la homografía, se sabe cuál es la transformación que se debe realizar sobre la imagen de la izquierda para que encaje con la central. El problema se encuentra ahora en que la imagen central no está en la posición original, por lo que visualmente no casan. Como se ha calculado la homografía que desplaza la imagen central al centro del mosaico, se multiplica esta homografía con la calculada entre ambas imágenes.

El proceso de cálculo de una homografía hace uso de los keypoints, descriptores y matches de las dos imágenes sobre la que se quiere calcular. Una vez obtenidos estos valores, se hace uso de la función de OpenCV *findHomography( )*, que necesita al menos cuatro puntos de emparejamiento de las imágenes para hacerla correctamente. Esta función devuelve una matriz de tamaño 3X3 que es la que contiene la homografía en cuestión.

Llegados a este punto ya se tienen todas las homografías que permiten desplazar cada imagen a su lugar correspondiente del mosaico, pero, se debe tener en cuenta que se quiere realizar un suavizado entre los bordes de las imágenes para que no se aprecien los bordes.

- Algoritmo de Burt y Adelson:

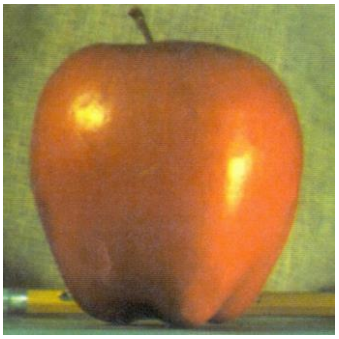
Para realizar el suavizado se utiliza el algoritmo de Burt y Adelson. Este algoritmo parte de dos imágenes y el resultado final es la mezcla de ambas, quedando en la parte izquierda la primera de ellas y en la parte derecha la segunda, con una suave transición entre ellas.

Burt y Adelson hicieron uso de la pirámide Gaussiana y Laplaciana para crear la transición entre las imágenes.

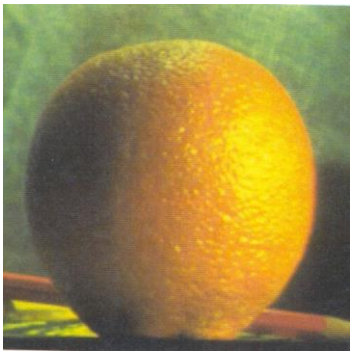
El primer paso consiste en crear la pirámide Gaussiana de *n* niveles de las dos imágenes. Una vez creada y haciendo uso de ésta, se crean las correspondientes pirámides Laplacianas. Cuando ya se tienen las dos pirámides Laplacianas creadas, se crea otra añadiendo en la parte izquierda de cada nivel de la pirámide, su correspondiente mitad de la primera y en la parte derecha, la mitad correspondiente de la segunda. Una vez creada esta pirámide mixta, se utiliza para recomponer las imágenes, obteniendo como resultado la imagen con ambas mitades y la transición entre ellas.

Se pone un ejemplo del resultado que ofrece la implementación de este algoritmo en el proyecto. Para realizarlo, se hace uso de la función creada *blend_images(img1, img2)* y de las imágenes que se encuentran en la web de la Universidad de Wisconsin-Madison:

Manzana:



Naranja:



Resultado del algoritmo de Burt-Adelson:

Una vez explicado el algoritmo de Burt-Adelson, se puede terminar de componer el mosaico. También se disponen de las homografías necesarias para ubicar en su lugar correspondiente cada imagen.

El proceso comienza añadiendo las imágenes de izquierda a derecha. Para añadir la primera imagen, se hace uso de la función de OpenCV *warpPerspective()*, en la que se indica la primea imagen y la primera homografía.

Antes de añadir la segunda imagen, se debe calcular el punto donde se ubicará ésta, de modo que se hace uso de la función creada *warp_point(x, y, h)*. Esta función recibe una posición (x, y) y una homografía (h), devolviendo el punto correspondiente en el mosaico. De este modo, se puede calcular la zona donde se colocará la segunda imagen, con el objetivo de recuperar esa área para utilizarla en el algoritmo de Burt-Adelson. Cuando ya se tiene el área guardada, se coloca la segunda imagen. Al colocar la imagen, se puede haber producido un cambio en la tonalidad de los colores en el borde de las imágenes, por lo que se obtiene esa segunda área, que será la que se envíe junto con la primera en el algoritmo de Burt-Adelson.

Como se ha comentado anteriormente, se envían las dos áreas (que deben tener el mismo tamaño y justo en su zona central es donde se debe producir la variación de tonalidad) a la función *blend_images(area1, area2)*. Esta función devuelve una imagen del mismo tamaño de cada área y será la que se sustituya en la posición correspondiente del mosaico.

Este proceso se repite con todas las imágenes y homografías disponibles, con el fin de componer el mosaico completo.

3. Valoraciones sobre la implementación y resultados

Para comprobar el resultado, se han utilizado las imágenes que ofrece la Universidad de Wisconsin-Madison en este enlace:

http://pages.cs.wisc.edu/~csverma/CS766_09/ImageMosaic/DataSet/Set7/RawImages/

Además, se han subido en la consigna de la Universidad de Granada, junto con los resultados obtenidos y los ejemplos de la manzana y naranja:

https://consigna.ugr.es/f/1t1SohSbVbeEJOe6/Im%C3%A1genes%20-%20Proyecto%20Final%20-%20Francisco%20Javier%20Caracuel%20Beltr%C3%A1n.zip

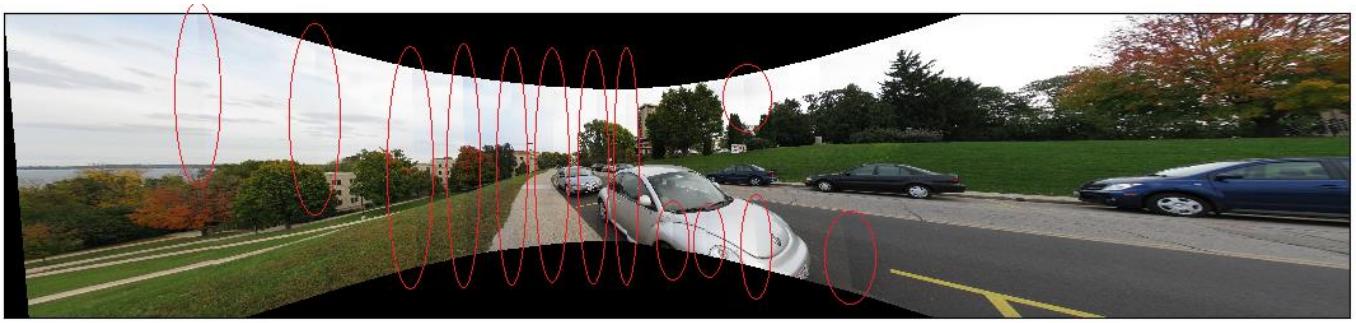
Se han utilizado las 13 primeras imágenes y se han redimensionado su tamaño a 768X1024, con el fin de realizar los cálculos de manera más ágil.

Además, para que funcione el código en Python, se debe crear un directorio llamado "*images*" en la misma ruta donde se encuentre el archivo fuente *proyecto_final.py*.

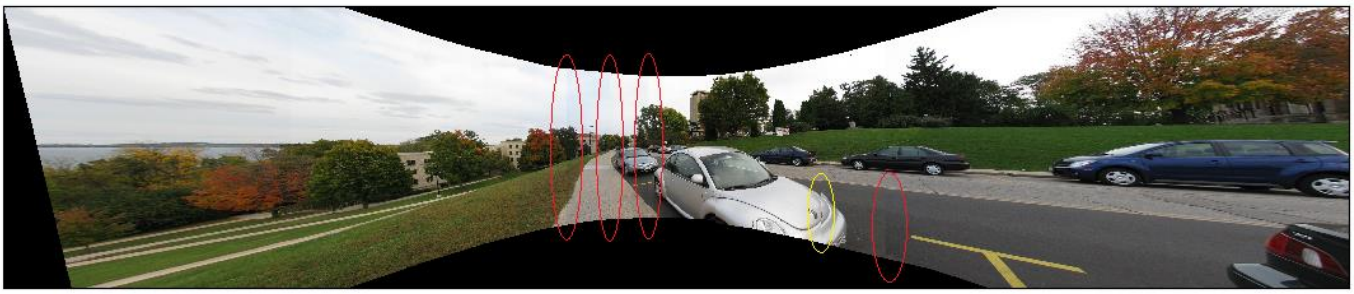
Resultado de la creación del panorama sin utilizar el algoritmo de Burt-Adelson:

Resultado de la creación del panorama utilizando el algoritmo de Burt-Adelson:

Zonas con cambios de intensidad de color sin utilizar el algoritmo de Burt-Adelson:



Zonas con cambios de intensidad de color utilizando el algoritmo de Burt-Adelson:



Tras aplicar el algoritmo se puede apreciar (sobre todo si se visualiza en las imágenes subidas en la consigna de la Universidad de Granada) como ha disminuido el nivel de cambios de intensidad en el color. Pese a eso, siguen manteniéndose algunas zonas donde si existen estos cambios, debido a que la diferencia entre ambas imágenes es más notoria.

Mi valoración sobre este algoritmo es que es correcto para una primera aproximación, ya que es fácil de comprender e implementar y no requiere demasiado tiempo de cómputo extra para realizarlo. Si se quisiera obtener unos resultados más profesionales, se debería optar por otras técnicas más avanzadas que consigan una transición menos brusca.

El principal problema observado de esta técnica, es que la imagen resultante que contiene la mezcla de las dos áreas del mosaico se ve influenciada en su totalidad por estas áreas, de modo que cuando estas áreas se vuelven a colocar en el mosaico, pueden provocar nuevos cambios en la intensidad del color, aunque eso sí, son menores que los originales.

4. Propuestas de mejora

Para mejorar este proyecto se podría implementar lo siguiente:

- Como paso natural para mejorar la composición del mosaico, realizar la creación del panorama con proyección en superficies cilíndricas o esféricas.
- Selección automática de la anchura de las áreas a enviar al algoritmo de Burt-Adelson, con el fin de que éstas sean las más óptimas para mezclar.
- Proceso iterativo de modo que, al colocar un área ya mezclada en el mosaico, se continúe mezclando esa parte final del área con la siguiente zona del mosaico para evitar los pequeños cambios en la intensidad del color que se puedan producir.
- Automatización de la posición de las imágenes, de modo que no importe si se hacen de izquierda a derecha o de derecha a izquierda o, incluso, si se realizan las fotografías en modo de tabla con *x* filas e *y* columnas.

5. Referencias

- http://www.cs.princeton.edu/courses/archive/fall05/cos429/papers/burt_adelson.pdf
- http://pages.cs.wisc.edu/~csverma/CS766_09/ImageMosaic/imagemosaic.html
- http://pages.cs.wisc.edu/~csverma/CS766_09/ImageMosaic/DataSet/Set7/RawImages/
- http://www.morethantechnical.com/2017/09/29/laplacian-pyramid-with-masks-in-opencv-python/
- https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_pyramids/py_pyramids.html
- http://alumni.media.mit.edu/~maov/classes/comp_photo_vision08f/lect/16_homography_blending_pyramids.pdf