Primer Examen Parcial

Mikel Ignacio Barajas Martínez, 336483, 202102300012

Ingeniería en Sistemas Inteligentes, 2021

Visión Computacional, 281501

1 de Marzo, 2024

1. Planteamiento del Problema

Aplicar a la imagen que está junto a este archivo en DidacTIC las siguientes operaciones (NOTA IMPORTANTE: aplicarlas en el orden que usted considere el adecuado para obtener el mejor resultado) :

a Aplicar mínimo 3 filtros y máximo 6 filtros, ya sea de convolución, morfológicos o de Fourier para eliminar el ruido de la imagen, de manera que el rostro de la persona aparezca lo más nítido posible.

b Recortar la imagen de manera que sólo aparezca la silueta de la persona en la misma, es decir, lo más ceñido posible a ella. Ver (b).

c Rotar la imagen de manera que la silueta de la persona aparezca completamente en vertical. Ver (c).

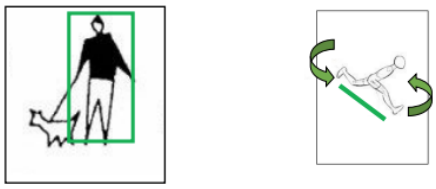

Figura 1: Imagen recortada (b) Imagen rotada (c)

2. Descripción de la Solución

A. Eliminación de Ruido

La primera actividad se trata de la restauración de una imagen ruidosa. En caso de que solamente se requiriera hacer la reducción de ruido, se podría usar un filtro de suavizado cualquiera. Dado que también se busca preservar el detalle, se podría aplicar un filtro bilateral.

Sin embargo, como se puede observar en la imagen, A pesar de utilizar este filtro la imagen resultante no mantiene los bordes del rostro. Una posible solución es que, antes de aplicar el filtro bilateral, se realiza un proceso de extracción de bordes de la imagen, los cuales se almacenan de manera independiente y se le agregan a la imagen suavizada para que se obtenga la reducción de ruido sin que desaparezca el detalle (los bordes).

1. El primer paso para la extracción de bordes consiste en "aumentar" su presencia en la imagen original, es decir, mejorar el contraste y perfilado de la imagen. Esto se hace con

la finalidad de obtener bordes mejor demarcados en el paso de extracción, además de facilitar el encontrar los parámetros para ese mismo proceso. Existen varias técnicas que se pueden utilizar para lograr el efecto esperado. Se hicieron pruebas ecualizando el histograma de la imagen (lo cual resultaba en más ruido al final), y pruebas usando un kernel de perfilado laplaciano (que creaba bordes muy poco definidos), pero al final se decidió utilizar un aumento de contraste simple utilizando la función **cv2.convertScaleAbs(alpha, beta)**. El parámetro alpha determina el contraste (default = 1), por lo que se usó un valor de 30. Beta determina el brillo, el cual no se modifica (se mantiene en 1).

```
contrast_img = cv2.convertScaleAbs(img, 30, 1)
```

2. Únicamente se busca preservar los bordes de la cara del sujeto, por lo que se decide eliminar las áreas más oscuras de la imagen recién modificada. Esto se hace con una operación pixel a pixel que convierte todos los valores menores a 70 directamente a 0, usando la función **np.clip(lim_inferior, lim_superior)**. De esta manera, las partes oscuras (como el abrigo) no se toman en cuenta en la siguiente fase.

```
img_c = np.clip(contrast_img, 70, 255)
```

3. Una vez se preproceso la imagen (contraste y recorte de intensidades), se inicia el proceso de extracción de bordes. Dados los resultados negativos utilizando filtros de convolución para perfilado, directamente se utiliza un filtro de Fourier. El primer paso para utilizar este tipo es pasar la imagen al dominio de frecuencias, lo cual se hace con la función de numpy **np.fft.fft2(img)**. Sin embargo, para aplicar los filtros se requiere que la frecuencia 0 esté en el centro del espectro obtenido, por lo que además se tiene que aplicar la función **np.fft.fftshift(freq_img)**. *OpenCV* cuenta con funciones implementadas para realizar estas mismas tareas, pero la implementación de *NumPy* facilita las operaciones posteriores ya que trabaja un tipo de dato de número complejo en lugar de separar los componentes reales e imaginarios en diferentes matrices, y ambas implementaciones tienen el mismo resultado al final (En el código se incluye una comparación entre ambos).

```
fft_img, fshift = fft_np(img_c)
```

Los bordes y otros cambios abruptos en intensidad se asocian con componentes de alta frecuencia, por lo que se pueden obtener utilizando un filtro pasa-altas. Se implementa un filtro Butterworth pasa-altas basado en la descripción de R. C. González y R. E. Woods (2018). El parámetro D0 determina la frecuencia de corte, y n determina el orden del filtro. Se puede ver como el radio del "círculo" obtenido y la "suavidad" de la transición entre las frecuencias que pasan.

```
HPF = high_pass_butterworth(fft_img, 30, 4)
G_shift = fshift * HPF
```

Finalmente, para que el espectro regrese al dominio de imagen se debe aplicar la transformada inversa. Esto se hace con las funciones opuestas a las del primer paso: **np.fft.ifftshift(shift)** y **np.fft.ifft2(f_ishift)**. También se convierte la imagen a valores reales, ya que el tipo de dato se mantiene como complejo (a pesar de que el coeficiente de i es prácticamente 0, es decir, no hay parte imaginaria). Con esto se obtienen únicamente los bordes de la imagen con contraste aumentado.

```
edges = ifft_np(fshift, G_shift)
```

4. Con los bordes ya extraídos en una imagen aparte, ahora sí es válido aplicar el filtro de mediana en la primera imagen producida (Paso 1) y eliminar el ruido. Este filtro convolucional se aplica usando la función **cv2.bilateralFilter(img, 9, 75, 75)**, indicando un tamaño de kernel de 9x9, un criterio de similitud (*sigmaColor*) de 75, y un criterio de cercanía (*sigmaSpace*) de 75 igualmente (valores por default).

```
noise_reduction = cv2.bilateralFilter(contrast_img, 9, 75, 75)
```

5. Finalmente, se combinan los bordes y la imagen suavizada sumando ambas matrices. Los bordes se ven más ruidosos que el fondo suavizado, así que se "difuminan" ambas partes utilizando un filtro de suavizado estándar (box filter) con la función **cv2.blur(img, (5,5))**, indicando un kernel de 5x5.

```
new_img_v1 = cv2.blur(new_img_v1, (3, 3))
```

C. Rotación

Antes de proceder con el recortado de imagen se decide hacer la rotación, ya que de esta manera al momento de recortar únicamente se necesitará "clippear" columnas y renglones de una matriz, sin preocuparse por la perspectiva.

Para la rotación se utilizó una transformación afín con una matriz de rotación estándar. La matriz se obtiene utilizando **cv2.getRotationMatrix2D ()**, donde se indica el centro (eje de rotación), el grado y la escala. Para aplicar la rotación se usa **cv2.warpAffine()**, que requiere la imagen a transformar y la matriz de rotación anteriormente creada. El ángulo se encontró de manera experimental, probando valores de 0-45 y comparando resultados.

```
rotation_M = cv2.getRotationMatrix2D((M/2, N/2), 19, 1)
rotated_img = cv2.warpAffine(img_a, rotation_M, (N, M))
```

B. Recorte

Gracias a que la rotación se hizo en el paso anterior, recortar la imagen se puede ver como una operación de matrices simple. Python permite utilizar el operador **[:]** (*slice operator*), donde se indica un rango de índices a tomar de una lista. La imagen se maneja como una matriz (lista bidimensional), por lo que únicamente es necesario elegir la posición máxima y mínima de los pixeles en x y en y que abarca la imagen. Esto se hace de manera empírica, con los valores de y [75:N], (se eliminan 75 filas de pixeles de la parte superior), y x [53, 220] (se eliminan 53 columnas a la izquierda y M – 220 columnas a la derecha).

```
cropped_img = img_c[y0:y1, x0:x1]
```

3. Descripción de los resultados

En la siguiente imagen se muestra el efecto que tiene aplicar únicamente el filtro bilateral (5). Se puede observar la perdida de detalle del rostro.

Figura 2: Imagen con filtro bilateral (5)

La siguiente imagen muestra el resultado de los pasos (2-3), con la imagen en el espacio de frecuencias a la izquierda (con el filtro Butterworth ya aplicado) y los bordes extraídos a la derecha.

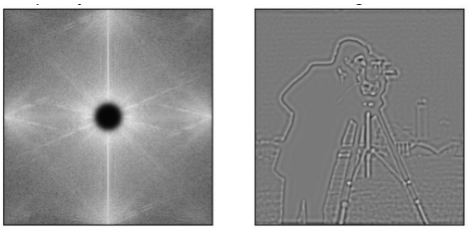

Figura 3: Imagen filtrada en dominio de frecuencias, y resultado del filtro)

Aquí se muestra la imagen resultante del proceso de eliminación de ruido, donde se nota la reducción de ruido, pero sigue siendo aparente el detalle del rostro.

Figura 4: Original vs resultados del punto a.

La siguiente imagen muestra el efecto de la transformación afín de rotación, los bordes que desparecen se convierten en 0 (color negro).

Figura 5: Resultados del punto c.

Finalmente, la imagen final (ya recortada) en comparación de la imagen original. Se incluyen dentro de la carpeta de resultados para una mejor comparación, ya que en el documento no se llega a aprecias el suavizado por completo.

Figura 6: Original vs resultado final

4. Discución de los Resultados

A pesar de que la solución descrita logra cumplir los objetivos, los resultados pudieron mejorar. Lo más importante es que se nota mejor el detalle del rostro, pero quizás probando diferentes kernels laplacianos, o comprobando que les pasaba a los datos dentro de la matriz, pude haber obtenido un mejor perfilado (Se me hizo extraño que los bordes fueran tan burdos en las pruebas con laplacian filters). También me di cuenta al final que rotar una imagen afecta levemente la calidad (mi teoría es que causa un aliasing de las intensidades), así que si hacía el punto A al final quizás mis resultados serían mejores. Intenté solo cambiar de orden, pero sí era necesario además buscar nuevamente los parámetros correctos.

5. Conclusiones

Me siento satisfecho con la solución descrita, especialmente porque problemas que inicialmente aparecían ser sencillos (o incluso triviales) en realidad representaban un reto para poder cumplir con los requerimientos. Específicamente, el punto A. me ayudó a repasar y a poner a prueba tanto lo que vimos en clase como lo que pude consultar en otros sitios. OpenCV es un paquete altamente capaz, y los resultados que se pueden obtener son increíbles. Puede que la interfaz (nombres de funciones, etc.) no tenga tanta cohesión (se nota que es un port de un proyecto de C/C++), pero la realidad es que es una herramienta casi perfecta para este tipo de tareas. En especial me agrada el hecho de que existen varios niveles de abstracción, ya que puedes usar tanto llamadas de alto nivel a filtros completos como entrar directo a aplicar convoluciones o transformaciones casi manuales.

6. Referencias

Gonzalez, R.C. & Woods, R.E. (2018) Digital Image Processing. 4th Edition, Pearson Education, New York.

* Referencias de documentación están en el código