

Tarea de Investigación No. 3

TRANSFORMACIONES GEOMÉTRICAS UTILIZANDO EL CONCEPTO DE MAPEO BILINEAL

Abiel Porras Garro - 2020209597¹, Elías Castro Montero - 2020098930²

¹Ingeniería en Computación, Instituto Tecnológico de Costa Rica, San José, Costa Rica

²Ingeniería en Computación, Instituto Tecnológico de Costa Rica, San José, Costa Rica

I. DESARROLLO

Para los ejemplos se utilizará la siguiente imagen como base:



Fig. 1: Imagen de un paisaje

A. Punto 1.1

Para deducir la fórmula de mapeo inverso de acuerdo con el concepto general de mapeo bilineal, primero iniciar con la fórmula de mapeo bilineal:

$$w = \frac{az + b}{cz + d}$$

donde:

$$\{a, b, c, d\} \in \mathbb{C}$$

De acuerdo con [1] cuando el término $bc - ad$, llamado *determinante del mapeo*, es cero, el mapeo degenera en $w = \frac{a}{c}$, por lo que no tiene mapeo inverso, ya que dados los valores de a, c , sin importar cual sea el valor de z siempre se obtendrá el mismo valor w , por lo que cualquier valor de z mapeará al mismo punto en w .

Por otro lado, si el determinante del mapeo es distinto de cero, entonces el mapeo inverso existe, y está dado por el siguiente mapeo inverso, que también es invertible [1]:

$$z = \frac{-dw + b}{cw - a}$$

Con base en lo anteriormente descrito, es posible identificar que solo basta con definir un caso en el que la función de mapeo $w = f(z)$ sí tiene mapeo, este es, cuando la determinante del mapeo es distinto de cero. Por lo que se deduce la siguiente condición que cumplen los $w = f(z)$ que

tengan mapeo inverso:

$$bc - ad \neq 0 \Leftrightarrow z = f(w) = \frac{-dw + b}{cw - a}, \text{ con } \{a, b, c, d\} \in \mathbb{C}$$

Esto se puede describir con la siguiente fórmula,

$$z = \frac{-dw + b}{cw - a}, bc - ad \neq 0$$

donde:

$$\{a, b, c, d\} \in \mathbb{C}$$

B. Punto 1.2

La siguiente función recibe como entradas las constantes complejas a, b, c, d y retorna `True` si estas generan una función de variable compleja que tenga mapeo inverso o `False` si su mapeo inverso no existe; para lo cual se utiliza la condición/caso deducido en el punto anterior, esto es que su determinante del mapeo, $bc - ad$ sea diferente de cero.

```
def tiene_mapeo_inverso(a, b, c, d):
    return b*c-a*d != 0
```

C. Punto 1.3

La función `newImage()` recibe la imagen y las constantes complejas como entradas para generar una nueva representación en el plano w . Primeramente mediante la función `new_size()` que realiza el mapeo bilineal de los bordes de la imagen para obtener el tamaño de la nueva imagen que se va a generar.

Posteriormente se realiza un bucle donde se pasa por todos los nuevos píxeles y mediante mapeo inverso se obtiene el valor del píxel en la imagen original. Se debe verificar que el píxel que se está evaluando este dentro de los píxeles de la imagen original en el plano z . Finalmente se retorna la nueva representación creada.

```
def newImage(img, a, b, c, d):
    new_width, new_height = new_size(img, a, b,
                                      c, d)
    mapeo = crear_mapeo_inverso(a, b, c, d)
    new_img = np.zeros((new_width, new_height),
                      np.uint8)
```

```

for x in range(new_width):
    for y in range(new_height):
        z = mapeo(complex(x, y))
        new_x = int(z.real)
        new_y = int(z.imag)
        if 0 <= new_x < img.shape[0] and 0 <=
            new_y < img.shape[1]:
            new_img[x, y] = img[new_x, new_y]
return new_img

```

Para el siguiente ejemplo se utilizaron los valores en las constantes complejas: $a = 3 + 2.1j$; $b = 0$; $c = 0.004$; $d = 4 + 6j$.

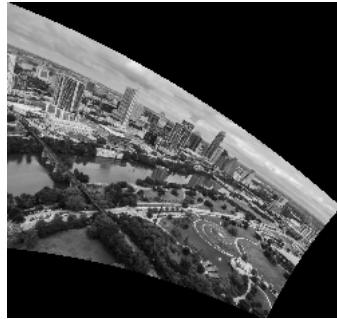


Fig. 2: Nueva representación obtenida

D. Punto 1.4

Para realizar los siguientes ejemplos se realizó una función que solo recibe como entrada la imagen y las constantes complejas a y b . Las constantes c y d tienen como valor 0 y 1 respectivamente con el fin de solo afectar el escalado, rotación y desplazamiento de la nueva representación.

```

def newImageAB(img, a, b):
    new_img = newImage(img, a, b, 0, 1)
    return new_img

```

1) Punto 1.4.1

Para este ejemplo se debía utilizar un valor real en la constante a y el valor 0 en la constante b , por lo que se utilizaron los valores $a = 3$; $b = 0$. Como resultado la cantidad de píxeles se incrementó de 800x416p a 2397x1245p.



Fig. 3: Ejemplo para el punto 1.4.1

2) Punto 1.4.2

Para este ejemplo se debía utilizar un valor complejo en la constante a y el valor 0 en la constante b , por lo tanto se utilizaron los valores $3+1j$; $b = 0$. Como resultado la cantidad

de píxeles se incrementó de 800x416p a 2812x1245p, y se giró la imagen hacia la izquierda.

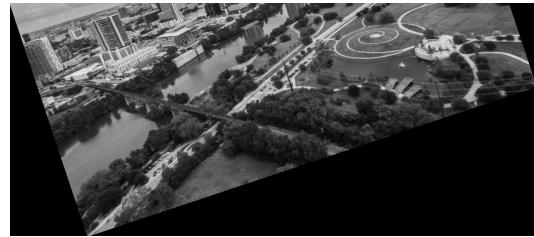


Fig. 4: Ejemplo para el punto 1.4.2

3) Punto 1.4.3

En este ejemplo el valor de a debe ser 1 y el valor de b debe ser distinto de 0 para comprobar el desplazamiento de la imagen. Para esta representación se utilizaron los valores $a = 1$; $b = 600 + 450j$. La imagen se desplazó hacia abajo y a la derecha.

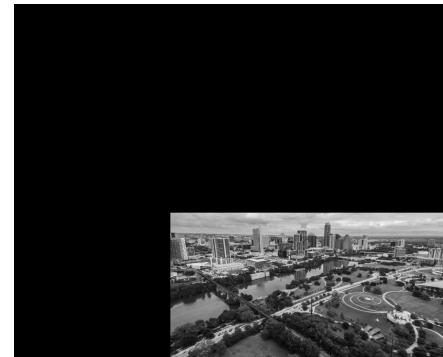


Fig. 5: Ejemplo para el punto 1.4.3

4) Punto 1.4.4

Finalmente, para este ejemplo los valores de a y b deben ser distintos de 0, por eso se les asignó los valores $a = 3+1j$; $b = 400 + 300j$. La cantidad de píxeles cambió a 3112x1645p, la imagen giro hacia la izquierda, y se desplazó hacia abajo y a la derecha.



Fig. 6: Ejemplo para el punto 1.4.4

E. Punto 2

En este punto a diferencia del método anteriormente explicado en los anteriores puntos, la nueva representación en el plano w se genera primeramente mediante mapeo directo

con las constantes complejas obtenidas. Las constantes que se utilizaron en este ejemplo y en el siguiente de ellos serán: $a = 2.1 + 2.1j$; $b = 0$; $c = 0.003$; $d = 1 + j$. La siguiente imagen es el resultado obtenido con estos valores.

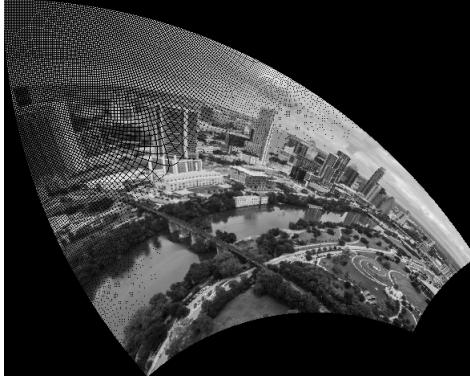


Fig. 7: imagen2 nueva representación

Como se puede observar mediante este método se crea un patrón de píxeles negros debido a la falta de información en esas áreas. Esto es debido a que la representación se estira en esa zona y en la imagen original no cuenta con esos píxeles para llenar toda la zona por lo que se generan estos problemas.

F. Punto 3

En este ejemplo para obtener los píxeles faltantes en la nueva representación se utilizó la técnica de mapeo inverso que consiste en pasar por todos los píxeles sin valor y obtener su valor de coordenadas en la imagen original y dárselas a este píxel. Para saber si el píxel tiene algún valor obtenible basta con identificar si los valores de coordenadas obtenidos están dentro de los límites de la imagen original, si este es el caso entonces se le asigna el valor obtenido.



Fig. 8: imagen3 mapeo inverso

En la imagen obtenida se puede notar como se recupera toda la información. Sin embargo dependiendo del área alguna información se ensucia y no queda de todo claro la diferenciación de límites en algunas partes de la imagen.

G. Punto 4

Para este ejemplo se utilizó la técnica de interpolación con $N = 4$ que consiste en obtener los valores de las

coordenadas del píxel faltante en la imagen original para saber su localización. Luego se obtienen los valores de cuatro píxeles adyacentes al píxel original y se hace un promedio de estos.

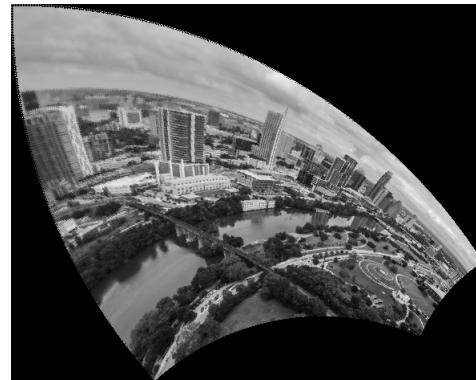


Fig. 9: imagen4 interpolación 4

En la imagen a diferencia de la obtención solo mediante mapeo inverso se puede ver un poco más de claridad en las zonas más delimitadas, aunque la obtención de los valores del borde de la imagen son un problema debido a que falta información para completar el promedio.

H. Punto 5

En este ejemplo la técnica utilizada es prácticamente la misma que la anterior con la diferencia de que ahora no solo se obtendrán los valores de cuatro píxeles adyacentes, sino que ahora se obtendrá el valor de ocho de estos y se promediaran para darle el valor al píxel faltante. Esta técnica es llamada interpolación con $N = 8$.



Fig. 10: imagen5 interpolación 8

Como se aprecia en la imagen esta es la técnica que nos da un resultado más limpio donde se pueden apreciar mucho mejor la información faltante y hace que los bordes de algunas regiones sean mucho más claros. No obstante, cuenta con el mismo problema de la técnica de interpolación anterior donde obtener los valores de los límites es bastante complicado.

I. Punto 6

Para aplicar filtro Gausseano, se puede hacer uso de la función `cv2.GaussianBlur`, la cual permite realizar el

proceso de suavizado de imágenes [2]. La firma de esta función es `cv2.GaussianBlur(img, mascara, sigma)`, donde el primer parámetro corresponde a la imagen que se desea suavizar, el segundo es la máscara que se desea usar, representada con tuplas, por ejemplo `(5, 5)` si se desea aplicar una máscara de 5×5 , y por último el sigma, es decir, la desviación estándar de la distribución Gausseana, si se establece como 0, OpenCV calculará automáticamente el valor de la desviación estándar de acuerdo a la máscara establecida. Un ejemplo de invocación de la función es `cv2.GaussianBlur(image, (3, 3), 0)` [2].

En nuestro caso, envolvimos la función de OpenCV en una nueva función, que recibe la imagen y el rango de la máscara, en nuestro caso será 5, ya que la máscara es de 5×5 , por último se utiliza 0 como valor fijo para la desviación estándar para que sea OpenCV el que calcule la desviación estándar que deberá tener la distribución normal utilizada:

```
def addGaussianBlur(img, range):
    img = cv2.GaussianBlur(img, (range, range),
                          0)
    return img
```

En este punto 6, se debe aplicar un filtro Gausseano con una máscara de 5×5 , sobre imagen2 representada en la figura 11. El resultado de este filtro se puede observar en la figura 14

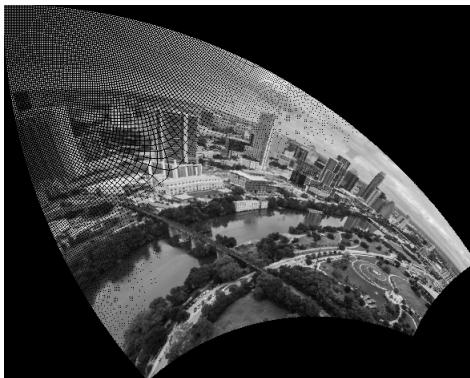


Fig. 11: imagen2 original

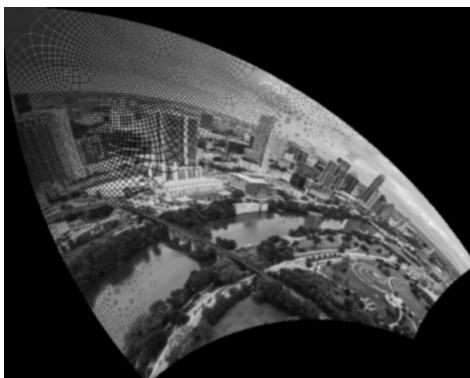


Fig. 12: imagen2 con el filtro Gausseano aplicado

En la subsección I-L se describirán los resultados de aplicar

filtro Gausseano, con respecto a los resultados obtenidos en las siguientes imágenes

J. Punto 7

Se debe aplicar un filtro Gausseano con una máscara de 5×5 , sobre imagen4 representada en la figura 13. El resultado de este filtro se puede observar en la figura ??

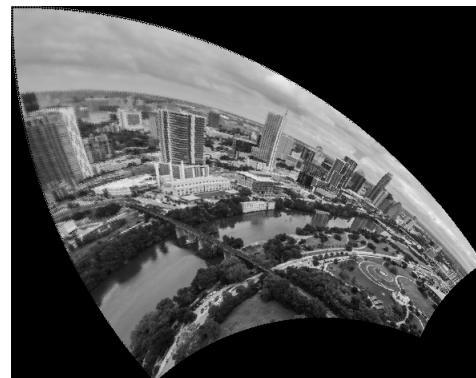


Fig. 13: imagen4 original

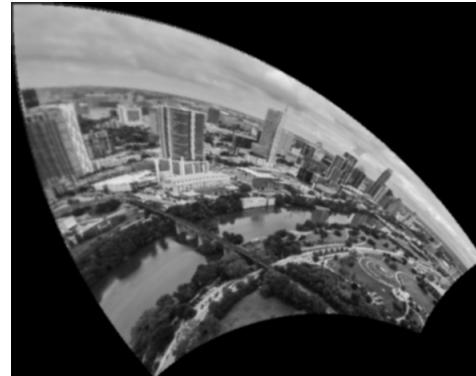


Fig. 14: imagen4 con el filtro Gausseano aplicado

K. Punto 8

Se debe aplicar un filtro Gausseano con una máscara de 5×5 , sobre imagen5 representada en la figura 15. El resultado de este filtro se puede observar en la figura 16



Fig. 15: imagen5 original



Fig. 16: imagen5 con el filtro Gausseano aplicado

L. Punto 9

En las imágenes obtenidas con las tres técnicas de para obtener los píxeles faltantes se evidencia que la tarea de obtener esta información es bastante complicada y nunca es perfecta. Sin embargo, la que más se acerca a dar un resultado con un acabado limpio y con mucha más calidad es la técnica de interpolación con $N = 8$ donde se notan bordes mucho mejor logramos e información más clara. Las dos técnicas restantes también logran buenos resultados, pero en las pruebas realizadas es indudable que no tienen mejor calidad que la primera al perder muchos más detalles de la información original.

Con respecto a las tres imágenes sobre las que se aplicó el filtro Gausseano, como es posible observar, este proceso genera un suavizado sobre la imagen, lo cual es más sencillo de notar en los bordes, que antes tenían un efecto de escalera, y ahora tienen un efecto de bordes continuos. Sin embargo, una desventaja de aplicar este filtro, es que la imagen es más borrosa, y cuesta identificar más los detalles como las ventanas de los edificios.

Desde nuestro punto de vista, la imagen4 es la que tiene mejor calidad, ya que el mapeo inverso permite obtener valores de píxeles faltantes, lo que elimina el patrón de píxeles en la parte superior sin color alguno que se observa en las primeras imágenes cuando solo se usaba mapeo bilineal, mientras que la interpolación logra realizar una buena interpretación de la

información ausente que debería estar contemplada en la nueva representación.

Por otro lado, el uso de filtro Gausseano, más bien provocó un efecto de desenfoque o difuminado que hace ver a las imágenes de menos calidad, y que no nos termina de convencer, aunque cabe destacar que suaviza con un alto nivel de precisión los bordes de la imagen, sin embargo, esto no es suficiente para que consideremos las últimas 3 imágenes como de mejor calidad

II. CONCLUSIÓN

- Existen una amplia variedad de recursos matemáticos que pueden emplearse en área que en principio uno no pensaría, como en este caso la edición de imágenes digitales, sin embargo, si entendemos a las imágenes como arreglos bidimensionales de píxeles discretos, podremos aplicar una gran cantidad de conceptos, dentro de ellos, los números complejos, y como en este caso, mapeo bilineal e inverso, que permiten realizar transformaciones en principio complicadas como transformaciones, rotaciones y magnificaciones, de forma sencilla, siendo suficiente con cambiar el valor de variables o incógnitas en las fórmulas matemáticas, para alcanzar los resultados deseados.
- No siempre será necesario aplicar una gran cantidad de algoritmos complejos en el procesamiento de una imagen, para nosotros bastó con aplicar mapeo inverso e interpolación para obtener una transformación de la imagen original, de alta calidad. El hecho de aplicar filtro Gausseano, no permitió mejorar la calidad de la transformación, por el contrario, hizo que esta se viese más borrosa, lo cual en nuestro caso, está lejos de lo que se desea. No obstante, a pesar de que el filtro Gausseano no nos diese los resultados deseados, no lo podemos descartar, ya que en otros escenarios, podría ser de gran utilidad, como vimos en el trabajo anterior, el filtro Gausseano se utiliza como paso intermedio en el algoritmo Canny Edge Detection para la detección de bordes.
- La resolución y definición de la imagen siempre será un factor importante a tomar en cuenta a la hora de realizar técnicas de recuperación de información, debido a que es mucho más fácil para los algoritmos obtener datos detallados de una imagen de alta calidad; con los que se pueda reconstruir mejor la imagen original. Cuando una imagen tiene pocos píxeles, este tipo de algoritmos posiblemente terminen ensuciando la nueva representación debido a que información incorrecta se puede filtrar en donde no debe, por ejemplo en el algoritmo de interpolación donde la información es obtenida con píxeles distintos al original, los datos son fácilmente alterables, ya que se utiliza la información de los píxeles que se encuentren alrededor de otros para llenar píxeles faltantes.
- Es complejo obtener información nueva de una imagen existente debido a que los píxeles no guardan más que un valor muy cerrado que representa tan solo un dato de una matriz, la cual junta, o vista como un todo, sí presenta una gran cantidad de información. Debido a

esto los algoritmos siempre se verán limitados a trabajar con los datos que ya cuentan para lograr obtener una representación completa con los datos que logre generar, intentando que sean lo más cercanos al verdadero, pero difícilmente logrando que sean los reales, ya que son solo aproximaciones, unas más precisas que otras.

REFERENCES

- [1] P. Moya, *Señales y Sistemas. Fundamentos Matemáticos*, 1st ed. Costa Rica: Instituto Tecnológico de Costa Rica, Centro de Desarrollo de Material Bibliográfico, 2008.
- [2] A. Rosebrock, “OpenCV Smoothing and Blurring,” 2021. [Online]. Available: <https://pyimagesearch.com/2021/04/28/opencv-smoothing-and-blurring/>