Inpainting de imágenes RGB preservando textura del fondo y bordes lineales

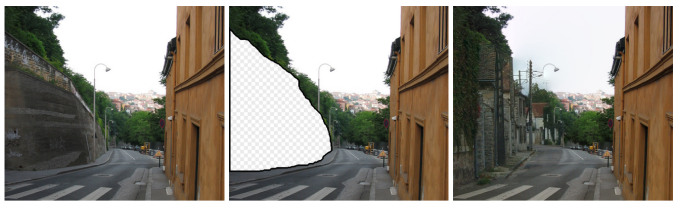
Daniel Acuña U.

***Abstract*—El inpainting de imágenes consiste en la eliminación imperceptible de un objeto indeseado en una imagen. Dentro de sus aplicaciones está el procesamiento de imágenes biomédicas y mejoramiento de imágenes publicitarias entre otros. En este artículo se presenta un algoritmo focalizado en el impainting de objetos inmersos en un fondo de líneas paralelas conservando texturas presentes en la imagen. Se distinguen cuatro etapas principales del algoritmo: Acondicionamiento de la imagen y reducción de ruido, detección de bordes, reconstrucción de bordes y finalmente el relleno del objeto indeseado. Se muestra una aplicación del algoritmo retirando exitosamente el objeto de la imagen y se incorpora una serie de mejoras para robustecer el rendimiento del algoritmo en distintas situaciones.**

***Index Terms*—Edge-preserving smoothing, Inpainting, Image Processing.**

I. Introducción

La reconstrucción de partes de una imagen de forma indetectable se conoce como Impainting. El impainting puede ser utilizado para reconstruir zonas dañadas o quitar objetos indeseados como se muestra en la figura 1.



Original Input

Fig. 1. Imagen reconstruida por [1]

Frente a diversas investigaciones referentes al tema podemos clasificarlas en dos grandes grupos. El primer grupo es impainting basado en difusión de la imagen. Este tipo de impainting rellena agujeros de la imagen mediante el uso de ecuaciones diferenciales parciales. Es un método que funciona bien cuando los agujeros son pequeños y/o el contorno del agujero consiste en estructuras sencillas como lineas y curvas bien delimitadas. El segundo grupo es impainting basado en reconocimiento de similitudes en la imagen. Consiste en considerar otras regiones de la imagen, reconocer patrones de su textura mediante una descripción estadística y luego replicarla en la zona que se desea reconstruir de una nueva forma o copiarla y pegarla directamente. En este artículo se presenta un algoritmo focalizado en el impainting de objetos inmersos en un fondo de líneas paralelas conservando texturas presentes en la imagen. Se distinguen cuatro etapas principales del algoritmo: Acondicionamiento de la imagen y reducción de ruido, detección de bordes, reconstrucción de bordes y finalmente el relleno del objeto indeseado. Se muestra una aplicación del algoritmo retirando exitosamente el objeto de la imagen y se incorpora una serie de mejoras para robustecer el rendimiento del algoritmo en distintas situaciones.

II. Metodología

A. Acondicionamiento y reducción de ruido

Considerando que la imagen posiblemente contenga ruido, se simula dicha situación incorporando un ruido Gaussiano a la imagen. Su implementación en python se realiza mediante el método `random_noise` de la libreria `skimage.util` que suma en cada canal un set de datos que distribuyen normal con parámetros de media y desviación estándar dados.

El acondicionamiento de la imagen se realiza mediante un filtro Gaussiano pasa bajos. Matemáticamente en el espacio de la frecuencia se define como:

$$H_{gauss}(u,v) = e^{-D^2(u,v)/2D_0^2} \tag{1}$$

donde (u, v) corresponden a los pares coordenados del espacio transformado de Fourier, $D(u, v)$ corresponde a la distancia euclidiana de un par coordenado al origen y D_0 es la frecuencia de corte del filtro pasa bajos. La figura 2 muestra una máscara de filtrado pasa bajos.

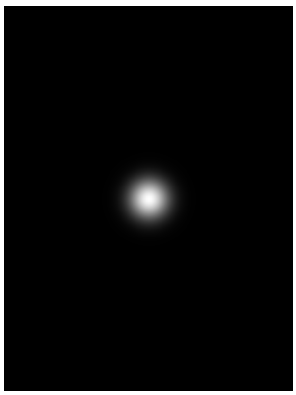


Fig. 2. Ejemplo de máscara de filtro Gaussiano en el espacio de Fourier

Se distinguen dos objetivos con este proceso:

- Reducir el ruido presente en la imagen: se ha comprobado en [2] que el filtro Gaussiano es muy efectivo para reducir el ruido Gaussiano
- Reducir incidencia de la textura en la detección de bordes de la imagen: Se busca eliminar sustancialmente la textura de la imagen para que en el siguiente paso del algoritmo se dificulte menos la detección de lineas presentes en la imagen.

B. Detección de bordes

Los bordes son detectados por una variación del algoritmo detector de bordes de Canny.

1) Gradiente de la imagen: En cada canal se calcula el gradiente en las direcciones horizontal (G_x) y vertical (G_y) convolucionando los operadores de sobel con cada canal.

$$g_x = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad g_y = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{2}$$

La matriz de magnitudes del gradiente se calcula mediante la ecuación (3):

$$M(x, y) = \sqrt{(g_x * f)^2 + (g_y * f)^2} \tag{3}$$

donde f representa la imagen.

2) Umbralización: En cada posición de las matrices de magnitud de gradiente se verifica si superan un umbral determinado. Si alguno de los tres lo logra, se asigna un uno en esa posición. De lo contrario se asigna un cero.
3) Eliminación de estructuras aisladas: Se retiran objetos aislados que no conforman parte de lineas de borde. El algoritmo específico corresponde a "Hit-or-Miss" y se implementa en python con el método `remove_small_objects`.
4) Definición de bordes: Considerando que sólo quedan elementos de la estructura de las lineas de bordes, se procede a erosionar y adelgazar a fin de definir los bordes. Los métodos de python utilizados para ambos procesos corresponden a `binary_erosion` y `thin` respectivamente. Se asegura que los procesos de erosión y adelgazamiento no debiliten las lineas en exceso con una dilatación final.

C. Reconstrucción de bordes

La reconstrucción de bordes ocultos por el objeto indeseado se lleva a cabo mediante una parametrización de lineas con la Transformada de Hughs. Este algoritmo parametriza lineas rectas mediante un ángulo θ medido respecto a la dirección horizontal y la distancia al origen de la imagen ρ. Recorre la diagonal de la imagen variando el ángulo. Finalmente se obtienen una lista de pares (ρ, θ) con la mayor coincidencia de puntos. Obtenidas estas rectas, se extienden a través de toda la imagen. La sensibilidad de este algoritmo se puede regular según la imagen particular. De este proceso se obtiene una nueva máscara que contiene los bordes del fondo de la imagen.

D. Relleno de objeto indeseado

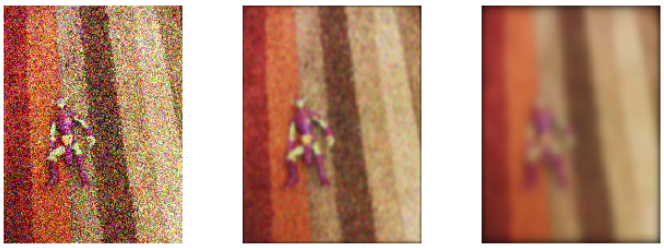
1) Búsqueda de ventanas representantes: Dado que el algoritmo está enfocado en imágenes con bordes del fondo de lineas rectas, se puede encontrar una ventana representativa de cada rectángulo del fondo de la imagen. Dichas ventanas se obtienen recorriendo los espacios entre lineas de borde del paso anterior. Se calcula la distancia entre lineas para definir el punto medio entre ambas. Respecto a ese punto medio entre lineas, se guarda una ventana entorno a ese punto. El algoritmo pasa al siguiente intervalo de lineas hasta que recorre todas las lineas de la imagen a una altura en que no está el objeto.
2) Rellenado de objeto indeseado: Se recorren los espacios delimitados por el objeto indeseado. Se pinta el punto visitado tomando un valor al azar de la ventana representativa de la posición respectiva. El proceso aleatorio sigue una distribución normal.

III. Resultados

El algoritmo se aplica a una imagen de (400, 300) pixceles. El tiempo de ejecusión es de 4.917382717132568 segundos.

A. Acondicionamiento y reducción de ruido

La figura 3 muestra la imagen con ruido añadido y el resultado de dos filtros. El resultado de la sub-figura (b) se utiliza para aplicar el inpainting y la sub-figura (c) se utiliza para aplicar el algoritmo de detección de bordes.

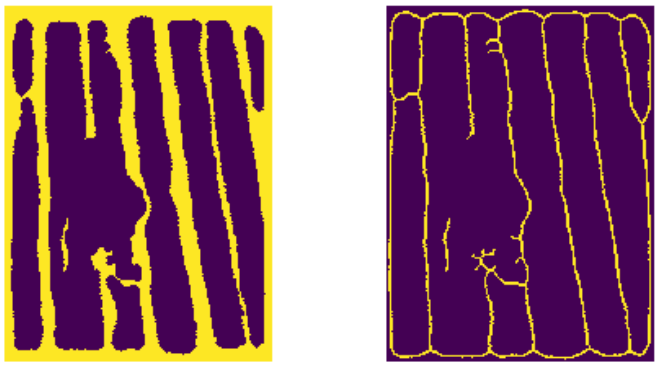


(a) Imagen ruidosa (b) Filtrado para inpainting (c) Filtrado para bordes

Fig. 3. Resultados del proceso de filtrado.

B. Detección de bordes

La figura 4 muestra resultados parciales del proceso de detección de borde. La sub-figura (a) presenta el resultado de umbralización del gradiente de la imagen. La sub-figura (b) muestra el resultado final.

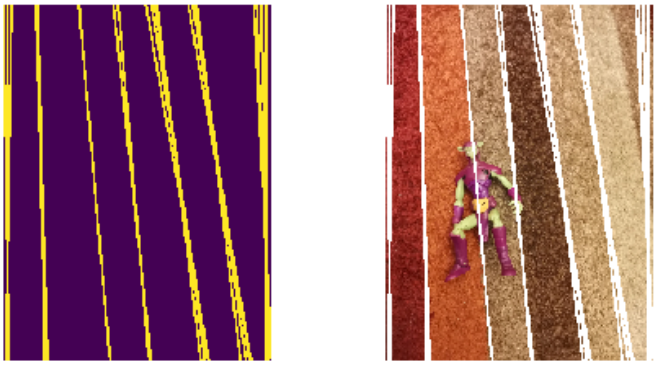


(a) Gradiente umbralizado (b) Bordes definidos

Fig. 4. Resultados del proceso de detección de bordes.

C. Reconstrucción de bordes

La figura 5 muestra resultados parciales del proceso de reconstrucción de bordes. La sub-figura (a) presenta el resultado del proceso de parametrización por Transformada de Hughs. La sub-figura (b) muestra la superposición de los bordes detectados con la imagen original.

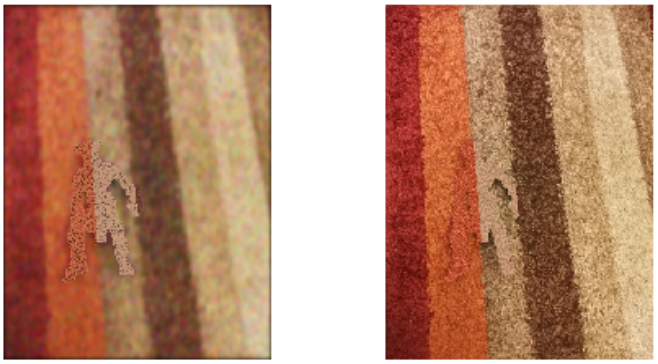


(a) Bordes reconstruidos (b) Bordes en la imagen original

Fig. 5. Resultados del proceso de reconstrucción de bordes.

D. Rellenado de objeto indeseado

La figura 6 muestra resultados parciales del proceso de rellenado. La sub-figura (a) presenta el resultado del algoritmo completo en la imagen ruidosa. La sub-figura (b) muestra el resultado del algoritmo completo en la imagen sin ruido.



(a) Inpainting con ruido (b) Inpainting sin ruido

Fig. 6. Contraste del efecto del ruido en el resultado final de Inpainting.

IV. Análisis y Conclusiones

El algoritmo cumple con el objetivo de completar la imagen a cabalidad respetando los bordes del fondo de la imagen. Junto con ello, genera una textura similar a la textura presente alrededor del objeto. Sin embargo, se puede notar una leve desmejoría de la preservación de la textura por efecto del ruido en la imagen.

Respecto a la detección de bordes y cálculo del gradiente de la imagen se pueden realizar algunas variaciones. En el diseño del algoritmo se probaron al menos tres formas distintas de obtener la umbralización del gradiente total de la imagen. Algunas de las opciones que se estudiaron fueron sumar los tres canales y saturar por el valor máximo posible. Otra opción que se estudió fue elegir el máximo valor entre los tres canales y posterior a ello umbralizar. La opción que mejor logra segmentar rectas es la descrita en la sección de Metodología.

Si bien los resultados son satisfactorios el algoritmo presenta una baja autonomía en su implementación producto de una serie de parámetros que se deben regular para obtener buenos resultados. Dentro de estos parámetros se encuentran las frecuencias de corte de ambos filtrados Gaussianos, la sensibilidad del algoritmo ”Hit-or-Miss” del proceso de detección de bordes y la sensibilidad de la parametrización por Transformada de Hughs en el proceso de Reconstrucción de bordes.

El algoritmo requiere que se le ingrese un objeto previamente segmentado. Para futuras versiones se espera que lo realice de una forma un poco más automática mediante algún método como el método de Ótsu.

References

[1] J. Hays and A. A. Efros, “Scene completion using millions of photographs,” *ACM Transactions on Graphics (SIGGRAPH 2007)*, vol. 26, no. 3, 2007.

[2] C. Chen. (2020) Medium digital image processing using fourier transform in python. [Online]. Available: https://medium.com/@hicraigchen/digital-image-processing-using-fourier-transform-in-python-bcb49424fd82

Daniel Acuña Estudiante de Ingeniería UC, Mayor en Ingeniería en Robótica, Minor en Ingeniería en Sistemas de Automatización. Pontificia Universidad Católica de Chile.