

Informe Tarea 3A

Efecto Bloom

Integrantes: Valentina Garrido
Profesor: Nancy Hitschfeld
Auxiliar: Andrés Cerda
Pablo Pizarro
Alonso Utreras
Ayudantes: Tomás Calderón
Beatriz Grabolosa
Sebastián Olmos
Nadia Decar

Fecha de realización: 5 de enero de 2021
Fecha de entrega: 5 de enero de 2021
Santiago, Chile

1. Solución propuesta

La solución se realiza en varias etapas. Primero, se identifican todos los píxeles con el color correspondiente al entregado y se almacenan sus índices fila-columna en un conjunto (set). Este conjunto corresponde a los píxeles de las condiciones de borde. Se prefiere trabajar con los índices más que con una matriz de ceros y unos ya que se ha supuesto que los píxeles con cierto valor serán más escasos que el total de píxeles.

Como en esta implementación se usa norma 1 para la distancia, se genera una máscara de índices relativos a un píxel central, que tiene coordenadas (0,0), donde cada píxel relativo está a lo más a distancia N del píxel central, es decir, si i es la fila y j es la columna de cada píxel relativo cumple la siguiente desigualdad:

$$|i| + |j| \leq N$$

Donde los índices de los píxeles relativos cumplen $(i, j) \in \mathbb{Z}^2 / (0, 0)$, porque se excluye el punto del centro.

Para obtener los vecinos de un píxel dado, simplemente esta máscara se traslada, sumándole las coordenadas de los píxeles relativos a las coordenadas del píxel de interés. Luego, las coordenadas obtenidas se agregan a un nuevo set de índices, que conformarán las variables sobre las cuales se construirá el sistema más tarde.

Debe corroborarse que el índice obtenido no esté en el set de píxeles de condiciones de borde, ni tampoco debe estar ya agregado en el set de variables.

El algoritmo que escanea los vecinos alrededor de los píxeles condiciones de borde es ingenuo. El tiempo de ejecución es alrededor de $O(\text{ancho imagen} \times \text{largo imagen} \times \text{área de la máscara de diamante})$, donde el área de la máscara es menor y cercana a $(2N)^2$. Por esta razón, y para no empeorar el tiempo de ejecución, usar conjuntos para almacenar los índices se hace necesario, ya que la operación de chequear si el índice está en un conjunto toma tiempo constante $O(1)$ y mejora el tiempo de la misma operación para una lista.

Luego, se pasa el conjunto de índices de las variables a lista para ordenarlos por fila-columna, y se crea un diccionario donde se realiza el mapeo unidimensional a cada variable, es decir, se le asigna un entero a cada tupla de posición.

El sistema matricial por cada componente RGB a resolver es:

$$Ax = c \cdot B$$

Donde A será la matriz con el sistema de ecuaciones correspondiente a cada variable, en x se guardarán las soluciones de los brillos obtenidos, c es un escalar que corresponde a una componente RGB del color de la condición de borde y B contiene los coeficientes correspondientes a cuántos píxeles condiciones de borde habían cerca de cada variable.

Para representar la matriz con las variables se usan matrices sparse, una estructura que es capaz de resolver ecuaciones lineales más eficientemente cuando muchas de sus entradas son 0, como es en este caso. Una entrada de una matriz sparse se representa como un triplete fila-columna-valor.

Cada variable se rige por la ecuación de Laplace

$$\Delta f = 0$$

Que en forma discreta se puede aproximar, para una variable u en posición (i,j) , como:

$$\frac{u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}}{h^2} = 0$$

Como en este caso, h es la distancia píxel a píxel, $h = 1$ simplemente.

Para cada variable, entonces, se analiza el stencil de puntos a la izquierda, derecha, abajo y arriba. Como la forma del conjunto de puntos no es fija, se pueden dar muchos casos:

diagrams/borde_{N=2}width = 10cmvariablesconradioN=2

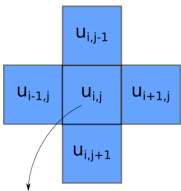
Figura 1:

(b)

Aquí se ilustra que para las variables que están sobre el radio de difusión (en este caso, $N=2$), los variables que están fuera del radio y dentro de la imagen se les asigna simplemente un valor $u = 0$. Como no forman parte de las variables del sistema, simplemente no se agregan a la matriz.

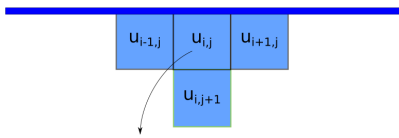
También se muestra el sistema para una variable contigua a una condición de borde, que pasa al lado derecho de la igualdad con un coeficiente negativo.

También se muestra una variable contigua a un píxel con condi



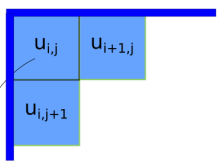
$$u_{i,j+1} + u_{i,j-1} + u_{i+1,j} + u_{i-1,j} - 4 u_{i,j} = 0$$

(c) stencil normal



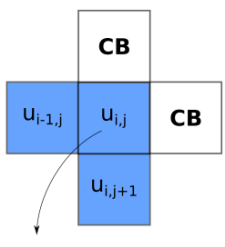
$$2 u_{i,j+1} + u_{i-1,j} + u_{i+1,j} - 4 u_{i,j} = 0$$

(d) stencil borde



$$2 u_{i,j+1} + 2 u_{i+1,j} - 4 u_{i,j} = 0$$

(e) stencil esquina



$$u_{i,j+1} + u_{i-1,j} - 4 u_{i,j} = -2c$$

(f) stencil con dos CB contiguos

Figura 2: Tipos de colisiones

En la imagen (a) se ve el stencil normal de cinco puntos y el sistema asociado. En las imágenes (b) y (c), sin embargo, el stencil cambia y se reduce porque en ciertas posiciones se alcanza el borde

de la imagen y no hay píxel. Entonces, el valor faltante se aproxima como la variable en la posición opuesta, ponderado por dos.

En la imagen (d) se ilustra que la ponderación del píxel de condición de borde aumenta según cuántos píxeles de condición de borde hay contiguos a una variable.

Estos son sólo algunos ejemplos de las posibilidades que se pueden dar y otros casos pueden ser combinaciones de los anteriores.

Tras resolver los tres sistemas para cada componente, lo que resultan son los brillos relativos. Para obtener el resultado final, se suman estos brillos relativos a los componentes de la matriz de imagen y se exporta el resultado.

1.1. Alto Rango Dinámico

Es probable que tras sumar los brillos relativos muchas componentes RGB del color de un píxel superen el valor máximo 1.0 y entren en lo que se llama alto rango dinámico (o high dynamic range, HDR en inglés). Estos colores deben ser convertidos de vuelta al rango $[0,1]$ para ser válidos, sin embargo.

El script ofrece dos soluciones: un mapeo tonal simple iguala todas las componentes que exceden el valor 1.0 a 1.0, o hacer un mapeo tonal de Reinhard, que pasa los colores a bajo rango dinámico (LDR) usando el máximo valor de luminosidad encontrado en la imagen para cambiar la luminosidad de los píxeles acorde a dicho valor.

La luminosidad L de un color RGB es un escalar que se puede obtener por la relación:

$$L = 0.2126R + 0.7152G + 0.0722B$$

Entonces, si L_{old} representa la luminosidad del color antes del mapeo, y L_{max} es la luminosidad máxima de la imagen, la nueva luminosidad L_{new} del color se calcula como:

$$L_{\text{new}} = \frac{L_{\text{old}} \cdot \left(1 + \frac{L_{\text{old}}}{(L_{\text{max}})^2}\right)}{1 + L_{\text{old}}}$$

Finalmente, si c_{old} es el color antes del mapeo, entonces c_{new} el color resultante se puede obtener por una simple relación de proporción:

$$c_{\text{new}} = c_{\text{old}} \frac{L_{\text{new}}}{L_{\text{old}}}$$

De esta manera, se distribuyen los colores de manera más uniforme en el bajo rango dinámico y se preservan detalles que con el mapeo simple podrían aparecer más brillantes y cercanos a blanco y por ende perderse.

2. Instrucciones de ejecución

El programa se ejecuta con los siguientes argumentos:

```
1 python bloom_effect.py [-h] [--reinhard] image_filename N R G B
```

Donde `bloom_effect.py` es el nombre del script, **image_filename** es la ruta con el nombre de la imagen a la que se desea aplicar el efecto bloom, **N** es el radio de la difuminación y los valores **R**, **G**, **B** corresponden a las componentes RGB para identificar los píxeles con dicho color desde los cuáles se quiere difuminar. Cada componente se da con un rango de [0,255].

Los argumentos en brackets son opcionales. `[-h]` muestra en la consola una ayuda respecto a cómo usar los comandos y `[-reinhard]` realiza un mapeo de de Reinhard de los píxeles de la imagen de salida para reducir las componentes RGB de los píxeles a [0,1]. Si no se coloca esta opción, los píxeles se llevan al rango [0,1] simplemente igualando a 1.0 los valores que exceden 1.0 (clamping).

La salida del programa es una imagen de mismo formato de la imagen original y con el mismo nombre, pero con `_out` al final del nombre. La ruta de la imagen de salida también es igual al de la imagen de entrada. Por ejemplo, si la imagen de entrada se dio por la ruta `examples/sample_image.png`, la imagen de salida estará en la ruta `examples/sample_image_out.png`.

3. Resultados

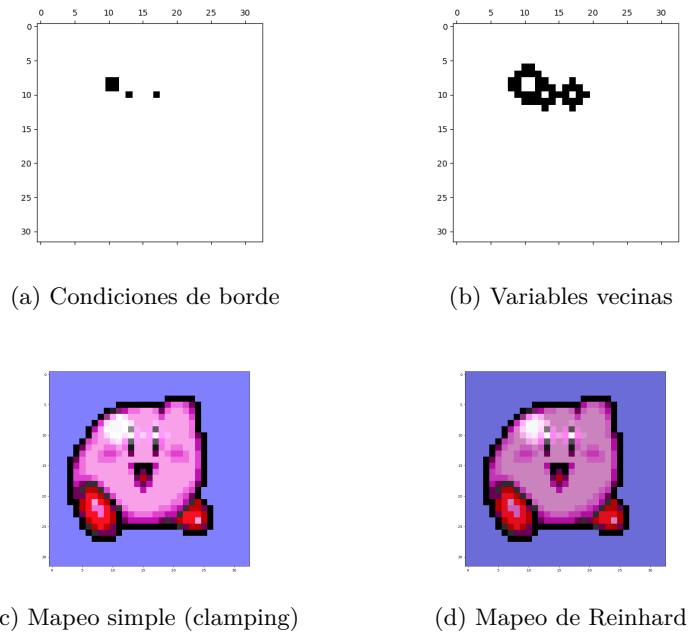
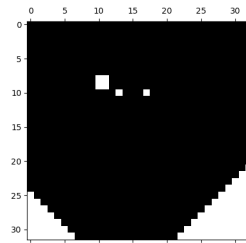
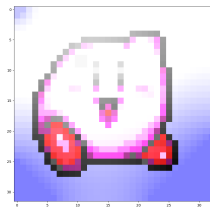


Figura 3: Ejemplo con imagen pequeña

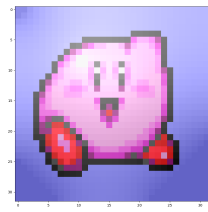
Resultados de aplicar el efecto bloom a una imagen de dimensiones 33×32 y para el color RGB $[248, 248, 248]$ y $N = 2$. En este caso, se observa que para el mapeo de Reinhard la máxima luminosidad causa que la imagen completa se oscurezca ligeramente. El clamping simple parece ofrecer mejores resultados para este N pequeño. También se observa que entre los ojos hay un píxel variable que recibe la influencia de ambos píxeles de condición de borde y que por dicha razón se ve más brillante.



(a) Variables vecinas



(b) Mapeo simple (clamping)

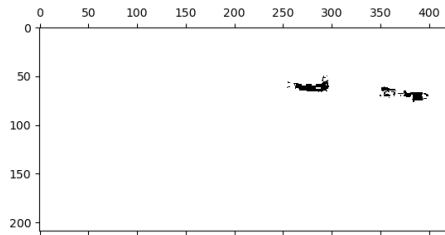


(c) Mapeo de Reinhard

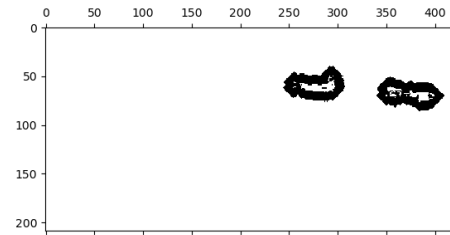
Figura 4: Ejemplo con imagen pequeña, N grande

Para este set de ejemplos se usó la misma imagen pero con $N = 25$ para mostrar un caso donde algunos píxeles relativos de la máscara se salen de la imagen y se ajusta, como se ve en (a). En este caso, el mapeo de Reinhard favorece más el resultado final pues la alta intensidad resultante de los píxeles se distribuye de mejor manera y como consecuencia también, la imagen no se ve tan opaca. En (b) el brillo de los píxeles es tal que incluso contrastan con los colores ligeramente grises de las condiciones de borde.

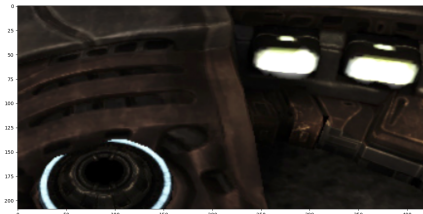
4. Resultados



(a) Condiciones de borde



(b) Variables vecinas



(c) Mapeo simple (clamping)



(d) Mapeo de Reinhard

Figura 5: Ejemplo con imagen más grande

Estos son los resultados de aplicar el efecto bloom a una imagen más compleja, con $N = 10$ y color $[255, 255, 255]$ (blanco). Aquí se pueden apreciar los problemas de tomar un sólo color para seleccionar el área donde aplicar el efecto, ya que uno esperaría que en (a) el conjunto de puntos extraídos tuvieran una forma semejante al de las pantallas brillantes de la imagen, por lo que el resultado es una vecindad bastante irregular y una difuminación que no es consistente con la forma esperada.