Primer Proyecto - Colorizando una colección de fotos

Juan Miguel Gutierrez y Felipe Guzmán

Febrero 2021

1. Introducción

Sergei Mikhailovich Prokudin-Gorskii (1863-1944) fue un hombre que en su momento visualizo las fotos a color. Para esto viajo por el mundo y decidió tomar fotos con placas de filtro rojo, verde y azul. Lamentablemente las fotos fueron imprimidas en una misma cinta, el trabajo actual se encarga recortar las imágenes y ajustarlas de tal manera que al hacer la proyección en RGB, salgan las imágenes lo mas parecidas a la realidad posible.

2. Teoría y Métodos

2.1. Teoría

Función Gamma: Esta función se encarga de realizar un ajuste de contraste de tal forma que mapea los valores de intensidad de entrada a nuevos valores de intensidades en la imagen de salida. Su formula se encuentra determinada por

$$y = ((x - a)/(b - a))^{\gamma} * (d - c) + c$$

donde se convierte de un rango de imágenes de (a, b) a (c, d). Esta función es útil en nuestra implementación pues nos permite ajustar los valores de intensidad, contraste y brillo de la imagen. Aplicamos una transformación no lineal que mejora las correcciones de la imagen.

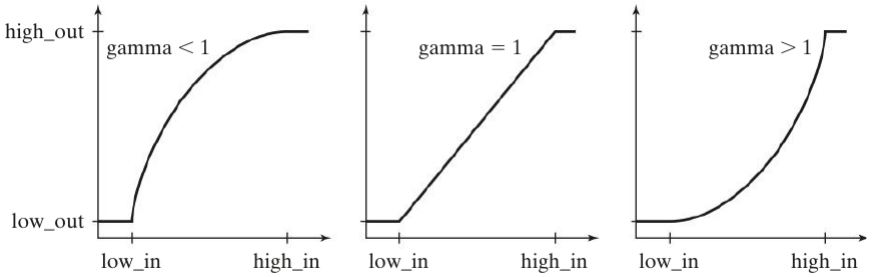

Figura 1: Función Gamma

Métrica L2: Para una imagen existen diferentes normas que nos pueden ayudar a ver que tanto se diferencia dos imágenes, en este proyecto utilizamos la norma L2 entre dos imágenes donde W es el ancho de la imagen y H la altura de la imagen

$$L_2 = \sqrt{\sum_{j=0}^{H-1} \sum_{i=0}^{W-1} (\text{pixel}_{IM1}(j, i) - \text{pixel}_{IM2}(j, i))^2}$$

2.2. Implementación

En esta sección explicaremos de que se encarga cada función creada en el jupyter.

- **metric(IM1,IM2)**: Esta función se encarga de aplicar la métrica L2 sobre las imágenes IM1 y IM2.

- **detectborder_white(IM)**: Se encarga de ir iterativamente por los píxeles de forma diagonal de arriba hacia bajo hasta que deje de encontrar un borde blanco. Es útil para las imágenes que posee bordes blancos.
- **detectborder_black(IM)** Se encarga de ir iterativamente por los píxeles de forma diagonal de arriba hacia bajo hasta desde la posición de frontera de borde blanco que deje de encontrar un borde blanco. Es útil para las imágenes que posee bordes blancos.
- **align_black(IM,IM2,p,h,v)**: Función que utiliza el algoritmo piramidal con altura 'h' para encontrar el mejor ajuste entre la imagen1 y la imagen2, utilizando una vecindad de tamaño 'v' y con un paso de búsqueda 'p'.
- **fixImage(IM,i,h,v)**: Función que se encarga de recortar la imagen en tres partes iguales de tal manera que se pueda obtener cada imagen en una componente del RGB. Luego se procede a encontrar la componente que al compararla con las demás componentes tenga la menor norma, ya que eso mejora la precisión de alineación de imágenes. Luego se procede un filtro pasa altas que ayuda en la comparación de los bordes y se ajusta. Luego se ajustan las capas de colores teniendo en cuenta el color base elegido anteriormente.
- **colorspace(IM,space)**: Función que se encarga mapear la imagen de la componente RGB a otro espacio ya sea grises,hed,hsv,lab,xyz,ycbcr,ybdr,yiq,yuv.

3. Resultados

El algoritmo planteado presenta buenos resultados tanto en las imágenes pequeñas como con las grandes, y por lo tanto es una buena aproximación para resolver el problema planteado. Los resultados por imagen varían en función de los parámetros como la cantidad de vecinos a revisar (s) o la altura utilizada para el algoritmo piramidal (h), sin embargo en muchos casos los resultados no son perfectos.

(a) Imagen Arreglada con el Algorimto (b) Imagen primera aproximación

Figura 2: (a) imagen obtenida tras aplicar el algoritmo con una altura de 3 y revisando 4 vecinos. (b) imagen obtenida al dividir la imagen original por tres y los resultados superponerlos

El error en la aproximaciones lo atribuimos a que la métrica utilizada (euclídea) tiende a buscar la igualdad en las intensidades y no en los detalles de la imagen, por lo que no logra reconocer muy bien la similitud entre los filtros RGB. Debido a esto el algoritmo asigna un valor mayor en la métrica al mejor desplazamiento posible y por lo tanto retornando un desplazamiento que no ajusta a los filtros de la mejor manera.

(a) Imagen Arreglada con el Algorimto

(b) Imagen primera aproximación

Figura 3: (a) imagen obtenida tras aplicar el algoritmo con una altura de 6 y revisando 18 vecinos. (b) imagen obtenida al dividir la imagen original por tres y los resultados superponerlos

Con el fin de mejorar los resultados sin cambiar la métrica se implementaron mejoras en el reconocimiento de los bordes de la imagen, la aplicación de un filtro pasa altas a cada capa RGB y pequeñas rotaciones en las imágenes. Las primeras dos resultaron en una mejora significante en el ajuste de la imagen, sin embargo, la última lo empeora drásticamente lo cual, nuevamente, lo atribuimos a la métrica utilizada.

4. Discusión

Como ya se ha dicho la solución planteada sirve como una aproximación para resolver el problema planteado, aunque no sea perfecto. Para lograr mejores resultados se podría utilizar otra métrica que se centre en los detalles de las imágenes y no en las intensidades de color, aunque puede darse el caso de que estas métricas impliquen una perdida en la eficiencia del programa, debido a la cantidad y complejidad de los cálculos necesarios.

5. Conclusión

- La distancia euclídea, aunque es muy fácil de implementar, solo funciona para lograr una solución parcial al problema.
- El procesamiento de imágenes tiene muchos campos de aplicación y sus metodologías como trabajo teóricos son de mucha utilidad para resolver estos problemas.
- Utilizar diferentes mapeos como HSV fueron útiles para suavizar saturación y iluminación y mejorar la alineación de las imágenes.
- Si bien el algoritmo funciona, existen muchos parámetros que son necesarios ajustar para poder encontrar una alineación justa.