Corrección de color a partir de un patrón de referencia en escenario controlado

Ramiro Andrés Feichubuinm

Trabajo Final de Procesamiento de Imágenes

Motivación

El autor de este trabajo busca encontrar la forma de minimizar la varianza en las adquisiciones por celular de un escenario controlado como el que se muestra a continuación:

En dicha figura se identifican:

- el patrón de referencia (arriba y abajo),
- a sus costados los fiduciarios (marcas para luego encontrar los patrones) y
- los granos de maiz, que en este caso son el objeto a corregir.

La imagen mostrada fue adquirida con un celular. Lógicamente, distintos celulares introducirán distintas perturbaciones. Estas pueden deberse a la lente que usan, el hardware y software que poseen. Se precisa entonces de un algoritmo que pueda resolver este problema de forma dinámica para cada celular.

Introducción

En este trabajo se busca implementar un algoritmo de corrección de color a partir de un patrón de referencia. Para ello se dispone de un escenario controlado en el cual se encuentra el/los objeto/s a ser corregidos y dicho patrón. De este modo, al adquirir una imagen, las distorsiones y/o perturbaciones que se den (por óptica, hardware o software) afectarán a toda la escena. Al poseer tanto el patrón de referencia Original como el Afectado/Modificado, se podrá encontrar una transformación que invierta estos efectos.

En el caso determinístico, donde se tiene la imagen Original no es necesario un patrón, ya que a partir de una inversión matricial se puede encontrar dicha transformación mencionada anteriormente. Pero en general solamente se dispone de la imagen Modificada, solamente pudiendo aproximar dicha transformación y minimizar el error entre ambas imágenes.

El objetivo de este trabajo, es entonces encontrar un algoritmo que desarrolle esta tarea.

Análisis

Es importante entender que este problema, tal como se muestra en la imagen, se encuentra mal condicionado. Esto es debido a que en tanto cada objeto, en este caso cada grano que es introducido en el escenario, sea del mismo tipo (todos granos de maiz, todos granos de soja, etc.), no es necesario utilizar todas las gamas de colores. Aun más, podrían utilizarse más colores de los que se muestran en la imagen, siempre limitándose al área útil designada a esta tarea observada en la imagen.

Esto fue implementado en la notebook *color.ipynb* adjunta, para el caso en cuestión. En dicho trabajo se busca obtener muchos crops (área de interés del grano previamente segmentado) de un mismo tipo, con el fin de estimar los centros tonales. Es decir, estadísticamente hablando, alrededor de qué colores se suelen concentrar los pixeles.

Para ello, se utiliza el algoritmo *k-means*, donde dado un conjunto de observaciones $(x_1, x_2, ..., x_n)$, donde cada observación es un vector real de d dimensiones (en este caso 3), k-medias construye una partición de las observaciones en k conjuntos (k $\leq$ n) a fin de minimizar la suma de los cuadrados dentro de cada grupo (WCSS): S = $\{S_1, S_2, ..., S_k\}$.

$$\underset{S}{argmin} \sum_{i=1}^{k} \sum_{x_j \epsilon S_i} \|x_j - \mu_i\|^2$$

A los colores obtenidos se les debe adicionar 8 (aunque podrían ser más) grises, a modo de poder realizar un balance de blancos.

Posteriormente se analizaron distintas posibilidades para llevar a cabo la tarea ya mencionada:

- Utilizando una LUT en 3 dimensiones de 256x256x256 de manera de disponer de la conversión de todos los colores existentes en RGB. Los problemas son: la cantidad de memoria disponible, ya que el código debe ser ejecutado en la nube, y la imposibilidad de albergar esa cantidad de colores en la aplicación real. Debido a ello el anterior responsable de esta tarea, desarrolló una LUT con un conjunto reducido de colores, obteniendo los restantes por interpolación cúbica. Pero los resultados de este método producían variaciones agresivas en la imagen.

- Utilizando algoritmos complejos. Existen redes neuronales que se ocupan de estas tareas, pero entrenarlas demanda demasiados recursos (búsqueda de arquitectura óptima, digitalización para generar un dataset, etc.).

- Utilizar matrices de corrección de color (CCM). Esta solución es aparentemente intermedia en términos de dificultad, tiempo y resultados y por ello es la que fue aprobada por los superiores del responsable a cargo.

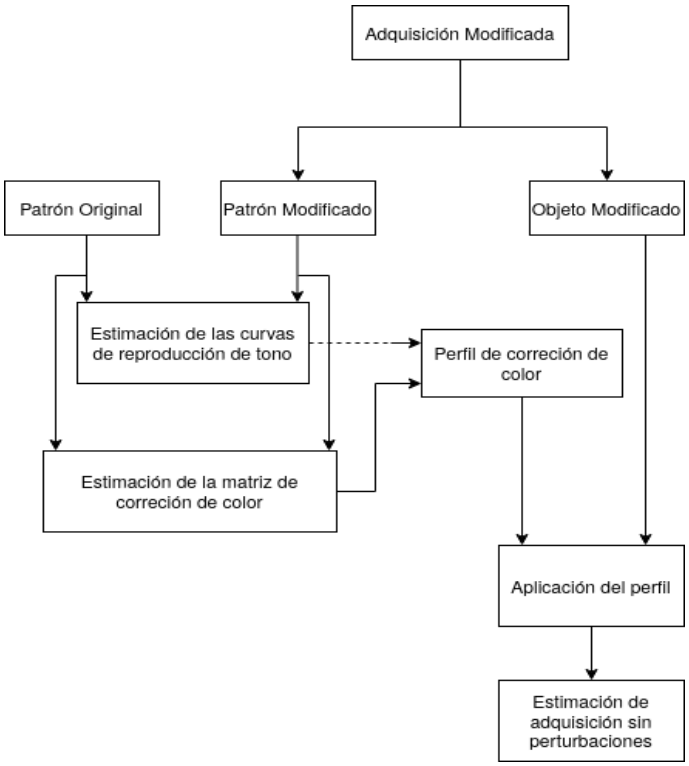
Se dejó para futuras iteraciones la posibilidad de realizar un análisis más exhaustivo de las adquisiciones, modelar el sistema físico de digitalización a modo de mejorar propiedades ópticas del escenario para la obtención de la imagen y cancelar efectos no deseados.

Desarrollo

El algoritmo desarrollado a continuación tiene origen en un trabajo ajeno cuyo propósito es obtener el perfil ICC/ICM de una imagen para asegurar la fiabilidad de reproducción de colores en otros dispositivos ([1]). A grandes rasgos, el mismo consta de tres etapas:

1. Compensación de brillo en el patrón
2. Estimación de las curvas de reproducción de tono
3. Estimación de la matriz de corrección de color

Luego se aplican los pasos 2 y 3 a la imagen de interés. A la hora de la implementación, se encontró que el primer paso resultaba desfavorable, de todas formas se lo mantuvo en la implementación para futuros ajustes. Al desconocer el motivo, se lo dejó de lado.



Estimación de las curvas de reproducción de tono

Este paso es necesario a fin de realizar el balance de blancos anteriormente mencionado. A su vez, se busca corregir errores de forma independiente en cada canal, es decir, asumiendo que las perturbaciones en uno de ellos no afecta a los demás. Para ello, se aplica el siguiente proceso ([2]) a los canales R, G y B por separado:

$$g(x) = a_1 + a_2 x + a_3 sin(x) + a_4 e^x$$

$$0 \leq x \leq 1$$

Como entrada de este proceso se utilizaran los 8 colores grises adicionados de forma manual anteriormente mencionados. Cada canal debe ser normalizado al rango [0, 1], para ello se realiza una división por 255.

Los coeficientes se ajustan siguiendo un criterio LS (Least Squares):

$$C = \sum_{i=1}^{8} [y_1 - \sum_{n=4}^{4} a_n f_n(x)]$$

donde C es la función costo y:

$$f_1(x) = 1 \quad f_2(x) = x \quad f_3(x) = sin(x) \quad f_4(x) = e^x$$

Para minimizar el tiempo de procesamiento inherente del tiempo de procesamiento (el tiempo demandado al *modificar* la imagen de interés con este perfil), se construye una LUT para cada canal.

Si bien los autores de la publicación original no aclaran ningún requerimiento adicional, este proceso no converge a la solución necesaria si no se fuerzan dos puntos (dos colores, pero puntos en la curva) adicionales más:

$$g(0) = 0$$

$$g(1) = 1$$

De este modo se restringe la forma de la curva, evitando que los colores fuente lleven hacia un colores destino no representables en el espacio de colores utilizado (en este caso RGB).

Estimación de la matriz de corrección de color

Este procedimiento es el que intenta corregir errores de forma dependiente, es decir, asume que todos los canales producen una contribución hacia los demás, y los vincula según la siguiente relación:

$$X_{destino} = x_1 X_{fuente} + x_2 Y_{fuente} + x_3 Z_{fuente}$$

$$Y_{destino} = y_1 X_{fuente} + y_2 Y_{fuente} + y_3 Z_{fuente}$$

$$Z_{destino} = z_1 X_{fuente} + z_2 Y_{fuente} + z_3 Z_{fuente}$$

Por lo tanto, previamente se debe realizar una conversión de espacios de colores $RGB \rightarrow XYZ$ ([3]). Esta conversión se realiza con el fin de

Un caso análogo a este es el de una rotación en el espacio de cuerpos sólidos. En este problema se debe sustraer la ubicación del centro de masa del cuerpo, de modo que la transformación sirva para cualquier sólido en ese espacio. Del mismo modo, en el caso en cuestión, se debe sustraer la media.

Los coeficientes x_i, y_i y z_i se obtienen a partir de un criterio LS similar al anterior. Se puede ver fácilmente que:

$$CCM = \begin{pmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ z_1 & z_2 & z_3 \end{pmatrix}$$

donde:

$$IMG_{original} = CCM \,.\, IMG_{modificada}$$

y CCM^{-1}es la matriz de perturbaciones que se obtiene a partir de la estimación de la matriz de corrección de color.

La metodología mencionada hasta ahora funciona, pero con el fin de mejorar los resultados se optó por tratar a la transformación en su forma polinómica de segundo grado:

$$X_{destino} = x_0 + x_{11}X_{fuente} + x_{12}Y_{fuente} + x_{13}Z_{fuente} + x_{21}X^2_{fuente} + x_{22}Y^2_{fuente} + x_{23}Z^2_{fuente}$$

$$Y_{destino} = y_0 + y_{11}X_{fuente} + y_{12}Y_{fuente} + y_{13}Z_{fuente} + y_{21}X^2_{fuente} + y_{22}Y^2_{fuente} + y_{23}Z^2_{fuente}$$

$$Z_{destino} = z_0 + z_{11}X_{fuente} + z_{12}Y_{fuente} + z_{13}Z_{fuente} + z_{11}X^2_{fuente} + z_{12}Y^2_{fuente} + z_{13}Z^2_{fuente}$$

Se dejaron fuera ordenes superiores ya que la relación de compromiso *tiempo de procesamiento-precisión adicional de resultados* no era permisible en esta aplicación. Los términos cruzados XY, XZ e YZ no mostraron mejoría alguna. También se suelen utilizar funciones root square a modo de acoplar el primer grado con el segundo, pero tampoco resultaron útiles.

También se puede representar estas operaciones de forma matricial:

$$IMG_{original} = CCM_1 \,.\, IMG_{modificada} + CCM_2 \,.\, sqr(IMG_{modificada}) + BIAS$$

donde:

$$CCM_1 = \begin{pmatrix} x_{11} & x_{12} & x_{13} \\ y_{11} & y_{12} & y_{13} \\ z_{11} & z_{12} & z_{13} \end{pmatrix} \qquad CCM_2 = \begin{pmatrix} x_{21} & x_{22} & x_{23} \\ y_{21} & y_{22} & y_{23} \\ z_{21} & z_{22} & z_{23} \end{pmatrix}$$

$$BIAS = \begin{pmatrix} x_0 \\ y_0 \\ z_0 \end{pmatrix}$$

y la función *sqr()* es la función cuadrado element-wise.

Los valores iniciales de los coeficientes convergen de forma adecuada cuando se los inicializa en 1. De todas formas queda para futuras pruebas mejores estimaciones a priori de los mismos.

Aplicación del perfil

Luego de obtener el perfil se lo debe aplicar a la imagen que se quiere corregir. Pero antes se debe entender que cualquier dispositivo realiza una compresión gamma al momento de adquirir. Dicho valor de gamma es *device-specific,* de modo que no hay forma de conocer a priori este valor. Por lo tanto, la única forma de aproximarlo es iterar reiteradas veces con distintos valores y elegir un valor que aproxime mejor en el sentido estadístico para la mayoría de las imágenes en cuestión (en este caso, tomadas con diferentes celulares).

Una vez revertida la compresión, se utilizan las LUT obtenidas para cada canal para realizar el ajuste de la etapa 2. Para ello nuevamente se deben normalizar los valores de la imagen, aplicar las curvas, y luego volver al rango [0, 255]. Para evitar que algún valor haya quedado por fuera de este rango, se realiza un clipping. Si bien anteriormente se forzaron valores para que esto no pase, a veces dicho comportamiento no puede ser totalmente asegurado ya que los coeficientes obtenidos buscan minimizar una relación sobre toda la imagen.

Nuevamente, se debe convertir la imagen a espacio de color XYZ y sustraer la media. Ahora si, se está en condiciones de aplicar la transformación.

Finalmente se vuelve a RGB, se adiciona la media y se realiza un clipping. Cabe aclarar que siempre se debe cambiar el tipo de dato entre pasajes de estos dos espacios de color.

Cada una de las micro-operaciones necesarias (clipping, conversión de tipo de datos, etc.) que se pueden encontrar en el código, hacen una contribución al error final obtenido independiente del error de las estimaciones realizadas. De esta forma, se puede expresar la función error *pixel-wise* de la siguiente manera:

$$e(r_e, g_e, b_e) = y(r, g, b) - \hat{y}(r', g', b')$$

Donde $y(r, g, b)$ es la imagen sin perturbaciones, y $\hat{y}(r, g, b)$ es la estimación de la misma, que se obtiene luego de aplicar el perfil a la imagen de interés. Esta resta debe ser *channel-wise:*

$$r_e = r - r'$$

$$g_e = b - b'$$

$$b_e = b - b'$$

Luego se establece una métrica con el fin de obtener una forma de realizar comparaciones cualitativas. Para ello se utiliza el MSE:

$$MSE = \frac{1}{n} \sum_{i=1}^{n} |e(i)|^2$$

donde n es el número total de pixeles en la imagen. Para una imagen, el valor absoluto de un pixel se define en base a la distancia euclideana:

$$MSE = \frac{1}{n} \sum_{i=1}^{n} \sqrt{(r_i - r'_i)^2 + (g_i - g'_i)^2 + (b_i - b'_i)^2}$$

Conclusiones

Como se mencionó durante el trabajo: Se deja para futuras iteraciones la posibilidad de realizar un análisis más exhaustivo de las adquisiciones, modelar el sistema físico de digitalización a modo de mejorar propiedades ópticas del escenario para la obtención de la imagen y cancelar efectos no deseados. También se deja para futuras pruebas mejores estimaciones a priori de los coeficientes iniciales de la CMM.

Dicho esto, se puede concluir (a partir de observar los resultados disponibles en las notebooks que acompañan el trabajo), que dicha resolución maximiza la relación de compromiso recursos utilizados-resultados en el corto plazo para la tarea solicitada.

Fuentes

[1] *Color Correction System Using a Color Compensation Chart for the Images from Digital Camera Seok-Han Lee, Sang-Won Um, and Jong-Soo Choi*

[2] *S. Nakamura, "Applied Numerical Methods in C," Prentice Hall*

[3] Procedural Elements for Computer Graphics by David F. Rogers