

Eliminación del reflejo indeseado de una imagen a partir de la misma.

Álvaro Campillos Delgado y Marcel Espejo Cuenca.

29 de enero de 2021

Resumen

En este trabajo vamos a estudiar y tratar de explicar de forma didáctica el estudio realizado por Yu Li y Michael S. Brown sobre la separación en capas de una imagen, aplicable a la descomposición intrínseca de imágenes y la eliminación del reflejo no deseado.

En este, se resuelve el problema a partir de la optimización de una función objetivo que impone un gradiente suave sobre la capa del reflejo y un gradiente disperso sobre la capa del fondo. Se basa en la observación de que el reflejo está normalmente menos enfocado que la capa del fondo y por tanto tiene valores más pequeños del gradiente.

Palabras clave: reflejo, gradiente, Fourier, transformada, bordes, half-quadratic regularization, transformada rápida de Fourier, procesamiento de imágenes.

1. Introducción.

Cuando se toma una imagen a través de un cristal, ventana o material capaz de reflejar la luz nos encontramos ante un problema para los usuarios y los sistemas de visión por computador puesto que afectan al contenido que proporciona la imagen y a la detección de figuras en la misma, pudiendo ser una pérdida significativa de información.

Algunos ejemplos en los que la eliminación del reflejo puede ser fundamental son, por ejemplo, la detección del uso del cinturón en conductores por la luna del vehículo a causa del reflejo de la luz solar o la de otros vehículos.

Nuestro trabajo se fundamenta en el artículo “Single Image Layer Separation using Relative Smoothness” publicado por Yu Li y Michael S. Brown [7], en el que se propone un método para la descomposición de imágenes a partir de una sola imagen con aplicación en la eliminación del reflejo indeseado de la misma.

Dentro de las posibles técnicas a utilizar, contamos con el uso de múltiples imágenes o solamente una. La eliminación del reflejo en el caso de utilizar múltiples imágenes supone un menor coste y complejidad que en el caso de una sola imagen, aunque es ese último caso el más frecuente.

A continuación, se explican los antecedentes [3] para ambas técnicas con el fin de conocer el contexto en el que se desarrolla el artículo que utilizamos como base en el trabajo.

- **Enfoque utilizando múltiples imágenes:**

Los trabajos de Kong et al. and Schechner et al. utilizan varias imágenes tomadas con diferentes ángulos de polarización para la estimación de la capa reflejada. De forma similar, Farid and Adelson a través del análisis independiente de componentes (separa una señal multivariante en un conjunto de subcomponentes aditivos) para estimar la matriz combinada de dos imágenes tomadas con diferentes ángulos de polarización.

Agrawal et al. parten de dos imágenes de la misma escena tomadas con y sin flash, utilizando la proyección del gradiente (métodos para resolver problemas de optimización de una función objetivo sujeta a restricciones para sus variables) que permite la eliminación de reflejos y destellos en imágenes con flash.

Sirinukulwattana et al. aprovechan el hecho de que los reflejos varían entre distintas imágenes del mismo lugar pero tomadas desde puntos ligeramente diferentes. Imponen una restricción que suavice las áreas del reflejo según su disparidad con las otras capas manteniendo la nitidez del fondo.

Otras soluciones utilizan una secuencia de vídeo para reducir la autocorrelación o correlación cruzada (midiendo la similitud de una señal, siendo esta ella misma u otra respectivamente) del movimiento entre las capas de transmisión y reflejo.

Xue et al. utilizan las diferencias del movimiento para separar la imagen en una capa inicial de transmisión y otra de reflejo. Se extraen los campos de movimiento de las capas (representación ideal de un movimiento en 3D como si estuviese en 2D, de manera que dado cualquier punto en una imagen este es la proyección de un punto en 3D), repitiendo el proceso de actualización de ambas y estimando los campos de movimiento hasta converger.

Guo et al. clasifican el conjunto de restricciones para aprovecharse de la correlación de las capas de transmisión de las diferentes imágenes.

Otros clasifican la puntuación de las capas de movimiento para clasificar los bordes como pertenecientes a la capa de transmisión o a la de reflejo, ayudando a resolver la optimización asociada al problema de separación de las capas.

- **Enfoque utilizando una única imagen:**

Dado que se trata de la situación más habitual, es el enfoque principal adoptado para la resolución del problema. Sin embargo, la formulación del problema en base a esto resulta mucho más compleja.

Muchos de las investigaciones realizadas (incluido el nuestro) se basan a causa de ello, en la dispersión del gradiente para la distinción entre los bordes de la capa de transmisión y la reflejada.

Levin and Weiss resuelven un problema de optimización con restricciones imponiendo sobre los gradientes de imágenes un modelo mixto (modelo probabilístico que representa la presencia de subpoblaciones dentro de una población mayor sin requerir que el conjunto de datos observados identifique a qué observación individual pertenece cada subpoblación) del Laplaciano.

Ha habido mejoras con respecto al artículo tomado como base del proyecto. Algunas de ellas pueden ser Wan et al. calculando la profundidad de campo (zona que comprende desde el punto más cercano y el más lejano de nuestro campo que sea aceptable en cuanto a nitidez una vez formada su imagen en el mismo plano de enfoque) de cada píxel para utilizar la divergencia de Kullback-Leibler (medida no simétrica de la similitud o diferencia entre dos funciones de distribución de probabilidad P y Q) para detectar los bordes en el fondo y reflejo.

Arvanitopoulos et al. [3] que supone la siguiente mejora a partir de nuestro artículo consiguen la eliminación del reflejo indeseado a partir de una sola imagen mediante el uso del Laplaciano y un valor para dispersión del gradiente impuesto en la salida de la imagen final.

2. Planteamiento teórico.

Nuestro artículo trata la extracción de dos capas de una imagen donde se supone que una de ellas es más suave que la otra. Su principal aplicación es la descomposición intrínseca de imágenes (intrinsic image decomposition, proceso de separar una imagen en los elementos que la conforman) con aplicación en la eliminación del reflejo en imágenes. Es un problema inherentemente mal planteado y por ello requiere una serie de restricciones que permitan la obtención de una solución.

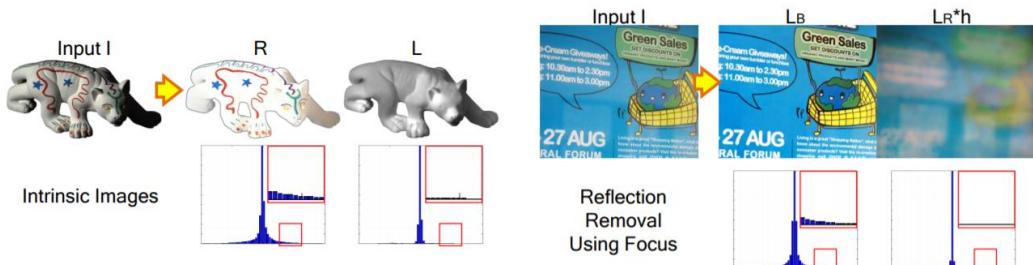


Figure 1. Problemas a los que se aplica el algoritmo: intrinsic image decomposition y eliminación del reflejo utilizando el enfoque. Las distintas capas muestran los histogramas de sus gradientes. En ambos, una de las capas posee menos gradientes altos.

Entendemos por problema mal planteado aquel que no está bien definido, siendo un problema bien definido un problema de Cauchy de valor inicial (problema consistente en resolver una ecuación diferencial con ciertas condiciones de frontera o valores iniciales sobre la solución) con propiedades adecuadas. Algunas de estas condiciones comprenden: la existencia de alguna solución, la unicidad de esta o la solución depende de manera continua de las condiciones iniciales.

Si el problema está bien definido, existe una buena probabilidad de que utilizando un algoritmo estable encontraremos una solución. En caso de no estarlo es cuando necesitamos reformular el problema para que pueda llegar a una solución, proceso que conocemos como regularización.

Un algoritmo estable se trata de aquel algoritmo numérico en el que los errores debidos a las aproximaciones a partir de los datos de entrada se atenúan a medida que la computación procede en lugar de propagarlos a lo largo de la ejecución.

Entendemos por regularizar el proceso de añadir información a un problema mal planteado para resolverlo. Se aplica a las funciones objetivo de los mismos a modo de penalización, imponiendo un coste para alcanzar una solución óptima.

El algoritmo propuesto regulariza el gradiente de las dos capas de manera que una tiene una distribución de cola larga o larga estela y la otra una distribución de cola corta siendo estas la capa correspondiente al fondo y al reflejo respectivamente.

Una distribución de cola larga es aquella que tiene la mayoría de sus elementos en la parte inicial o cabeza mientras que la frecuencia del resto de elementos en la sección restante disminuye gradualmente.

Se trata de un subtipo de la distribución de cola “pesada” (heavy-tailed distribution) que comprende distribuciones de probabilidad cuyas colas son más pesadas que las de la distribución exponencial, hecho que podemos entender de forma sencilla a partir de la siguiente figura.

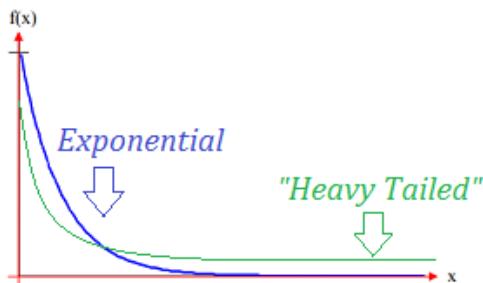


Figure 2. Comparación entre una distribución de cola pesada y una exponencial. Permite visualizar la diferencia entre las mismas.

En nuestro caso (distribución de larga estela), decimos que la distribución de una variable aleatoria X con función de distribución F tiene una cola derecha larga si para todo $t > 0$:

$$\lim_{x \rightarrow \infty} \Pr[X > x + t | X > x] = 1$$

de forma equivalente

$$\bar{F}(x + t) \sim \bar{F}(x) \quad \text{as } x \rightarrow \infty.$$

Todas las distribuciones de larga estela son distribuciones de cola pesada pero su inverso es falso.

Atendiendo a la *Figure 1* y a la *Figure 3* podemos comprobar las distribuciones que se asumen para cada capa como se comentó anteriormente.

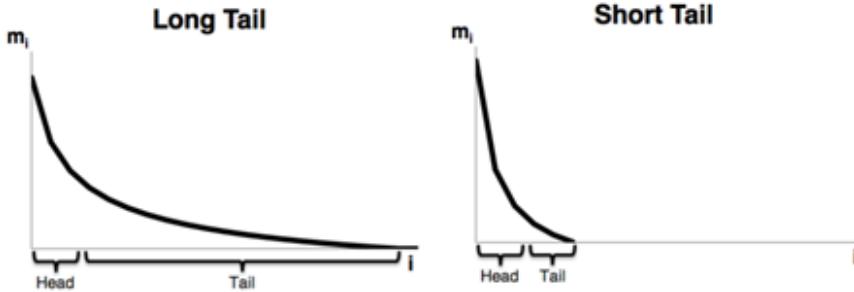


Figure 3. Distribución de larga estela (izq.) y distribución de cola corta (der.).

Hasta el momento de publicación del artículo se había utilizado el enfoque de imponer la distribución de cola larga para los problemas en los que se utilizase la dispersión del gradiente, siendo el uso de una distribución de cola una innovación por parte de los creadores del artículo.

Como hemos comentado anteriormente, este problema se puede formular como la suma de dos capas partiendo de que la imagen es el producto del albedo (porcentaje de radiación que cualquier superficie refleja respecto a la radiación que incide sobre ella) de cada píxel y la iluminación.

En nuestro caso utilizamos una modificación de este planteamiento de Schechner et al.'s, [9] haciendo que una de las capas cuente con un filtro de desenfoque Gaussiano (la capa del reflejo). Lo expresamos como:

$$I = L_B + L_R * h$$

donde la capa del reflejo es convolucionada con el filtro de desenfoque Gaussiano h .

Como hemos visto en clase, el desenfoque Gaussiano se trata del resultado de aplicar una función Gaussiana (*Figure 4*) a una imagen. Reduce la nitidez y en definitiva produce una pérdida de detalles. Dentro de sus ventajas encontramos que es en forma de filtro es separable, permitiendo aplicar dos máscaras unidimensionales (una vertical y otra horizontal).

$$\begin{aligned} G_\sigma(x, y) &= \frac{1}{2\pi\sigma^2} \exp^{-\frac{x^2+y^2}{2\sigma^2}} \\ &= \left(\frac{1}{\sqrt{2\pi}\sigma} \exp^{-\frac{x^2}{2\sigma^2}} \right) \left(\frac{1}{\sqrt{2\pi}\sigma} \exp^{-\frac{y^2}{2\sigma^2}} \right) \end{aligned}$$

Figure 4. Función Gaussiana junto a su separación en función de sus variables. Muestra la separabilidad del filtro correspondiente a esta función.

La separabilidad del filtro reduce el coste computacional de una imagen de $N \times M$ con un filtro de $m \times n$ de $O(MNmn)$ a $O(MN(m+n))$.

Matemáticamente, aplicar un desenfoque Gaussiano a una imagen es equivalente a realizar la convolución de la imagen con una función Gaussiana (conocida como una transformada 2-dimensional de Weierstrass). Se trata de un filtro de paso bajo ya que la transformada de Fourier de un Gaussiano es otro Gaussiano consiguiendo un efecto de reducción de las altas frecuencias en la imagen.

En el caso de la capa que se supone suave (siendo la suave la capa reflejada y emborronada) esta tendrá pocos valores altos del gradiente. Esto significa que necesitamos una restricción adicional en la capa suave.

2.1. Modelo.

Suponemos que la capa L_2 es más suave que L_1 , de forma que es más probable que los valores altos del gradiente pertenezcan a L_1 (siendo el gradiente un indicador de bordes). Modelamos estas probabilidades como:

$$P_1(x) = \frac{1}{z} \max\left\{e^{-\frac{x^2}{\sigma_1^2}}, \epsilon\right\},$$

$$P_2(x) = \frac{1}{2\pi\sigma_2^2} e^{-\frac{x^2}{\sigma_2^2}},$$

Ecuación 1. Modelo probabilístico del problema. $P_1(x)$ representa la probabilidad de que un gradiente pertenezca a la capa del fondo LB y $P_2(x)$ que pertenezca a la del reflejo.

dónde x es el valor del gradiente, z es un factor de normalización, σ_1 y σ_2 son dos pequeños valores que generan dos distribuciones estrechas Gaussianas (ver Figure 5) que decrecen muy rápidamente. Utilizamos el operador *max* con ϵ en P_1 para añadir una cola que previene que la función de probabilidad se acerque a 0.

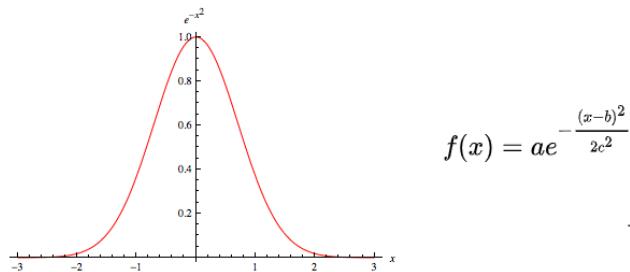


Figure 5. Distribución gaussiana en \mathbb{R}^2 (para ciertos valores de los parámetros de $f(x)$). Gráfica (izq.) y función correspondiente (der.).

El modelado mediante estas probabilidades tiene el objetivo de buscar el reparto más sensato entre las dos capas que componen la imagen y así resolver el problema planteado.

En esencia estamos maximizando la probabilidad conjunta de ambas capas. Lo conseguimos mediante la minimización del negativo del logaritmo de las probabilidades.

$$\begin{aligned}-\log P_1(x) &\propto \min\left\{\frac{x^2}{\sigma_1^2(-\log \epsilon)}, 1\right\} + C_1, \\ -\log P_2(x) &\propto \frac{x^2}{\sigma_2^2} + C_2.\end{aligned}$$

Ecuación 2. Probabilidades del modelo en forma logarítmica. El símbolo \propto denota proporcionalidad.

donde C_1 y C_2 son dos constantes que desaparecerán posteriormente.

- **¿Qué es una log-probabilidad y por qué se utiliza?**

Se trata simplemente de tomar el logaritmo de una probabilidad. Su uso consigue la representación de las probabilidades en la escala logarítmica en lugar del intervalo estándar $[0,1]$. Su utilidad principal es la creación de algoritmos estables numéricamente y la simplificación de las operaciones en estos.

Dado un conjunto de eventos independientes cuyas probabilidades son multiplicadas, mediante el uso del logaritmo convertimos dichas operaciones a sumatorios. Esto consigue una mayor eficiencia computacional, siendo un sumatorio más fácilmente optimizable puesto que la derivada de una suma es la suma de sus derivadas (regla del producto frente a la suma de derivadas).

$$\log(\prod_i P(x_i)) = \sum_i \log(P(x_i))$$

Figure 6. Equivalencia entre un producto y un sumatorio mediante el uso de logaritmos.

Teniendo sumatorios en lugar de productos también ahorramos espacio puesto que podemos realizar los cálculos de los sumatorios de forma independiente, acumularlos y finalmente aplicar el paso correspondiente a la optimización.

De igual forma, es especialmente útil cuando la función que modela la probabilidad asociada pertenece a la familia de las funciones exponenciales por las simplificaciones que consigue.

Todo ello es posible gracias a que produce una transformación monótona que preserva el orden de los elementos de la transformada.

Por convención utilizamos la minimización como forma de optimización en lugar de la maximización. El hacer la función a optimizar negativa realmente estamos maximizando.

Podemos observar en la fórmula (ver **Error! Reference source not found.**) que $\log P_2(x)$ está en la forma correspondiente a L_2 pero $-\log P_1(x)$ está en forma truncada que posteriormente simplificamos como $p(x)=\min\{x^2 / k, 1\}$.

Entendemos por forma truncada a que tiene una distribución truncada. Una distribución truncada es una distribución condicional que surge de una restricción en el dominio de otra distribución de probabilidad. En nuestro caso, esto se debe al operador *max/min*. La k es un número pequeño 10^{-4} en nuestro método.

Se asume que las probabilidades para las dos capas son independientes (i.e. $P(L_1, L_2) = P(L_1) \cdot P(L_2)$) y que la salida al aplicar filtros son independientes (i.e. $\prod_i P_t(f_j * L)_i, t \in \{1,2\}$), minimizando $-\log P(L_1, L_2)$ obtenemos la función objetivo:

$$\min_{L_1, L_2} \sum_{i,j} \left(\rho(L_1 * f_j)_i + \lambda(L_2 * f_j)_i^2 \right),$$

Ecuación 3. Función objetivo obtenida a partir de la Eq. 2.

donde i es el índice del píxel, f_j denota los distintos filtros. Utilizamos dos filtros direccionales de primer orden y un filtro Laplaciano (segundo orden).

$$f_1 = [-1 \ 1], \quad f_2 = [-1 \ 1]^T, \quad f_3 = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Figure 7. Filtros utilizados para la detección de bordes. f_1 y f_2 son de primer orden mientras que f_3 es de segundo (Laplaciano).

Los filtros direccionales son detectores de bordes que pueden utilizar para calcular las primeras derivadas. Pueden ser diseñados para cualquier dirección dentro de un espacio dado.

Con f_1 detectamos los bordes en el eje x mientras que con f_2 detectamos los bordes en el eje y calculando las derivadas parciales en dichas direcciones para la imagen a la que le realizamos la convolución.

El Laplaciano es un operador diferencial calculado a partir de la suma de las segundas derivadas parciales de una función con respecto a cada variable independiente. Es por ello que lo utilizamos para la detección de bordes en el procesamiento de imágenes digitales.

$$L(x, y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

Figure 9. Operador Laplaciano.

0	-1	0
-1	4	-1
0	-1	0

-1	-1	-1
-1	8	-1
-1	-1	-1

Figure 8. Filtros comúnmente utilizados para la aproximación de la derivada de segundo orden.

Ya que aproximan la segunda derivada de la imagen, son muy sensibles al ruido. Para contrarrestarlo se suele hacer un suavizado Gaussiano antes de aplicar el filtro Laplaciano. Sin embargo, como la operación de convolución es asociativa podemos primeramente hacer la convolución de ambos filtros y posteriormente aplicarlo a la imagen.

Conseguimos así un menor número de operaciones puesto que la convolución entre ambos filtros supone menos cálculos a causa de su tamaño y que el filtro combinado de estos (Gaussiano del Laplaciano) se puede calcular de forma previa para únicamente requerir una convolución en tiempo de ejecución sobre la imagen a filtrar.

$$LoG(x, y) = -\frac{1}{\pi\sigma^4} \left[1 - \frac{x^2 + y^2}{2\sigma^2} \right] e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Figure 10. Filtro combinado.

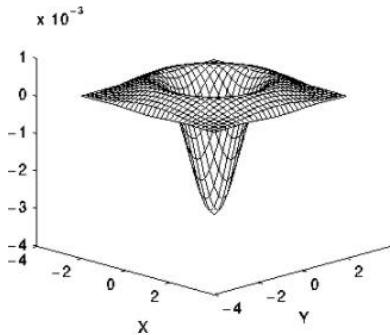


Figure 12. Gráfica de la función que representa el filtro combinado. Los ejes x e y representan las desviaciones estándar.

v	1	1	2	2	2	1	1	0
1	2	4	5	5	5	4	2	1
1	4	5	3	0	3	5	4	1
2	5	3	-12	-24	-12	3	5	2
2	5	0	-24	-40	-24	0	5	2
2	5	3	-12	-24	-12	3	5	2
1	4	5	3	0	3	5	4	1
1	2	4	5	5	5	4	2	1
0	1	1	2	2	2	1	1	0

Figure 11. Ejemplo del filtro híbrido para $\sigma=1.4$.

En el artículo se comenta que de forma experimental los filtros derivativos de primer orden en L_1 y el Laplaciano en L_2 han resultado en buenos resultados. Aunque se asume que las imágenes no tienen ruido puede ser de útil para la fase de experimentación.

Los filtros derivativos de primer orden consiguen obtener los bordes significativos de L_1 y el Laplaciano refleja las variaciones de suavidad en L_2 .

Nuestras probabilidades están definidas en términos del gradiente y para recuperar capas significativas debemos acotar los posibles valores de la solución. Los rangos son establecidos en función de la aplicación del algoritmo.

$$\begin{aligned} \min_{L_1} \sum_i & \left(\sum_{j=1,2} \rho(F_i^j L_1) + \lambda (F_i^3 L_1 - F_i^3 I)^2 \right) \\ \text{s.t. } & lb_i \leq (L_1)_i \leq ub_i. \end{aligned}$$

Ecuación 4. Función objetivo general para L_1 .

La acotación de los valores de la capa con la que trabajamos se debe a que, en caso contrario, las convoluciones y sumatorios que realizamos posteriormente en el algoritmo pueden hacer que la imagen tome tonos demasiado oscuros o claros produciendo una pérdida del contenido de la misma que no es aceptable.

Las cotas de dicho rango tomarán forma de matrices cuyos valores variarán en función de la aplicación del algoritmo. En el caso de la eliminación del reflejo, el rango inferior \mathbf{l}_b será una matriz nula y el rango superior \mathbf{u}_b será la propia imagen, ambas con las dimensiones de esta. Con esta formulación, sea \mathbf{L}_l igual a \mathbf{L}_B (capa del fondo):

$$\begin{aligned} \min_{\mathbf{L}_B} & \sum_i \left(\sum_{j=1,2} \rho(F_i^j \mathbf{L}_B) + \lambda(F_i^3 \mathbf{L}_B - F_i^3 I)^2 \right) \\ \text{s.t. } & 0 \leq (L_B)_i \leq I_i. \end{aligned}$$

Ecuación 5. Función objetivo para el caso de la eliminación del reflejo.

El parámetro λ de la función objetivo controla la suavidad que existe en la capa \mathbf{L}_2 , parámetro de gran importancia en nuestra implementación puesto que debemos encontrar su valor mediante prueba y error para distintas fotografías según la importancia del reflejo existente en ellas. A fin de cuentas, representa la intensidad que la capa del reflejo \mathbf{L}_2 da a la imagen (una mayor magnitud indicará una menor intensidad del reflejo).

2.2.Optimización.

La optimización de la función objetivo anterior requiere de cierto tratamiento con el fin de poderse llevar a cabo y encontrar una solución. Se utiliza un enfoque en dos pasos.

Primeramente, la función es no convexa a causa de su componente $p(x)$ el cual se introdujo anteriormente. Para resolver esta complicación, se sigue el esquema de regularización semicuadrático [1][8] en el cual se introducen una serie de variables auxiliares junto a una versión de la función objetivo extendida en la que las variables originales son cuadráticas y las variables auxiliares están desacopladas.

La minimización de estas variables auxiliares por sí mismas es equivalente a la función objetivo no extendida u original y, por tanto, la estimación de la imagen final (que en el caso de los artículos referenciados es la recuperación ante el ruido) puede conseguirse mediante minimización conjunta. Lo que nos permite este esquema es resolver un problema de optimización no-convexo sin tener la restricción de desigualdad.

En segundo lugar, debemos devolver los valores de la capa al rango como se indicó anteriormente.

Por el método de esquema semicuadrático se introducen variables auxiliares g_i^j en cada píxel para permitir sacar $F_i^j \mathbf{L}_l$ fuera de la función $p(\cdot)$.

$$\min_{\mathbf{L}_l, g_i^j} \sum_i \left(\sum_{j=1,2} (\beta(F_i^j \mathbf{L}_l - g_i^j)^2 + \rho(g_i^j)) + \lambda(F_i^3 \mathbf{L}_l - F_i^3 I)^2 \right),$$

Ecuación 6. Función objetivo tras aplicar el esquema de regularización semicuadrático.

donde β es un peso que incrementamos durante la optimización. Conforme β crece la solución se acerca a la óptima. Los distintos efectos que tiene en el algoritmo se desarrollan en sus respectivas secciones.

La minimización de esta fórmula para una β fija se puede conseguir alternando entre el cálculo de L_1 y la actualización de g^j .

2.2.1. Actualización de g^j :

Manteniendo L_1 fijo, la solución en cada píxel de la imagen minimiza la ecuación anterior con g^j como:

$$g_i^j = \begin{cases} F_i^j L_1, & (F_i^j L_1)^2 > \frac{1}{\beta} \\ 0, & \text{otherwise.} \end{cases}$$

Esta operación de umbralización se sostiene cuando $\beta < 1/k$.

Aquí β se utiliza como valor para la umbralización que determinará que valores pertenecen a L_1 (fondo) y cuáles a L_2 (reflejo).

La fórmula tiene una convolución en el dominio del espacio entre el filtro j en el pixel i con $L_1 (F_i^j L_1)$ mediante la cual se detectan los bordes en el eje x e y (ya que los filtros que se le aplican son el 1 y el 2).

Se comprueba si supera el umbral determinado por β , de manera que si lo hace se considera que los bordes detectados por la convolución han de mantenerse en la capa del fondo sobre la que estamos trabajando y en caso contrario hacemos que se haga cero (indicando la pertenencia al reflejo al no pertenecer al fondo).

Su aumento a lo largo de las iteraciones supone la disminución del valor para la umbralización y por tanto más valores pasarán a ser parte de la capa del fondo. Aunque de primeras parezca no tener sentido, ya que puede entenderse que la imagen del fondo tendrá más información de la que debería con partes de reflejo no eliminado, en las distintas iteraciones se trabajará con L_1 actualizado en esta fase, en la del cálculo de la capa en si (2.2.2) y en la normalización (2.2.3) de manera que los valores que pasan a formar parte de L_1 irán convergiendo a los óptimos.

2.2.2. Cálculo de L_1 :

Primeramente, se desarrollan una serie de conceptos teóricos relacionados con el procesamiento de señales y algoritmos dentro de este campo.

- **Transformada de Fourier:**

La transformada de Fourier es una transformada integral que transforma señales entre el dominio espacial (del tiempo, no limitado a este si no que nos referimos así al dominio de la función original) y el dominio de la frecuencia. La transformada de Fourier de una función respecto al tiempo es una función de variable compleja de la frecuencia, cuya magnitud representa la cantidad de esa frecuencia que está presente en la función original y cuyo argumento (ángulo comprendido entre el eje real positivo del plano complejo y el vector que une el punto con este plano) será la fase del senoide en dicha frecuencia.

Las operaciones en un dominio tienen su equivalente en el otro, las cuales suelen ser más fáciles de realizar y por ello hace conveniente la transformación. Tras realizar las operaciones deseadas volvemos al dominio espacial.

Dominio espacial	Dominio de la frecuencia
Diferenciación	Multiplicación
Convolución	Multiplicación
Deconvolución	División

Todo ello bajo el Teorema de convolución, el cual establece qué bajo determinadas circunstancias, la transformada de Fourier de una convolución es el producto punto a punto de las transformadas.

$$\mathcal{F}\{f * g\} = \mathcal{F}\{f\} \cdot \mathcal{F}\{g\}$$

$$\mathcal{F}\{f \cdot g\} = \mathcal{F}\{f\} * \mathcal{F}\{g\}$$

Aplicando la inversa de la transformada

$$f * g = \mathcal{F}^{-1}\{\mathcal{F}\{f\} \cdot \mathcal{F}\{g\}\}$$

$$f \cdot g = \mathcal{F}^{-1}\{\mathcal{F}\{f\} * \mathcal{F}\{g\}\}$$

*Nota: En este contexto el asterisco * denota la convolución y no la multiplicación.*

Algunas de las propiedades básicas de la transformada de Fourier (siendo f absolutamente integrable) son las siguientes, de las cuales faremos uso posteriormente:

- Linearidad: $\mathcal{F}\{a \cdot f + b \cdot g\} = a \mathcal{F}\{f\} + b \mathcal{F}\{g\}.$
- Traslación: $\mathcal{F}\{f(t - a)\}(\xi) = e^{-\pi i \xi a} \cdot \mathcal{F}\{f\}(\xi)$
- Conjugado:
If $h(x) = \overline{f(x)}$, then
 $\hat{h}(\xi) = \overline{\hat{f}(-\xi)}.$

La equivalencia entre la transformada y su inversa, siendo $f: \mathbb{R} \rightarrow \mathbb{C}$ y sea la transformada de Fourier de la forma:

$$(\mathcal{F}f)(\xi) := \int_{\mathbb{R}} e^{-2\pi i y \cdot \xi} f(y) dy,$$

entonces la inversa se define como

$$f(x) = \int_{\mathbb{R}} e^{2\pi i x \cdot \xi} (\mathcal{F}f)(\xi) d\xi.$$

$$\tilde{f}(\xi), \tilde{f}(\omega), F(\xi), \mathcal{F}(f)(\xi), (\mathcal{F}f)(\xi), \mathcal{F}(f), \mathcal{F}(\omega), F(\omega), \mathcal{F}(j\omega), \mathcal{F}\{f\}, \mathcal{F}(f(t)), \mathcal{F}\{f(t)\}.$$

I. Distintas notaciones para la transformada de Fourier.

- **Transformada discreta de Fourier (DFT):**

Realiza la transformación de una función del dominio espacial al dominio de la frecuencia, pero requiere que la función de entrada sea una secuencia discreta y de duración finita.

Únicamente evalúa suficientes componentes de la frecuencia para reconstruir el segmento finito que analiza, de forma que implica que el segmento analizado es un único periodo de una señal periódica que se extiende de forma infinita; si no se cumple debemos utilizar una sección de la señal. Esto afecta de igual forma a la IDFT.

Formalmente, definimos la DFT para una secuencia N de números complejos x_0, \dots, x_{N-1} se transforma en la secuencia de N números complejos X_0, \dots, X_{N-1} mediante la DFT con la fórmula:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} kn} \quad k = 0, \dots, N-1$$

donde i es la unidad imaginaria y $e^{\frac{2\pi i}{N}}$ es la N-ésima raíz de la unidad.

La transformada inversa de Fourier (IDFT) la definimos como:

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k e^{\frac{2\pi i}{N} kn} \quad n = 0, \dots, N-1.$$

Podemos entender para estas ecuaciones que los números complejos X_k representan la amplitud y la fase de diferentes componentes sinusoidales de la señal de entrada x_n .

Una de sus propiedades principales que interviene en el cálculo de L₁ es el teorema de Plancherel donde para las DFTs X_k e Y_k de x_n y y_n se establece que

$$\sum_{n=0}^{N-1} x_n y_n^* = \frac{1}{N} \sum_{k=0}^{N-1} X_k Y_k^*$$

donde el asterisco denota la conjugación compleja.

- **Transformada rápida de Fourier (FFT):**

Se trata de un algoritmo que calcula la transformada de Fourier discreta (DFT) o su inversa (IDFT). El cálculo de la DFT a partir de su definición resulta muy costoso computacionalmente pasando de $O(N^2)$ a $O(N \log N)$ con la FFT. Existen múltiples implementaciones del algoritmo.

El algoritmo más utilizado es el algoritmo de Cooley-Turkey, algoritmo que sigue el paradigma de divide y vencerás que reformula la DFT de tamaño arbitrario igual a un número compuesto (cualquier número natural no primo) de la forma $N=N_1N_2$ con N_1 como DFTs menores de tamaño N_2 para reducir el tiempo de ejecución para números lisos (enteros que pueden factorizarse completamente en números primos pequeños).

Para ello, dividimos la transformada de Fourier en subsecuencias indexadas por su paridad según la siguiente función:

$$\begin{cases} n = 2r & \text{if even} \\ n = 2r + 1 & \text{if odd} \end{cases}$$

where $r = 1, 2, \dots, \frac{N}{2} - 1$

Con ciertas transformaciones a la transformada podemos conseguir convertirla en la suma de dos términos que nos servirán para mejorar la eficiencia del algoritmo.

$$x[k] = \sum_{n=0}^{N-1} x[n] e^{\frac{-j2\pi kn}{N}}$$

$$x[k] = \sum_{r=0}^{\frac{N}{2}-1} x[2r] e^{\frac{-j2\pi k(2r)}{N}} + x[k] = \sum_{r=0}^{\frac{N}{2}-1} x[2r+1] e^{\frac{-j2\pi k(2r+1)}{N}}$$

$$x[k] = x_{even}[k] + e^{\frac{-j2\pi k}{N}} x_{odd}[k]$$

$$x[k] = \sum_{r=0}^{\frac{N}{2}-1} x[2r] e^{\frac{-j2\pi k(2r)}{N}} + x[k] = e^{\frac{-j2\pi k}{N}} \sum_{r=0}^{\frac{N}{2}-1} x[2r+1] e^{\frac{-j2\pi k(2r)}{N}}$$

$$x[k] = \sum_{r=0}^{\frac{N}{2}-1} x[2r] e^{\frac{-j2\pi k(r)}{N/2}} + x[k] = e^{\frac{-j2\pi k}{N}} \sum_{r=0}^{\frac{N}{2}-1} x[2r+1] e^{\frac{-j2\pi k(r)}{N/2}}$$

Figure 13. División de la DFT en término par e impar.

La ventaja de esto es que permite calcular las subsecuencias de forma concurrente (ver *Figure 14*).

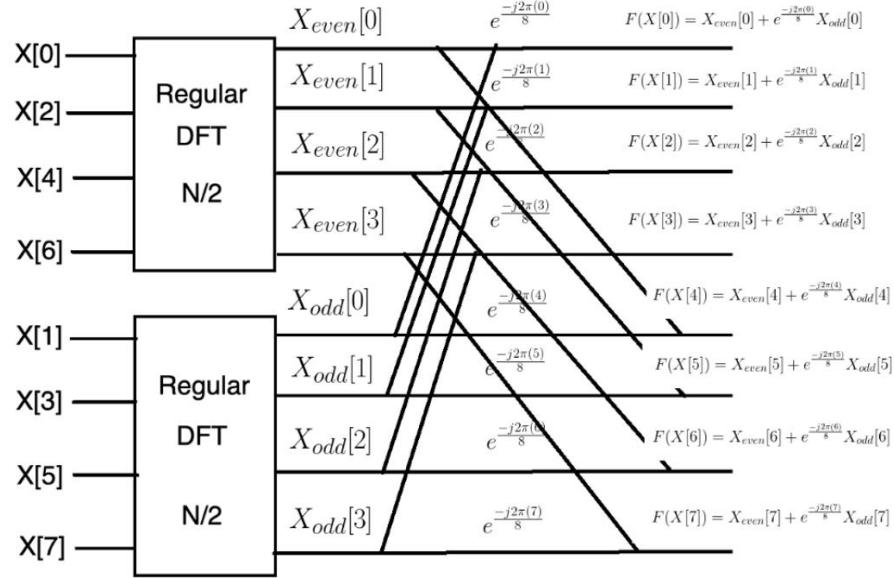


Figure 14. Ejemplo de la ejecución de FFT (sin dividir y vencerás) para N=8.

$$\begin{aligned}
 & \text{2 DFT} \quad \downarrow \quad \text{DFT on } N/2 \text{ elements} \\
 & 2 \times \left(\frac{N}{2}\right)^2 + N \quad x[k] = x_{\text{even}}[k] + e^{-j\frac{2\pi k}{N}} x_{\text{odd}}[k]
 \end{aligned}$$

$$\begin{aligned}
 & = 2 \times \frac{N^2}{4} + N \\
 & = \frac{N^2}{2} + N \\
 & O\left(\frac{N^2}{2} + N\right) \sim O(N^2)
 \end{aligned}$$

$$\begin{aligned}
 \frac{N}{2} & \rightarrow 2 \left(\frac{N}{2}\right)^2 + N = \frac{N^2}{2} + N \\
 \frac{N}{4} & \rightarrow 2(2 \left(\frac{N}{4}\right)^2 + \frac{N}{2}) + N = \frac{N^2}{4} + 2N \\
 \frac{N}{8} & \rightarrow 2(2(2 \left(\frac{N}{8}\right)^2 + \frac{N}{4}) + \frac{N}{2}) + N = \frac{N^2}{8} + 3N \\
 & \quad \cdot \quad \cdot \quad \cdot \\
 \frac{N}{2^P} & \rightarrow \frac{N^2}{2^P} + P N = \frac{N^2}{N} + (\log_2 N) N = N + (\log_2 N) N \\
 & \sim O(N + N \log_2 N) \sim O(N \log_2 N)
 \end{aligned}$$

Figure 15. Comparación de la complejidad del algoritmo hasta el momento y utilizando divide y vencerás. El algoritmo sin usar la división (izq.) supone un coste bastante mayor que con la división (der.).

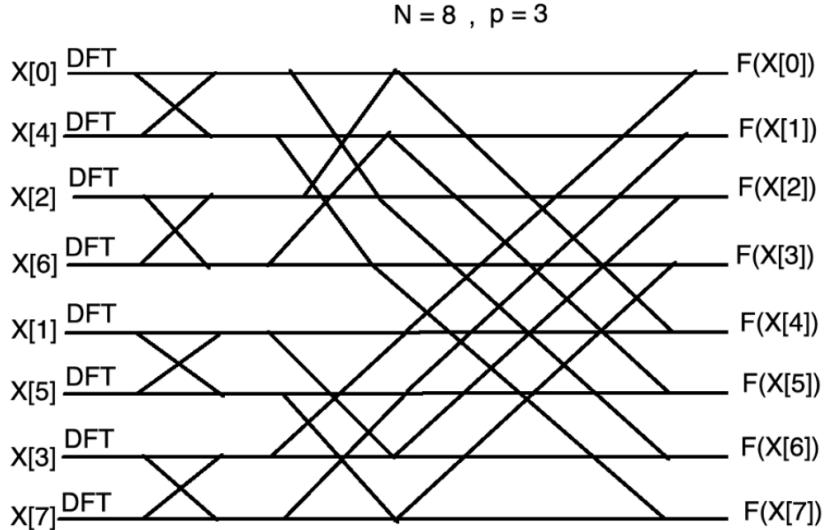


Figure 16. Ejemplo para la ejecución de FFT con divide y vencerás ($N=8$).

Ahora podemos pasar a entender la fórmula utilizada para la actualización o cálculo de la capa L_1 .

Si mantenemos g^j fijo, L_1 es cuadrática. Asumiendo condiciones de frontera periódicas, aplicamos el algoritmo de FFT que diagonaliza las matrices de convolución F^j para encontrar el L_1 óptimo.

$$L_1 = \mathcal{F}^{-1}(A),$$

$$A = \frac{\beta \sum_j (\mathcal{F}(F^j)^* \mathcal{F}(g^j)) + \lambda \mathcal{F}(F^3)^* \mathcal{F}(F^3) \mathcal{F}(I)}{\beta \sum_j (\mathcal{F}(F^j)^* \mathcal{F}(F^j)) + \lambda \mathcal{F}(F^3)^* \mathcal{F}(F^3) + \tau}$$

Ecuación 7. Fórmula para el cálculo de L_1 . Se realiza el procesamiento correspondiente en el dominio de la frecuencia para A y finalmente se obtiene L_1 .

dónde \star simboliza el complejo conjugado, el parámetro τ se trata de una cantidad necesaria para aumentar la estabilidad del algoritmo propuesto (prevenir que el denominador sea 0).

La resolución de esta ecuación requiere el cómputo de solo dos FFT para g^1 y g^2 y una IFFT en cada iteración ya que los otros términos se pueden calcular previamente.

La idea del cálculo de L_1 es utilizar las variables auxiliares anteriores para detectar bordes que se irán actualizando por cada iteración con el valor de β , junto a la detección de bordes del Laplaciano sobre la imagen original (ambos a través de sus respectivas convoluciones en el dominio de la frecuencia a través de la transformada de Fourier) de forma que a través de esta última parte ($\lambda \mathcal{F}(F^3)^* \mathcal{F}(F^3) \mathcal{F}(I)$) recuperaremos la información de la imagen (no los bordes que es con lo que estamos operando) mediante la deconvolución que realizamos a través de la división de los dos elementos del numerador con los elementos del denominador.

Finalmente calculamos su IFFT para devolver la imagen al dominio del espacio.

La función del denominador a fin de cuentas es la deconvolución de la matriz que está en el numerador eliminando el efecto que ha realizado la convolución que se ha realizado con estos mismos filtros, dejando en la matriz resultante los valores actualizados.

A medida que avanzan las iteraciones, la fase de actualización y el cálculo del nuevo L_1 interactúan entre sí de manera que con la actualización de g^j se elimina parte del reflejo, pero se recuperan partes de la imagen original (mediante $\lambda\mathcal{F}(F^3)^*\mathcal{F}(F^3)\mathcal{F}(I)$) para compensar que se hayan podido eliminar partes del fondo en la actualización (umbralización). A medida que el algoritmo avanza, β aumenta haciendo que el valor del umbral disminuya y se atribuyan más bordes (detectados en la convolución) al fondo y por tanto se recuperen menos partes de la imagen original (debido a que β gana más peso).

Este último hecho hace que se necesite una normalización de L_1 ya que aumentar el peso de una de las partes que interviene en la definición de L_1 oscurecerá la imagen (cosa que también se trata de reducir al formar parte β del denominador, aunque el sumatorio por el que se multiplica no tenga la misma magnitud que su correspondiente en el numerador).

Con las iteraciones aumenta el peso de β , de forma que lo que consigue nuestra fórmula para el cálculo de L_1 cambiará:

- **En el numerador:** Tomará más importancia el sumatorio de las convoluciones de los filtros f_1 y f_2 (derivada en x e y) con las variables auxiliares g^j que en iteraciones anteriores.

En función de la magnitud de β puede tomar más importancia que la convolución de f_3 (Laplaciano) con la imagen por parámetro de suavidad relativa λ .

- **En el denominador:** Tomará más importancia el sumatorio de las convoluciones de los filtros f_1 y f_2 (derivada en x e y) por si mismos que en iteraciones anteriores.

En función de la magnitud de β puede tomar más importancia que la convolución de f_3 (Laplaciano) con la imagen por parámetro de suavidad relativa λ .

2.2.3. Normalización de L_1 :

Este paso es de vital importancia puesto que la solución a (6) no es única y está relacionada por una constante global. Para buscar una t adecuada tratamos de minimizar la siguiente función objetivo:

$$\min_t \sum_i m_i((L_1)_i + t - lb_i)^2 + \sum_i n_i((L_1)_i + t - ub_i)^2$$

Ecuación 8. Función objetivo que se utiliza para la normalización.

donde m_i y n_i son funciones que indican la salida del rango para los valores de la imagen. m_i es 1 si $(L_1)_i + t < lb_i$ y n_i es 1 si $(L_1)_i + t > ub_i$. Produce la actualización de L_1 a $L_1 + t$. Podemos utilizar un método de optimización como es una versión simple del gradiente descendente.

- **Gradiente descendente o descenso por el gradiente:**

El gradiente descendente es un método de optimización iterativo de primer orden (métodos que tienen como máximo un error cometido por un paso del método) para buscar el mínimo local de una función diferenciable.

La idea general es iterar tomando la dirección opuesta al gradiente en el punto actual de manera que sea la dirección del descenso con mayor inclinación. El método opuesto es el gradiente ascendente y consiste en tomar la dirección del gradiente.

Sea la función a optimizar una función multivariante, $F(\mathbf{x})$ definida y diferenciable en el entorno de un punto a , entonces $F(\mathbf{x})$ decrece más rápidamente si uno va desde a en la dirección del negativo del gradiente de F en a , $-\nabla F(a)$. Se sigue que si

$$\mathbf{a}_{n+1} = \mathbf{a}_n - \gamma \nabla F(\mathbf{a}_n)$$

para un $\gamma \in \mathbb{R}_+$ lo suficientemente pequeño entonces $F(\mathbf{a}_n) \geq F(\mathbf{a}_{n+1})$. En otras palabras, se sustrae $\gamma \nabla F(a)$ a a para desplazarnos en la dirección contraria al gradiente hacia un mínimo local.

En nuestro caso, la función $F(\mathbf{x})$ utilizada para el gradiente descendente es la función objetivo compuesta por las funciones indicativas m_i y n_i .

3. Implementación.

A continuación, se presenta el algoritmo propuesto por los creadores del artículo, cuyos pasos se han desarrollado a nivel teórico previamente.

Algorithm 1 Layer Separation using Relative Smoothness

Input: input image I ; smoothness weight λ ; initial β_0 ; iterations number i_{\max} ; increasing rate η ;

Initialization: $L_1 \leftarrow I$; $\beta \leftarrow \beta_0$; $i \leftarrow 0$.

while $i < i_{\max}$ **do**

update g_i^j using Eqn. 7;

compute L_1 using Eqn. 8;

normalize L_1 using Eqn. 9;

$\beta = \eta * \beta, i++$;

end while

$L_2 = I - L_1$;

Output: The estimation of two layers L_1 and L_2 ;

3.1.Funciones auxiliares.

Necesitaremos una serie de funciones auxiliares que nos permitirán trabajar con la imagen en un rango adecuado para nuestro algoritmo y aumentar su eficiencia haciendo cálculos previos.

Las funciones que utilizaremos serán:

- **im2doubleAux(imagen):** Realiza la normalización de la imagen de [0..255] a [0. .. 1.]. La normalización se realiza según la fórmula:

$$z_i = \frac{x_i - \min(x)}{\max x - \min(x)}$$

dónde $x = (x_1, \dots, x_n)$ y z_i es el dato i-ésimo normalizado.

Esto se debe a requerimientos de la propia implementación propuesta por los creadores del artículo.

- **psf2otf(psf, shape):** Realiza la transformación de las PSF a las OTF correspondientes a través de la transformada rápida de Fourier (FFT). Fuente: <https://github.com/aboucaud/pypher/blob/master/pypher/pypher.py>

La última función nos permite calcular las funciones de los distintos filtros que convolucionamos con la imagen en el dominio espacial a su correspondiente en el dominio de la frecuencia puesto que todas las operaciones se realizan en este.

En el dominio espacial estos filtros representan funciones de dispersión de puntos (point-spread function, PSF) cuyo equivalente es una función de transferencia óptica (optical-transfer function, OTF). Las OTF no estará influenciada por el descentralizado.

- **Función de dispersión de puntos / point-spread function (PSF):**

En términos generales y en el contexto del procesamiento de señales, se define como la respuesta impulsional (respuesta en el dominio espacial) de un sistema óptico a un punto objeto. La respuesta impulsional o respuesta a un impulso de un sistema se define como la presente en la salida cuando en la entrada del mismo se introduce un impulso.

Para ello asumimos que se trata de un sistema LTI (Linear Time-Invariant) o sistema lineal e invariante en el tiempo, el cual produce una señal de salida para cualquier señal de entrada siguiendo las restricciones de linearidad e invarianza en el tiempo. Estos sistemas pueden ser caracterizados enteramente por una función llamada respuesta al impulso. La salida del sistema $y(t)$ es la convolución de la entrada al sistema $x(t)$ con la respuesta del impulso del sistema $h(t)$.

Se caracterizan en el dominio de la frecuencia a través de la función de transferencia del sistema que se trata de la transformada de Laplace de la respuesta al impulso del mismo (la transformada Z en el caso de sistemas discretos en el tiempo).

- **Linearidad:** La relación entre la entrada y la salida son el resultado de una ecuación diferencial lineal. Un sistema lineal relaciona una entrada $x(t)$ con una salida $y(t)$ con un escalado de la forma $a^*x(t)$ a $a^*y(t)$. Cumplen el principio de superposición.
- **Invarianza en el tiempo:** La aplicación de una entrada al sistema ahora o en T segundos producirá una salida idéntica excepto por un retraso de T segundos.

- **Función de transferencia óptica / optical-transfer function (OTF):**

Se entiende por función de transferencia óptica (optical-transfer function, OTF) a la transformada de Fourier de la PSF. Por tanto, contiene la misma información que la PSF pero en el dominio de la frecuencia.

3.2. Partes principales

Nuestro algoritmo trata de seguir los pasos indicados por el pseudocódigo del artículo, este consta de tres partes grandes a tener en cuenta: la inicialización de variables, las iteraciones correspondientes a la optimización con distintos valores de beta, y una normalización a los valores de L_1 en cada iteración. L_2 o el reflejo se obtiene mediante la diferencia de I con L_1 ,

Debemos tener en cuenta que debido a las transformaciones que se llevan a cabo en el algoritmo, se trata de ahorrar el máximo número de operaciones posibles dentro de la etapa iterativa. Así, contaremos con numerosas inicializaciones y precálculos que comentaremos a continuación.

- **Inicialización:** .

Se inicializa L_1 por defecto a la imagen original que se recibe por parámetro (en caso de recibirla será el proporcionado)

Se inicializan los filtros para la detección de bordes:

$$f1 = [[1,-1]] \quad f2 = [[1], [-1]] \quad f3 = [[0,-1,0],[-1,4,-1],[0,-1,0]]$$

$$A = \frac{\text{a)} \boxed{\beta \sum_j (\mathcal{F}(F^j)^\star \mathcal{F}(g^j))} + \text{b)} \boxed{\lambda \mathcal{F}(F^3)^\star \mathcal{F}(F^3) \mathcal{F}(I)}}{\text{c)} \boxed{\beta \sum_j (\mathcal{F}(F^j)^\star \mathcal{F}(F^j))} + \text{d)} \boxed{\lambda \mathcal{F}(F^3)^\star \mathcal{F}(F^3)} + \tau}$$

Figure 17. División de los componentes que forman la fórmula del cálculo de L_1 .

Atendiendo a la *Figure 17*, las únicas partes que podemos calcular previamente son b, c y d puesto que las variables que cambian su valor en las distintas iteraciones del algoritmo son G puesto que se produce una actualización de estas como primer paso del bucle.

- **Parte b):** Realizamos la FFT de f_3 (filtro Laplaciano) con la función psf2otf y realizamos una convolución de este con su conjugado. Convolucionamos el resultado con la transformada de la imagen.
- **Parte c):** Para cada filtro j (hasta 2, no incluye f_3) realizamos su FFT con psf2otf y convolucionamos esta transformada con su conjugado.
- **Parte d):** La convolución entre la FFT del filtro Laplaciano y su conjugado.

Ahora que se ha terminado con la parte previa de inicialización de variables y precálculos podemos seguir con la siguiente parte, un bucle que itera sobre distintos valores de β siguiendo las indicaciones del artículo (comienza en 20 y duplica por iteración) con $\beta = 2^{\frac{i-1}{umbral=0.05}}$. En la sección de experimentación se prueba con distintos valores del mismo y se sacan una serie de conclusiones a partir de estas pruebas.

3.2.1. Actualización de g^j :

Recordando la fórmula:

$$g_i^j = \begin{cases} F_i^j L_1, & (F_i^j L_1)^2 > \frac{1}{\beta} \\ 0, & \text{otherwise.} \end{cases}$$

Tenemos que para cada i, j el valor de g es 0 en caso de que el cuadrado de la convolución de L_1 con el filtro j no supere el umbral de $\frac{1}{\beta}$, por tanto en nuestra implementación necesitamos primero calcular cada convolución que sea necesaria, como j son valores en $\{1,2\}$, solo se aplican los filtros f_1 y f_2 , para hacer esta convolución usamos de OpenCV la función **filter2D**.

Después, mediante bucles que recorran toda la matriz de la convolución se aplica la función umbral (para la comprobar si se supera el umbral con los valores al cuadrado, hacemos la suma de los valores para los d canales).

La actualización mantendrá los nuevos valores que permiten distinguir las capas (en base a distinguir las intensidades de sus “bordes” o gradientes) en la capa del fondo L_1 sobre la que trabajamos durante todo el algoritmo.

3.2.2. Cálculo de L_1 :

Teniendo calculada gran parte de la formula antes de ya está calculada, solo nos queda el producto de la transformada de Fourier de g^j por el complejo conjugado de F^j .

3.2.3. Normalización de L_1 :

Aquí suponemos que el reflejo ya ha sido eliminado en L_1 , pero en el proceso hemos dejado la imagen en un rango de valores incorrecto, los valores que debe tener la imagen en cada valor i debe estar en el rango $[lb_i, hb_i]$, para ello nos basamos en la fórmula:

$$\min_t \sum_i m_i((L_1)_i + t - lb_i)^2 + \sum_i n_i((L_1)_i + t - ub_i)^2$$

Que consiste en minimizar el error cuadrático, tomando las funciones m_i y n_i como dos funciones que retornan 0 o 1, en nuestra implementación las usamos los operadores lógicos “ $<$ ” y “ $>$ ” para representarlos, y lo que hacemos consiste en un bucle que calcula para cada canal de la imagen la suma de todos los valores de los píxeles que salen del rango indicado en l_b y h_b , multiplicándolo por dos y dividiéndolo por el total de valores de la matriz de la imagen.

Si el error que representa esta salida del rango es inferior a cierto umbral ($1/(\text{total de valores})$ en nuestra implementación) se detiene el bucle de la normalización para ese canal de la imagen. Si ya no quedan más canales, la normalización ha terminado.

Solo queda hacer una última comprobación, los valores que superen la cota superior de h_b se quedan con el valor de h_b , y los que estén por debajo de la cota inferior l_b , se quedan con el valor de l_b .

4. Experimentación¹.

El ajuste de los parámetros del algoritmo no se trata de más que cambiar la función objetivo a optimizar con los pesos de los mismos. A continuación, se muestra dicha ecuación con el fin de facilitar la identificación de los distintos parámetros con los que se experimenta en la sección.

$$\min_{L_1, g^j} \sum_i \left(\sum_{j=1,2} (\beta(F_i^j L_1 - g_i^j)^2 + \rho(g_i^j)) + \lambda(F_i^3 L_1 - F_i^3 I)^2 \right)$$

¹ La sección de experimentación está ampliamente desarrollada en el notebook correspondiente al proyecto, aquí solo reflejamos algunos aspectos generales de esta fase.

Una de las imágenes con las que hemos hecho pruebas y su resultado:



Como se ve en el ejemplo, el reflejo ha sido casi completamente eliminado.

Hemos hecho distintas pruebas, entre ellas el cambio del valor de beta (mediante el cambio del valor umbral "thr" del que depende) y el cambio de lambda.

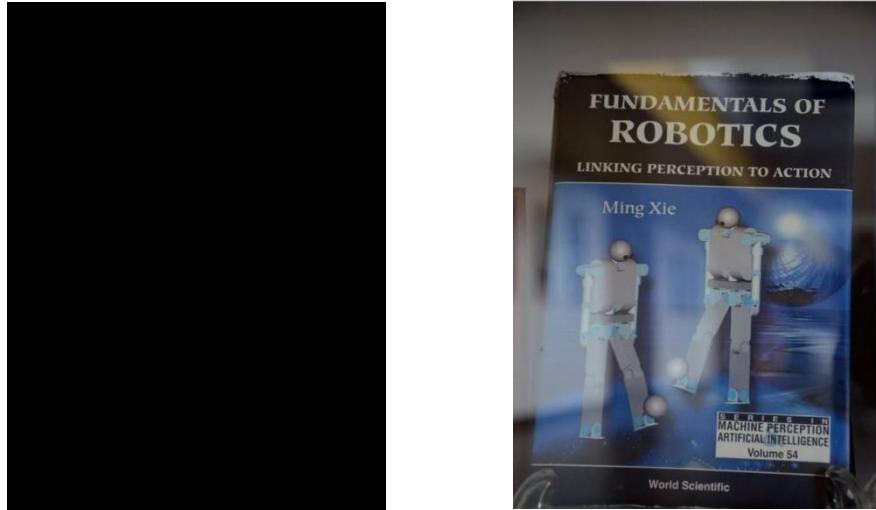
4.1. Alterando el valor de thr (β):

β actúa en la fórmula del cálculo de L1 dando peso a los filtros f_1 y f_2 y forma parte del valor umbral de la actualización de g ($1/\beta$).

En la actualización de g tenemos una convolución en el dominio del espacio que detecta los bordes de la imagen, para cada valor de g se comprueba si es inferior al umbral, esos valores que son inferiores al umbral son considerados como bordes del reflejo ya que no son lo suficientemente fuertes para pertenecer a los bordes del fondo, aumentar su valor implica disminuir ese umbral de la actualización de g y aumentar el peso de la convolución de g en el cálculo de L1.

Por tanto, si β toma un valor muy grande la imagen del fondo L1 obtenida será muy similar a la original, ya que al disminuir el umbral "se dejan pasar" todos los bordes de la imagen original y se le da un peso muy grande a ese cálculo en la fórmula del cálculo de L1.

Hacemos una prueba con un valor muy alto y otro muy bajo para probar lo que hemos dicho sobre los casos extremos de β :



Por lo que hemos visto, mayores valores de β implican aumentar β , lo cual significa dar mayor importancia a la convolución con los filtros de primer orden respecto al filtro laplaciano con la imagen original, es decir, se converge más rápido a una solución donde se elimine el reflejo, pero esto implica alejarse de la imagen original, se pierde mucha claridad, sin embargo, con valores excesivamente altos terminaremos por dejar L1 (capa del fondo) muy similar a la imagen original, ya que al hacer la umbralización en la actualización de g todos los valores pasan el umbral y después el cálculo de L1 el peso de esa convolución es mucho mayor que la convolución del laplaciano con la imagen original.

Tomar valores muy bajos de β impide que ningún valor en actualización de g sea capaz de pasar el umbral por tanto L1 quedará en negro.

4.2. Alterando el valor de λ :

Como se indica en el artículo, el parámetro lambda se utiliza para controlar la "suavidad" que se impone a la capa L2, en nuestro caso describe la intensidad del reflejo de la imagen original, es un parámetro que hay que ajustar dependiendo de la intensidad del reflejo de la imagen. De esta forma, la configuración de este parámetro con distintos valores puede proporcionarnos otra visión acerca de cómo ajustar con mayor precisión el algoritmo en según que casos.

Lo que se intenta conseguir con lambda y el filtro laplaciano es recuperar partes que se hayan eliminado como reflejo de la imagen original y que no lo eran (en función de la suposición inicial de como de intensa es la capa del reflejo).

Probamos primero con el valor 0, esto supone eliminar de la fórmula el filtro f_3 (Laplaciano), eliminar este filtro supone que la convolución del filtro laplaciano no aporta nada a L1, es decir, no intenta recuperar nada de la imagen original con tras el procesado mediante la convolución de g , lo que significa que la mayor parte de la imagen original es reflejo y L1 solo depende de la variable g .



Si le decimos al algoritmo que la intensidad del reflejo es demasiado alto, debe considerar que muy poco de la imagen es reflejo (el peso de la convolución de la imagen original con el laplaciano gana mucha fuerza), por tanto la salida debe ser muy parecida a la entrada.

El artículo dice que λ representa una estimación que hacemos de la suavidad relativa de la capa L2 en la separación de capas, para nuestro caso de la eliminación del reflejo, como hemos supuesto y visto en las pruebas, λ estima la intensidad del reflejo en la imagen, es decir, lo intensa que es la capa L2, para valores altos de λ , damos más peso a la parte del filtro laplaciano en la fórmula del cálculo de L1, lo cual implica que en cada iteración se elimina menor cantidad de reflejo ya que el peso de la convolución de la variable g es mucho menor, con un valor de λ extremadamente alto conseguiremos una L1 muy parecida a la imagen original.

Si tomamos λ con un valor de 0 lo que hacemos es eliminar ese filtro laplaciano de la fórmula del cálculo de L1 y estamos diciéndole al algoritmo que la intensidad del reflejo es extremadamente alta, por tanto, en la fórmula del cálculo de L1 se eliminan todos los bordes que no superan el umbral durante la actualización de g .

5. Conclusiones.

El artículo sigue la línea de numerosas investigaciones previas basadas en la dispersión del gradiente como base para distinguir entre capas con el fin de descomponer imágenes. Así, al ser un problema de gran complejidad consigue resultados relativamente decentes para la naturaleza de este y el estado del arte en el momento de la publicación (2014).

Se trata de un algoritmo que, a pesar de contar con ciertos parámetros ya ajustados para su ejecución, tiene sus mejores resultados para las imágenes de prueba proporcionadas por el artículo de forma que requieren una investigación por parte de aquel que pretenda usarlo para ajustarlo a cierto tipo de imágenes (ya que imágenes similares pueden recibir un tratamiento similar en este algoritmo).

A pesar de ello, supone una base para trabajos posteriores como *Single Image Reflection Suppression* de Nikolaos Arvanitopoulos, Radhakrishna Achanta and Sabine Susstrunk publicado en 2017 [3] y que se trata de la siguiente mejora utilizando esta técnica. Este artículo recopila todo el conjunto de artículos que tratan este problema y en base a ellos propone una solución basada en la dispersión del gradiente y el uso del Laplaciano para resolver el problema y posicionarse como el mejor en el momento.

Finalmente, concluir con que la elección de este trabajo ha estado condicionada por nuestra falta de conocimientos en el área de la inteligencia artificial, puesto que existen multitud de artículos que abordan este problema desde la perspectiva del aprendizaje profundo. La investigación de esta línea de trabajo que puede resultar más interesante es un buen punto de partida a partir de este trabajo.

De igual modo, la implementación e investigación del artículo de Nikolaos et al. [3] puede ser otro buen punto de partida tras asentar los conocimientos que se desarrollan en el que nosotros tratamos.

En cuanto a la mejora del algoritmo implementado, ya que parte directamente de la idea de implementación de los autores (ya optimizada), una mejora puede ser la búsqueda de cuellos de botella en la implementación en Python puesto que en la medida de lo posible se han utilizado librerías especializadas para cálculos matemáticos las cuales están muy optimizadas.

De igual forma, la implementación directa de las fórmulas en el código puede suponer un buen ejercicio para evaluar el rendimiento de ambas versiones.

Por otro lado, tratar de implementar otro algoritmo que trate de eliminar el reflejo y compararlo con la implementación que presentamos en Python junto a su rendimiento puede ser muy interesante. Algunos algoritmos que tratan el problema y se comparan en diversos artículos con el nuestro (conocido como LB14 por las siglas de sus autores y el año de publicación) son el SK15 [6] (la primera mejora después del nuestro), el SPBS-M [2], el SID [5], el WS16 [4] y el AN17 [3].

6. Autoevaluación.

- Álvaro Campillos Delgado

Comprensión y dominio del tema	1
Exposición didáctica	1
Integración del equipo	1
Objetivos	0.65
Aspectos didácticos	0,65
Trabajo reutilizable	1
Experimentación y conclusiones	1
Contenidos	0.65
Divulgación de los contenidos	1
Organización de documentación	0.65
Bibliografía. Uso de recursos científicos	1

- **Marcel Espejo Cuenca.**

Comprensión y dominio del tema	1
Exposición didáctica	1
Integración del equipo	1
Objetivos	0.65
Aspectos didácticos	1
Trabajo reutilizable	1
Experimentación y conclusiones	1
Contenidos	0.65
Divulgación de los contenidos	1
Organización de documentación	0.65
Bibliografía. Uso de recursos científicos	1

• **Comprensión y dominio del tema:**

Sin haber estudiado anteriormente en otras asignaturas nada relacionado con transformadas de Fourier ni sobre el dominio de la frecuencia, podemos defender con bastante seguridad los conceptos que se tratan en el paper y el trabajo.

• **Exposición didáctica:**

Hemos tratado de hacer lo más amena posible la presentación tratando de expresar la idea general de como funciona nuestro algoritmo tratando de distinguir las distintas partes del mismo para facilitar su comprensión sin que resulte demasiado pesado. Y creemos que se ha conseguido.

• **Integración del equipo:**

Ambos miembros del grupo responden a las preguntas y trabajan por igual.

• **Objetivos:**

Se cumplen todos los Objetivos revisados (11/1/2020) excepto el objetivo 3, la comprobación de la eficiencia del algoritmo del paper en comparación con otros (a partir de artículos de investigación, código de los mismos en nuestro hardware, etc.).

• **Aspectos didácticos:**

Con el notebook se puede seguir cada parte del algoritmo paso a paso para facilitar la comprensión del mismo lo máximo posible.

• **Trabajo reutilizable:**

El código está muy comentado y el propio notebook hace la función de un léeme ya que explica cómo funciona cada parte del algoritmo, también se puede copiar toda la función completa desde la celda en la que se encuentra definida.

- **Experimentación y conclusiones:**

Se ha dedicado mucho tiempo a la experimentación y se ha tratado de razonar antes y después de obtener los resultados con el fin de comprender el papel de cada parte del algoritmo.

- **Contenidos:**

Es cierto que hay mucho contenido pero es muy denso y por falta de tiempo no hemos podido plasmarlo en la documentación al completo.

- **Divulgación de los contenidos:**

Hemos tratado de presentar los contenidos de la forma más divulgativa posible.

- **Organización de documentación:**

Tratamos de organizar la documentación de la mejor forma posible pero tampoco ha sido nuestra principal preocupación durante el desarrollo del trabajo y puede que se haya visto perjudicada.

- **Bibliografía. Uso de recursos científicos:**

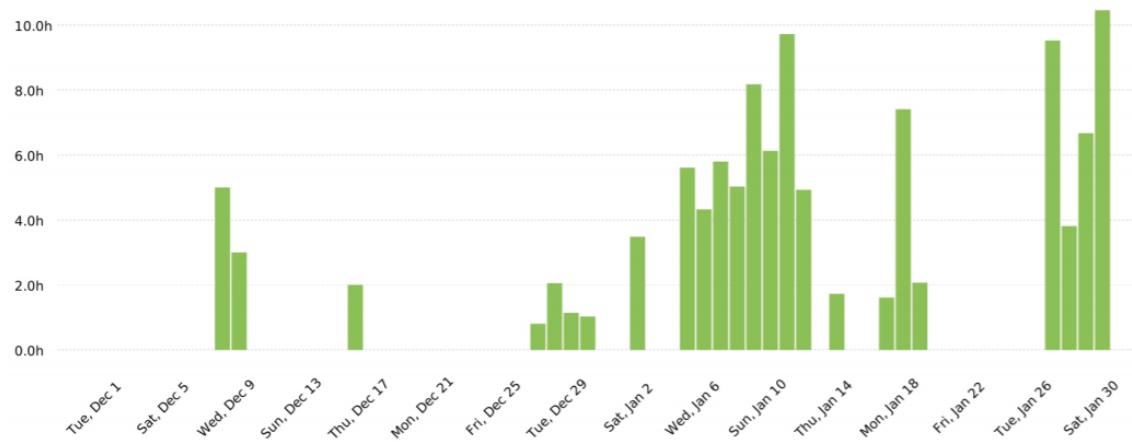
Hemos consultado gran cantidad de recursos para poder entender el artículo y la hemos dejado en la documentación.

7. Tabla de tiempos.

- Marcel Espejo Cuenca.

Fecha de actividad	Tiempo	Actividad realizada
7/12/2020	5:00:00	Búsqueda del TA
8/12/2020	3:00:00	Búsqueda de un artículo
15/12/2020	2:00:00	Cuestionario TA
26/12/2020	00:48:10	Lectura y comprensión
27/12/2020	2:03:10	Lectura y comprensión
28/12/2020	1:08:27	Lectura y comprensión
29/12/2020	1:05:35	Lectura y comprensión
1/01/2021	3:28:52	Lectura y comprensión
4/01/2021	5:36:23	Diseño del algoritmo
5/01/2021	00:26:13	Escritura de documentación
5/01/2021	3:53:23	Diseño del algoritmo
6/01/2021	5:47:53	Diseño del algoritmo
7/01/2021	3:31:20	Escritura de documentación
7/01/2021	1:22:27	Diseño del algoritmo
8/01/2021	8:10:35	Diseño del algoritmo
9/01/2021	3:26:38	Diseño del algoritmo
9/01/2021	2:40:56	Comprensión pseudocódigo
10/01/2021	4:53:45	Implementación del algoritmo
10/01/2021	4:49:40	Escritura de la documentación
11/01/2021	4:55:43	Implementación del algoritmo
13/01/2021	1:43:38	Diseño del algoritmo
16/01/2021	00:11:40	Escritura de la documentación
16/01/2021	1:25:00	Experimentación, testing y debugging
17/01/2021	00:57:23	Profundización de los conceptos teóricos
17/01/2021	3:23:43	Experimentación, testing y debugging
17/01/2021	3:03:25	Creación de la presentación
18/01/2021	2:03:55	Escritura de la documentación
26/01/2021	5:32:29	Escritura de la documentación
26/01/2021	3:59:01	Experimentación, testing y debugging
27/01/2021	3:48:35	Escritura de la documentación
28/01/2021	6:04:46	Experimentación, testing y debugging
28/01/2021	00:35:33	Escritura de la documentación

Total: 111:30:03 Billable: 111:30:03 Amount: 0.00 USD



Tag



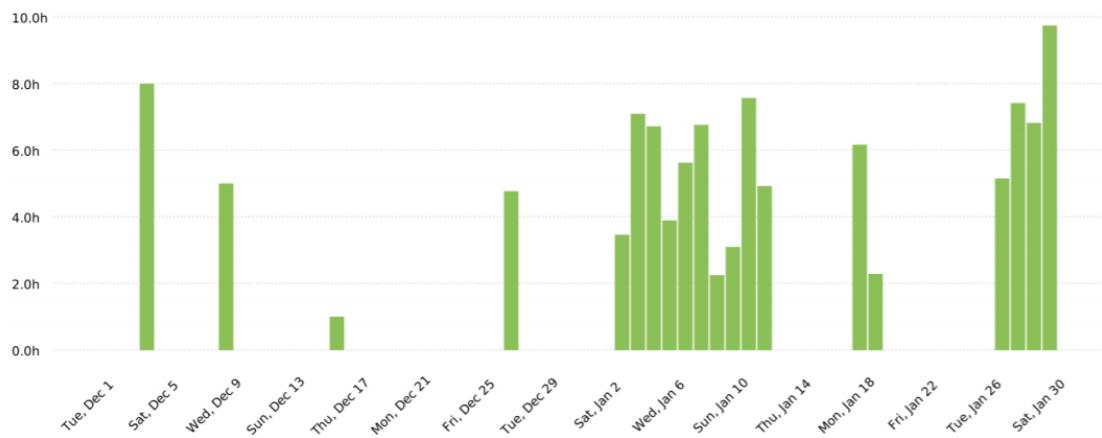
● Búsqueda de un artículo	03:00:02	2.69%
● Búsqueda del TA	05:00:01	4.48%
● Comprensión Pseudocódigo	02:40:56	2.40%
● Creación de la presentación	03:03:25	2.74%
● Cuestionario TA	02:00:02	1.79%
● Diseño del algoritmo	28:46:57	25.81%
● Diseño del algoritmo, Escritura de la documentación	03:31:20	3.16%
● Escritura de la documentación	14:25:09	12.93%
● Experimentación, testing y debugging	15:24:54	13.83%
● Implementación del algoritmo	09:49:28	8.81%
● Lectura y compresión.	08:30:14	7.62%
● Profundización en los conceptos teóricos	00:57:23	0.86%
● Prueba	08:47:43	7.89%
● Prueba, Escritura de la documentación	05:32:29	4.97%

- Álvaro Campillos Delgado.

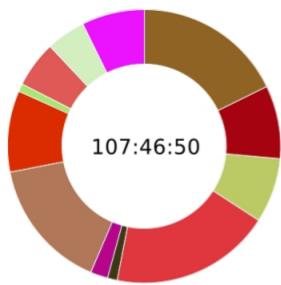
Fecha de actividad	Tiempo	Actividad realizada
3/12/2020	08:00:00	Búsqueda de un artículo
8/12/2020	05:00:00	Búsqueda del TA
15/12/2020	01:00:01	Cuestionario TA
26/12/2020	04:46:15	Lectura y compresión
2/01/2020	03:27:48	Lectura y compresión.
3/01/2020	07:05:46	Diseño del algoritmo
4/01/2021	01:24:00	Diseño del algoritmo
4/01/2021	05:19:03	Profundización en los conceptos teóricos
5/01/2021	03:53:38	Implementación del algoritmo
6/01/2021	00:43:11	Diseño del algoritmo
6/01/2021	04:54:32	Escritura de la documentación
7/01/2021	01:08:03	Diseño del algoritmo
7/01/2021	00:55:00	Escritura de la documentación
7/01/2021	03:42:55	Implementación del algoritmo
7/01/2021	01:00:00	Profundización en los conceptos teóricos
8/01/2021	02:15:01	Implementación del algoritmo
9/01/2021	03:05:53	Implementación del algoritmo
10/01/2021	07:34:23	Implementación del algoritmo
11/01/2021	04:55:31	Escritura de la documentación
17/01/2021	03:10:01	Escritura de la documentación
17/01/2021	03:00:00	Profundización en los conceptos teóricos
18/01/2021	01:39:52	Escritura de la documentación
18/01/2021	00:37:25	Profundización en los conceptos teóricos
26/01/2021	03:55:46	Creación de la presentación
26/01/2021	01:13:32	Escritura de la documentación
27/01/2021	01:58:31	Creación de la presentación
27/01/2021	02:09:26	Creación de la presentación
27/01/2021	01:16:29	Experimentación, testing y debugging
27/01/2021	02:00:38	Escritura de la documentación
28/01/2021	06:49:19	Escritura de la documentación
29/01/2021	09:44:51	Escritura de la documentación

12/01/2020 - 01/30/2021

Total: 107:46:50 Billable: 107:46:50 Amount: 0.00 USD



Tag

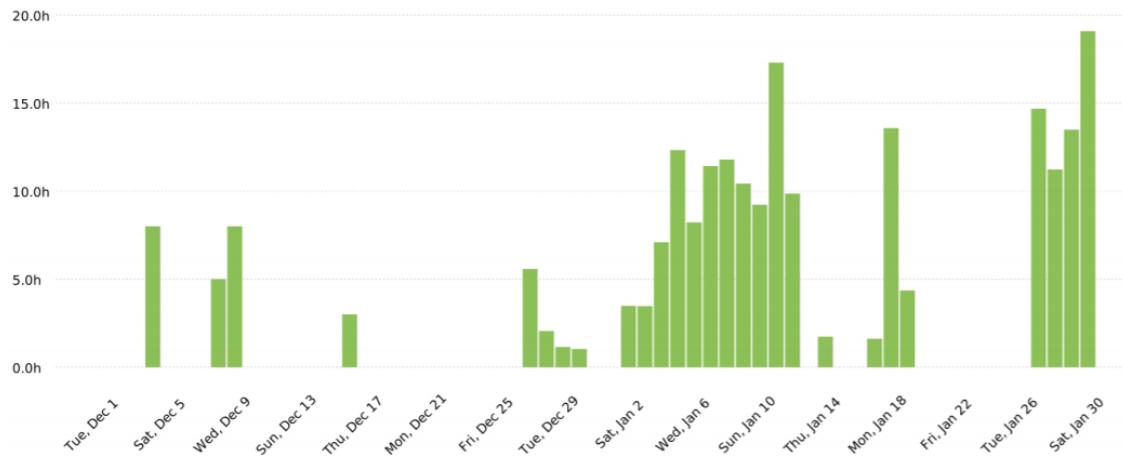


● Búsqueda de un artículo	08:00:00	7.42%
● Búsqueda del TA	05:00:00	4.64%
● Creación de la presentación	05:54:17	5.47%
● Cuestionario TA	01:00:01	0.93%
● Diseño del algoritmo	10:21:00	9.60%
● Escritura de la documentación	16:48:28	15.60%
● Escritura de la documentación, Creación de la presentación	02:09:26	2.00%
● Experimentación, testing y debugging	01:16:29	1.18%
● Implementación del algoritmo	20:31:50	19.05%
● Lectura y compresión.	08:14:03	7.64%
● Profundización en los conceptos teóricos	09:19:03	8.65%
● Profundización en los conceptos teóricos, Escritura de la documentación	19:12:13	17.81%

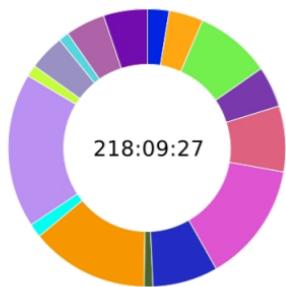
- Trabajo total:

12/01/2020 - 01/30/2021

Total: **218:09:27** Billable: **218:09:27** Amount: **0.00 USD**



Tag



● Búsqueda de un artículo	11:00:02	5.04%
● Búsqueda del TA	10:00:01	4.58%
● Comprensión Pseudocódigo	02:40:56	1.23%
● Creación de la presentación	08:57:42	4.11%
● Cuestionario TA	03:00:03	1.38%
● Diseño del algoritmo	39:07:57	17.94%
● Diseño del algoritmo, Escritura de la documentación	03:31:20	1.61%
● Escritura de la documentación	30:06:11	13.80%
● Escritura de la documentación, Creación de la presentación	02:09:26	0.99%
● Experimentación, testing y debugging	16:41:23	7.65%
● Implementación del algoritmo	30:21:18	13.92%
● Lectura y compresión.	16:44:17	7.67%
● Profundización en los conceptos teóricos	10:16:26	4.71%

8. Referencias.

- [1]. D. Geman and C. Yang. Nonlinear image recovery with halfquadratic regularization. TIP, 4(7):932–946, 1995.
 - [2]. K. Gai, Z. Shi, and C. Zhang. Blind separation of superimposed moving images using image statistics. IEEE Transactions on Pattern Analysis and Machine Intelligence, 34(1):19–32, Jan 2012.
 - [3]. N. Arvanitopoulos, R. Achanta, and S. Susstrunk. Single image reflection suppression. In CVPR, 2017.
 - [4]. R. Wan, B. Shi, A. H. Tan, and A. C. Kot. Depth of field guided reflection removal. In Proc. ICIP, 2016.
 - [5]. X. Guo, X. Cao, and Y. Ma. Robust separation of reflection from multiple images. In Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on, pages 2195–2202, June 2014.
 - [6]. Y. C. Shih, D. Krishnan, F. Durand, and W. T. Freeman. Reflection removal using ghosting cues. The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 3193–3201.
 - [7]. Y. Li and M. S. Brown. Single Image Layer Separation Using Relative Smoothness. In IEEE Conference on Computer Vision and Pattern Recognition, pages 2752–2759, 2014
 - [8]. Y. Wang, J. Yang, W. Yin, and Y. Zhang. A new alternating minimization algorithm for total variation image reconstruction. SIAM Journal on Imaging Sciences, 1(3):248–272, 2008.
 - [9]. Y. Y. Schechner, N. Kiryati, and R. Basri. Separation of transparent layers using focus. IJCV, 39(1):25–39, 2000
-
- R.F.Gonzalez and R.E.Woods. Digital Image Processing (4 ed.). Pearson.
 - Distribuciones de probabilidad (cola larga, pesada, etc.):
https://en.wikipedia.org/wiki/Heavy-tailed_distribution
https://en.wikipedia.org/wiki/Long_tail
 - Filtrado Gaussiano y separabilidad: http://www-edlab.cs.umass.edu/~smaji/cmpsci370/slides/hh/lec02_hh_advanced_edges.pdf
 - Uso de log-probabilidades y negativo de la minimización:
https://en.wikipedia.org/wiki/Log_probability
 - Operador de Laplace: https://en.wikipedia.org/wiki/Laplace_operator
https://en.wikipedia.org/wiki/Discrete_Laplace_operator
https://docs.opencv.org/3.4/d5/db5/tutorial_laplace_operator.html
<https://homepages.inf.ed.ac.uk/rbf/HIPR2/log.htm>
 - Programación cuadrática: https://en.wikipedia.org/wiki/Quadratic_programming
 - Transformada de Fourier: https://en.wikipedia.org/wiki/Fourier_transform
https://en.wikipedia.org/wiki/Convolution_theorem
https://en.wikipedia.org/wiki/Fourier_inversion_theorem
https://en.wikipedia.org/wiki/Frequency_domain
<https://en.wikipedia.org/wiki/Deconvolution>
 - Funciones de transferencia:
https://en.wikipedia.org/wiki/Point_spread_function
https://en.wikipedia.org/wiki/Linear_time-invariant_system
https://en.wikipedia.org/wiki/Impulse_response
https://en.wikipedia.org/wiki/Optical_transfer_function

- Transformada rápida de Fourier: https://en.wikipedia.org/wiki/Fast_Fourier_transform
https://en.wikipedia.org/wiki/Cooley%E2%80%93Tukey_FFT_algorithm
<https://towardsdatascience.com/fast-fourier-transform-937926e591cb>
- Descenso por el gradiente:
https://en.wikipedia.org/wiki/Gradient_descent
Russell, S. J., Norvig, P., & Davis, E. (2010). *Artificial intelligence: a modern approach*. 3rd ed. Upper Saddle River, NJ: Prentice Hall.
- Página de uno de los creadores del artículo con enlace a este: <https://yu-li.github.io/>
- OpenCV – <https://opencv.org>
- Numpy – <https://numpy.org/doc/stable/>