ScienceDirect
Disponible en www.sciencedirect.com
Revista Iberoamericana de Automática e Informática industrial 13 (2016) 450–461
www.elsevier.es/RIAI

Fusión de Imágenes Multi Foco basado en la Combinación Lineal de Imágenes utilizando Imágenes Incrementales

Felix Calderon[1,*], Adan Garnica-Carrillo, Juan J. Flores

División de Estudios de Posgrado.
Facultad de Ingeniería Eléctrica.
Universidad Michoacana de San Nicolas de Hidalgo

Resumen

En este artículo presentamos tres algoritmos para calcular la fusión de imágenes multi foco. Estos algoritmos se basan en la combinación lineal de un par de imágenes con diferentes niveles de enfoque. Los tres algoritmos maximizan una función lineal con restricciones de coherencia espacial; el objetivo de presentarlos es justificar como llegamos a plantear un algoritmo rápido y simple. El primer algoritmo llamado Combinación Lineal de Imágenes (CLI), se implementó utilizando Wolfram Mathematica, pero dado el número de variables a optimizar, la solución demandó de mucho tiempo de cómputo. El segundo algoritmo llamado Combinación Lineal de Imágenes por Ventanas (CLI-V) es una aplicación, sobre subregiones de las imágenes del algoritmo CLI, mejorando el desempeño en tiempo y logrando la implementación con el método Simplex. El tercer algoritmo llamado Combinación Lineal de Imágenes Simple (CLI-S), es una simplificación del algoritmo CLI-V, con resultados de calidad muy similares a los algoritmos CLI y CLI-V y a algunos algoritmos del estado del arte, pero con tiempos de solución muy rápidos. El algoritmo CLI-S se implementó utilizando imágenes incrementales con el propósito de tener soluciones en centésimas de segundo para las imágenes de prueba utilizadas. Para los tres algoritmos se presenta el desempeño y el tiempo de solución bajo condiciones similares, utilizando un par de imágenes sintéticas y cuatro pares de imágenes reales. Las imágenes reales han sido utilizadas por algoritmos del estado del arte y fueron seleccionadas con el objetivo de que el lector pueda hacer una comparación cualitativa. En el caso del par de imágenes sintéticas se hace una comparación cuantitativa con resultado de 98 % de aciertos en la selección de píxeles, en un tiempo de ejecución de 0.080 s. para una imagen de 512×512 píxeles, lo que nos permite decir que la velocidad lograda con algoritmo CLI-S permite efectuar el proceso de fusión en tiempo real, situación que no hemos encontrado reportada en el estado del arte.

Palabras Clave: Programación lineal, fusión de imágenes multi foco, filtros pasa altas, imágenes incrementales

1. Introducción

Todas las cámaras, microscopios y equipos que utilizan lentes tienen una limitación en la nitidez producida por la profundidad de campo (Kuthirummal et al., 2011). La profundidad de campo es el término utilizado en óptica y en fotografía para expresar el rango de distancias reproducidas por una lente, donde la imagen está en foco. Debido a las características de la lente de una cámara, es imposible tener profundidad de campo infinita dando como resultado que la imagen no sea totalmente nítida. Por ejemplo, si una cámara tiene una lente con poca profundidad de campo no será posible tener la misma nitidez en objetos que se encuentren a profundidades diferentes del plano focal del sistema óptico; será necesario decidir sobre qué objeto se desea tener la máxima nitidez o enfoque. En la Figura 1 se muestra una fotografía tomada con una cámara cuya lente tiene poca profundidad de campo; note que solamente se tiene definición en el plano de la flor mientras que el fondo aparece borroso. La fusión de imágenes multi foco se define como el proceso de combinar la información nítida presente en dos o más imágenes de una misma escena capturadas bajo diferentes condiciones de enfoque, con el propósito de obtener una imagen donde se tenga la mejor nitidez posible sobre la mayor parte de los objetos de interés.

Sea un conjunto de imágenes $I = \{I_k | k = 1, 2, , 3 \dots\}$, donde la k-ésima imagen I_k es un arreglo bidimensional e $I_k(y, x)$ representa el color correspondiente al renglón y (coordenada vertical) y la columna x (coordenada horizontal) de la imagen

*Autor en correspondencia.
Correos electrónicos: calderon@umich.mx (Felix Calderon), agarnica@dep.fie.umich.mx (Adan Garnica-Carrillo), juanf@umich.mx (Juan J. Flores)

http://dx.doi.org/10.1016/j.riai.2016.07.002

Figura 1: Fotografía tomada con cámara cuya lente tiene poca profundidad de campo

dentro de una retícula $L = [1 \cdots n_r] \times [1 \cdots n_c]$, donde n_r es el número de renglones y n_c el número de columnas de la imagen. Una manera de modelar la pérdida de nitidez de una imagen I_0 es convolucionándola con un kernel de acuerdo con (1).

$$J_0(y, x) = I_0(y, x) * G(y, x; \mu, \sigma) \tag{1}$$

donde * indica la operación de convolución con un kernel de tamaño $w \times w$, definida en (2).

$$J_0(y, x) = \sum_{k=1}^{w} \sum_{l=1}^{w} h(k, l) I_0(y - k, x - l) \tag{2}$$

Algunos autores (Elder and Zucker, 1998; Bae and Durand, 2007; Riaz et al., 2008) coinciden en que el kernel de convolución que mejor modela la pérdida de nitidez es el Gaussiano $G(y, x, \mu, \sigma)$, el cual está dado por (3):

$$\begin{aligned} G(y, x, \mu, \sigma) &= g(y, \mu, \sigma) \times g(x, \mu, \sigma) \\ g(y, \mu, \sigma) &= \tfrac{1}{\sqrt{2\pi}\sigma} exp\left(-\tfrac{(y-\mu)^2}{2\sigma^2}\right) \end{aligned} \tag{3}$$

donde $g(y, \mu, \sigma)$ es una función Gaussiana con media μ y desviación estándar σ.

Para resolver la fusión de dos imágenes tenemos que considerar que la pérdida de nitidez es sobre algunas partes de la imagen y que no tenemos información de los valores de desviación estándar σ que la afectan; así que necesitamos encontrar automáticamente las regiones borrosas y restaurarlas. Una forma de restaurar la imagen es aplicando el filtro de Wiener (Wiener, 1964), suponiendo que conocemos el valor de σ.

Para simular la pérdida de nitidez, a partir de una imagen original I_0 y su correspondiente imagen emborronada J_0 que previamente ha sido calculada utilizando (1), creamos un par de imágenes sintéticas I_1 e I_2 que simulen que fueron tomadas con una cámara con una lente de poca profundidad de campo, donde en la primera se enfocó en un plano y en la segunda en otro. Para tal propósito se define un peso binario $p_0(y, x)$ para cada uno de los píxeles de las imágenes I_1 e I_2 las cuales se construirán utilizando la regla dada por (4).

$$\begin{aligned} I_1(y, x) &= J_0(y, x) p_0(y, x) + I_0(y, x)(1 - p_0(y, x)) \\ I_2(y, x) &= J_0(y, x)(1 - p_0(y, x)) + I_0(y, x) p_0(y, x) \end{aligned} \tag{4}$$

Note que las imágenes I_1 e I_2 tendrán información nítida de la imagen original I_0 en forma complementaria, es decir un píxel no aparecerá borroso en ambas imágenes.

En la Figura 2 se muestra un ejemplo de cómo se utilizó (4) para simular la pérdida de nitidez. La Figura 2(a) muestra la imagen original I_0 donde aparecen dos lobos, la Figura 2(b) es un arreglo de pesos p_0 que utilizamos para generar las imágenes I_1 de la Figura 2(c) e I_2 de la Figura 2(d). En la Figura 2(c) se muestra la imagen sintética que simula un enfoque en el lobo de enfrente y desenfoque en el lobo de atrás, mientras que en la Figura 2(d) se muestra la misma imagen pero con las condiciones de desenfoque de manera complementaria. Adicionalmente, dado que en la literatura se plantea que el uso del kernel Gaussiano (que es un filtro pasa bajas) modela el desenfoque, podemos concluir que la pérdida de nitidez es equivalente a perder la componente de alta frecuencia en algunas regiones de la imagen.

(a) Imagen de dos lobos I_0 (b) Matriz de pesos p_0

(c) Imagen sintética I_1 (d) Imagen sintética I_2

Figura 2: Simulación de perdida de nitidez

Nuestra hipótesis es considerar que el proceso de fusión de imágenes multi foco, es un proceso inverso a lo planteado por (4) y consiste en calcular los valores de una matriz de pesos p que nos garantice tener la máxima nitidez en la imagen fusionada.

Este artículo está organizado de la siguiente manera, en la Sección 1 se hace una introducción del problema de fusión de imágenes multi foco. En la Sección 2 se describen algunas estrategias, del estado del arte, para realizar la fusión. En la sección 3 se presenta el algoritmo Combinación Lineal de Imágenes (CLI). En la Sección 4 se presenta una aplicación del algoritmo CLI sobre Ventanas (CLI-V) de la imagen. En la Sección 5 se presenta una simplificación del algoritmo CLI-V, al que

denominamos CLI-S (S por Simple), el cual permite resolver el problema en tiempos muy competitivos. En la Sección 6 se presentan los resultados con cinco pares de imágenes, de las cuales un par es sintético y cuatro son pares de imágenes reales ampliamente utilizados en trabajos publicados previamente. Con estos resultados queremos mostrar el desempeño de los algoritmos CLI, CLI-V y CLI-S para la fusión de imágenes. Finalmente, en la Sección 7, se presentan las conclusiones de este trabajo.

2. Antecedentes

Un método muy simple para realizar la fusión de las imágenes consiste en hacer el promedio del color en cada píxel utilizando (5). Este método puede considerarse como una combinación lineal de las imágenes multiplicadas por un peso de 0.5.

$$I_F(y,x) = 0.5I_1(y,x) + 0.5I_2(y,x) \tag{5}$$

Existen varios métodos propuestos para realizar la fusión de imágenes multi foco basados en redes neuronales. Li *et al* en (Li et al., 2002) hacen el proceso de fusión utilizando una Red Neuronal Artificial. Un trabajo más reciente en ésta dirección, lo presentan Pagidimarry y Babu (Pagidimarry and Babu, 2011). Ambos enfoques, dividen la imagen en bloques pero Pagidimarry y Babu utilizan un tamaño adaptable de bloque en lugar de que el tamaño de bloque sea constante.

Li *et al* en (Li et al., 2001) proponen una fusión eficiente de imágenes a nivel de bloque, basada en la frecuencia espacial o nivel de actividad. La frecuencia espacial se calcula para un bloque de la imagen I_k como:

$$S_k(y,x) = \sqrt{(R_k(y,x))^2 + (C_k(y,x))^2} \tag{6}$$

donde $R_k(y,x)$ y $C_k(y,x)$ son las estimaciones promedio de la alta frecuencia por renglón y columna respectivamente, correspondientes a la k-ésima imagen sobre una ventana de tamaño $w_r \times w_c$ centrada en el píxel de coordenadas (y,x).

$$R_k(y,x) = \sqrt{\frac{\sum_{i=y-w_r}^{y+w_r}\sum_{j=x-w_c}^{x+w_c}[I_k(i,j) - I_k(i,j-1)]^2}{(2w_r+1)(2w_c+1)}} \tag{7}$$

$$C_k(y,x) = \sqrt{\frac{\sum_{i=y-w_r}^{y+w_r}\sum_{j=x-w_c}^{x+w_c}[I_k(i,j) - I_k(i-1,j)]^2}{(2w_r+1)(2w_c+1)}} \tag{8}$$

Así dadas dos imágenes I_1 e I_2 de la misma escena, donde la primera se enfocó en el plano 1 y la segunda en el plano 2, la regla de fusión usada por Li *et al* en (Li et al., 2001) está dada por (9) y (10)

$$\begin{gathered} B_k(y,x) = \{I_k(i,j) | i \in [y-w_r, y+w_r], \\ j \in [x-w_c, x+w_c]\} \\ k = 1,2 \end{gathered} \tag{9}$$

$$I_F(y,x) = \begin{cases} B_1(y,x) & \text{Si } S_1(y,x) > S_2(y,x) + T_h \\ B_2(y,x) & \text{Si } S_1(y,x) < S_2(y,x) - T_h \\ \frac{B_1(y,x)+B_2(y,x)}{2} & \text{en otro caso} \end{cases} \tag{10}$$

donde $B_k(y,x)$ es un bloque centrado en las coordenadas (y,x) y T_h es un umbral. En esta implementación se selecciona el tamaño de bloque que tenga mayor respuesta al nivel de actividad, lo que garantiza que la imagen final I_F tenga los bloques con mayor nivel de actividad o alta frecuencia espacial.

Pajares y de la Cruz (Pajares and de la Cruz, 2004) presentan un tutorial completo de como utilizar la transformada wavelet para hacer la fusión de imágenes multi foco. La transformada wavelet crea una representación de la imagen en diferentes resoluciones, mediante una descomposición de la frecuencia espacial, utilizando una familia de kérneles. La información de la transformada wavelet es utilizada para hacer la fusión utilizando un criterio a nivel píxel, muy similar al presentado en (10). La transformada wavelet básicamente consiste en hacer la convolución con un kernel y aplicar submuestreos. Pajares y de la Cruz utilizan el wavelet de Haar dado por (11), el cual puede ser visto como un filtro pasa bajas l y un filtro pasa altas h respectivamente.

$$\begin{aligned} l &= \left[\tfrac{1}{\sqrt{2}}, \tfrac{1}{\sqrt{2}}\right] \\ h &= \left[\tfrac{1}{\sqrt{2}}, -\tfrac{1}{\sqrt{2}}\right] \end{aligned} \tag{11}$$

Orozco (Orozco, 2013) determina la imagen fusionada como la combinación lineal de las imágenes, multiplicadas por un peso $\hat{p}$, el cual se calcula aplicando un algoritmo de segmentación denominado ECQMMFM, presentado por Rivera *et al* (Rivera et al., 2007). Los valores iniciales $\hat{p}_0$ se obtienen a partir del valor absoluto de la respuesta de un filtro pasa altas $\hat{F}_k$. Si bien el algoritmo para el cálculo de la segmentación es muy robusto, la segmentación se realiza sin tomar en cuenta información de la nitidez de las imágenes originales, ya que solamente se segmentan los valores iniciales de $\hat{p}_0$ dado por (12).

$$\hat{p}_0(y,x) = \begin{cases} 1 & \text{Si } \hat{F}_1(y,x) > \hat{F}_2(y,x) \\ 0 & \text{en caso contrario} \end{cases} \tag{12}$$

Cao *et al.* (Cao et al., 2015) proponen un procedimiento de fusión donde utilizan la Transformada Discreta Coseno (DCT por sus siglas en inglés). Su propuesta comienza decodificando y haciendo una descuantificación de las imágenes, para posteriormente dividirlas en bloques de tamaño 8×8. Cada bloque en la posición (i,j) es definido por $B_1(i,j)$ y $B_2(i,j)$. Calculan la frecuencia espacial de cada bloque de la imagen utilizando la DCT y los denominan $SF_1(i,j)$ y $SF_2(i,j)$. Para el proceso de fusión se hace un cálculo de una variable $W(i,j)$ utilizando la regla dada por (13).

$$W(i,j) = \begin{cases} 1 & SF_1(i,j) > SF_2(i,j) + T \\ -1 & SF_1(i,j) < SF_2(i,j) + T \\ 0 & SF_1(i,j) = SF_2(i,j) + T \end{cases} \tag{13}$$

Posteriormente calcula un mapa de decisión R, en una ventana de 3×3 sobre W, por medio de un filtro de mayoría dado por (14).

$$R(i,j) = \sum_{y=i-1}^{i+1} \sum_{x=j-1}^{j+1} W(y,x) \tag{14}$$

Para obtener la representación de la DCT de la imagen fusionada basada en R aplica la siguiente regla

$$F(i,j) = \begin{cases} B_1(i,j) & R(i,j) > 0 \\ B_2(i,j) & R(i,j) < 0 \\ \frac{B_1(i,j)+B_2(i,j)}{2} & R(i,j) = 0 \end{cases}$$

Finalmente cuantizan los coeficientes $F(i,j)$ resultantes con una tabla de cuantización estándar para JPEG. Note la similitud que existe entre los enfoque presentados por Cao *et al.* en (Cao et al., 2015) y Li *et al* en (Li et al., 2001).

Otros ejemplos de aplicación de filtros para la detección de regiones nítidas pueden encontrarse en (Li and Yang, 2008; Zhang and long Guo, 2009; Redondo et al., 2009; Chai et al., 2011), donde adicionalmente aplican alguna técnica diferente para realizar la fusión. Así por ejemplo, Li y Yang en (Li and Yang, 2008) aplican técnicas de segmentación para determinar la fusión de imágenes.

En general estos métodos proponen:

1. Calcular la respuesta espacial de alta frecuencia de cada una de las imágenes fuente,
2. Hacer una selección de los píxeles o bloques de píxeles con la mayor respuesta a la alta frecuencia y
3. Aplicar una regla de fusión para obtener una imagen con la mayor nitidez

Un trabajo más reciente en la fusión de imágenes multi foco es el presentado por Alonso *et al* en (Alonso et al., 2015), en donde se hace una propuesta para fusionar un conjunto com más de dos imágenes multi foco, partiendo del supuesto de que cada píxel en la imagen es una combinación de una región enfocada y una región desenfocada la cual viene de una combinación en 2D de las partes enfocadas de otras imágenes, en éste articulo los autores reportan muy buenos resultados cualitativamente, aunque no se reporta el tiempo necesario para encontrar la solución, ni un mapa de decisión final.

2.1. *Cálculo de la respuesta de alta frecuencia*

Las primeras y segundas derivadas de una imagen son filtros pasa altas, el nivel de actividad propuesto por Li *et al* (Li et al., 2001) es el promedio de las primeras derivadas discretizadas sobre una ventana y por lo tanto también es un filtro pasa altas.

El Laplaciano se define como la suma de las segundas derivadas en x y y dado por (15), cuyo kernel de convolución discreto es (16). El kernel del Laplaciano discretizado lo podemos utilizar para realizar la convolución con la imagen y la respuesta es un medidor del nivel actividad en cuatro direcciones comparando con el trabajo de Li *et al.* en (Li et al., 2001).

$$\Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \tag{15}$$

$$\Delta = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{16}$$

Burt y Adelson definen las pirámides Laplacianas y las utilizan para compresión de imágenes (Burt and Adelson, 1983). Burt y Kolczynski realizan una aproximación del Laplaciano utilizando diferencias de Gaussianas y utilizan la pirámide Laplaciana para hacer la fusión de imágenes multi foco (Burt and Kolczynski, 1993). Existen otras aproximaciones para el Laplaciano; la más utilizada es la diferencia de Gaussianas DOG (Burt and Adelson, 1983). Esta aproximación se lleva a cabo calculado la convolución de una señal con una diferencias de Gaussianas $G(y,x,\mu,\sigma\tau)$ con diferente escala τ, dada por (17),

$$DoG(y,x,\sigma,\tau) = G(y,x,0,\sigma) - G(y,x,0,\sigma\tau) \tag{17}$$

Orozco (Orozco, 2013) utiliza la convolución de las imágenes originales I_k con un kernel h_σ dado por (18), con el propósito de encontrar la respuesta de alta frecuencia $\hat{F}_i$ dada por (19).

$$h_\sigma(y,x) = \delta(y-\mu, x-\mu) - G(y,x,\mu,\sigma) \tag{18}$$

$$\hat{F}_k(y,x) = |h_\sigma * I_k(y,x)| \tag{19}$$

donde la función impulso δ, representa una Gaussiana con media μ y desviación estándar $\sigma = 0$. Note que el kernel h_σ es una diferencia de Gaussianas

3. Algoritmo de Combinación Lineal de Imágenes CLI

De acuerdo con lo mencionado en las secciones anteriores, proponemos realizar el proceso de fusión de imágenes multi foco como una combinación lineal de dos imágenes I_1 e I_2, multiplicadas por un peso binario de acuerdo a lo planteado en (20).

$$I_F(y,x) = I_1(y,x)p(y,x) + I_2(y,x)(1-p(y,x)) \quad \forall < y,x > \in L \tag{20}$$

La formulación de (20) se basa en el hecho de que las imágenes que perdieron nitidez tienen partes de una supuesta imagen original I_0 retomando lo planteado en (4).

Si tenemos el valor de la matriz de pesos p_0 podemos calcular los valores de p que nos permitan determinar una Imagen fusionada I_F igual a la imagen original I_0. Para esto sustituimos en (20) las fórmulas de las imágenes I_1 e I_2 dadas por (4) y sustituimos $I_0(y,x)$ por $I_F(y,x)$ en (20). Al simplificar obtenemos un polinomio dado por (21)

$$(I_0(y,x) - J_0(y,x))\bar{p}(y,x) = 0 \tag{21}$$

donde $\bar{p}(y,x) = 2p_0(y,x)p(y,x) - p(y,x) - p_0(y,x) + 1$.

La solución para $p(y,x)$ es aquella que hace $\bar{p}(y,x) = 0$ dada por:

$$p(y,x) = \frac{p_0(y,x) - 1}{2p_0(y,x) - 1} \tag{22}$$

De acuerdo con esto podemos calcular p; así dado el valor de $p_0(y,x) = 0$ el valor de $p(y,x) = 1$ y si $p_0(y,x) = 1$ entonces $p(y,x) = 0$. Esto significa que si conocemos los valores iniciales p_0 los valores de p pueden calcularse de forma equivalente a (22) con (23), lo que demuestra que el proceso es perfectamente invertible.

$$p(y,x) = (1 - p_0(y,x)) \tag{23}$$

Por lo tanto proponemos una $p(y,x)$ binaria tal que la parte nítida presente en las imágenes I_1 e I_2 sea el resultado en la imagen fusionada I_F para todos los píxeles presentes en la retícula L. Así el valor $p(y,x)$, en el píxel con coordenadas (y,x), deberá ser:

$$p(y,x) = \begin{cases} 1 & \text{Si } I_1(y,x) \text{ es más nítido que } I_2(y,x) \\ 0 & \text{En otro caso} \end{cases} \tag{24}$$

Para calcular $p(y,x)$ debemos utilizar un filtro pasa altas que calcule los píxeles o regiones más nítidas para el par de imágenes dado. Para lograrlo proponemos utilizar la discretización del Laplaciano dado por (25). Para detalles del Laplaciano como estimador de la Respuesta de alta frecuencia ver (Gonzalez and Woods, 2008).

La respuesta de alta frecuencia F_k la calcularemos como el valor absoluto de la convolución de la imagen I_k con el kernel Laplaciano h (25) mediante (26).

$$h = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{25}$$

$$F_k(y,x) = |h * I_k(y,x)| \tag{26}$$

Dada la ecuación de fusión (20) y la Respuesta a la alta frecuencia (26), proponemos hacer el cálculo de la respuesta a la alta frecuencia F_F para la imagen fusionada como:

$$F_F(y,x) = |h * [I_1(y,x)p(y,x) + I_2(y,x)(1 - p(y,x))]| \tag{27}$$

considerando que $p(y,x)$ y $|h * I_k(y,x)|$ siempre son positivos, podemos reescribir el valor absoluto expresado en (27) como (28)

$$F_F(y,x) = |h * I_1(y,x)|p(y,x) + |h * I_2(y,x)|(1 - p(y,x)) \tag{28}$$

Con esto la respuesta a la alta frecuencia de la imagen fusionada $F_F(y,x)$, se puede expresar como la combinación lineal de las respuestas a alta la frecuencia F_k de cada una de las imágenes I_k, de acuerdo con (29).

$$\begin{gathered} F_F(y,x) = F_1(y,x)p(y,x) + F_2(y,x)(1 - p(y,x)) \\ F_F(y,x) = \Delta F(y,x)p(y,x) + F_2(y,x) \\ \forall (y,x) \in L \end{gathered} \tag{29}$$

donde $\Delta F(y,x) = F_1(y,x) - F_2(y,x)$.

Para la estimación de la frecuencia F_F dada por (29), si $p(y,x)$ es igual a 1 significa que el píxel en la coordenadas (y,x) de la imagen uno tiene mayor respuesta a la alta frecuencia que el píxel en las mismas coordenadas para la imagen dos. De lo contrario si es igual a cero la respuesta a la alta frecuencia será mayor en píxel de la imagen dos. Esto significa que el valor de $p(y,x)$ deberá ser aquel que maximiza la respuesta a la alta frecuencia de la imagen fusionada.

Con el objetivo de garantizar la máxima respuesta a la alta frecuencia en cada uno de los píxeles de la imagen fusionada, proponemos calcular la matriz de pesos p como aquella que maximiza la suma de las respuestas a la alta frecuencia $E(p)$ dada por (30), para todos los píxeles presentes en la retícula L de la imagen.

$$\begin{gathered} Max\ E(p) = \sum_{y=1}^{n_r} \sum_{x=1}^{n_c} \Delta F(y,x)p(y,x) \\ \text{s. a} \\ p(y,x) \in \{0,1\} \end{gathered} \tag{30}$$

Cabe mencionar que F_2 no aparece en (30) debido a que es un valor constante para el proceso de maximización. Adicionalmente al aplicar un proceso de maximización, la $p(y,x)$ binaria será remplazada por una $P(y,x)$ flotante, lo que significa que la restricción $p(y,x) \in \{0,1\}$ será remplazada por $0 \leq P(y,x) \leq 1$.

3.1. Coherencia espacial

En el caso de tener una imagen con poca textura, la pérdida de nitidez causada por el efecto de la profundidad de campo de la lente de la cámara, hará que píxeles vecinos correspondientes a la misma región no sean tomados de la misma imagen. La coherencia espacial, restringe a que píxeles vecinos, pertenecientes a una misma región tengan un valor de nitidez o desenfoque similar. Para lograr la coherencia espacial es indispensable una restricción que permita crear regiones coherentes o con un comportamiento similar. La restricción de coherencia espacial que proponemos restringe a que el valor absoluto de las diferencias de la matriz de pesos entre un píxel $P(y,x)$ y sus píxeles vecinos $P(y-1,x)$, $P(y,x-1)$ y $P(y-1,x-1)$ sea menor que un valor ϵ, de acuerdo con (31).

$$\begin{gathered} |P(y,x) - P(y-1,x)| < \epsilon \\ |P(y,x) - P(y,x-1)| < \epsilon \\ |P(y,x) - P(y-1,x-1)| < \epsilon \end{gathered} \tag{31}$$

La coherencia espacial, expresada en estas restricciones, favorece que los valores de P en píxeles vecinos tiendan a parecerse a medida que ϵ disminuye. Este conjunto de restricciones puede plantearse como un nuevo conjunto de restricciones lineales dadas por (32).

$$\begin{gathered} P(y,x) - P(y-1,x) < \epsilon \\ P(y-1,x) - P(y,x) < \epsilon \\ P(y,x) - P(y,x-1) < \epsilon \\ P(y,x-1) - P(y,x) < \epsilon \\ P(y,x) - P(y-1,x-1) < \epsilon \\ P(y-1,x-1) - P(y,x) < \epsilon \end{gathered} \tag{32}$$

3.2. Función objetivo

Si reunimos los elementos dados en las subsecciones anteriores, el problema de la fusión de imágenes multi foco se puede plantear como la maximización de una ecuación lineal con restricciones lineales dada por (33).

$$Max\ E(P) = \sum_{y=1}^{n_r} \sum_{x=1}^{n_c} \Delta F(y,x) P(y,x)$$

s. a

$$\begin{gathered} P(y,x) - P(y-1,x) < \epsilon \\ P(y-1,x) - P(y,x) < \epsilon \\ P(y,x) - P(y,x-1) < \epsilon \\ P(y,x-1) - P(y,x) < \epsilon \\ P(y,x) - P(y-1,x-1) < \epsilon \\ P(y-1,x-1) - P(y,x) < \epsilon \\ P(y,x) \leq 1 \\ P(y,x) \geq 0 \end{gathered} \tag{33}$$

$$\forall (y,x) \in \{1,\cdots,n_r\} \times \{1,\cdots,n_c\}$$

La solución puede llevarse a cabo mediante el método Simplex (Luenberger, 1973) y una vez calculada $P^\star$ (argumento máximo para (33)), sugerimos aplicar una binarización del arreglo de pesos para mejorar el desempeño de la fusión ya que los valores de $p(y,x)$ se definieron binarios. Esta binarización se lleva a cabo mediante (34).

$$p(y,x) = \begin{cases} 1 & \text{Si } P^\star(y,x) \geq 0.5 \\ 0 & \text{en caso contrario} \end{cases} \tag{34}$$

En resumen el Algoritmo 1 de Combinación Lineal de Imágenes (CLI) muestra el procedimiento de solución para hacer la fusión de imágenes multi foco de tamaño $n_r \times n_c$ maximizando (33).

Algoritmo 1 CLI(I_1, I_2, h)

1: Calcular la respuesta F_1 y F_2 aplicando (26)
2: Calcular $\Delta F = F_1 - F_2$
3: Calcular $P^\star$ maximizando (33) para P
4: Calcular p binarizando $P^\star$ con (34)
5: Calcular la imagen fusionada I_F aplicando (20)
6: **devolver** I_F

3.3. Análisis de Complejidad

La forma canónica para el método Simplex está dada por (35)

$$\begin{gathered} \text{Max } z = c_1 x_1 + c_2 x_2 + \cdots + c_N x_N \\ \text{s. a} \\ Ax + \mathcal{I} x_h = b \end{gathered} \tag{35}$$

donde $\mathcal{I}$ es una matriz identidad y x_h es el vector de variables de holgura para trasformar el problema de desigualdades en igualdades.

Si bien el problema puede ser resuelto, los requerimientos de memoria cuando se utilizan imágenes hacen casi imposible la solución utilizando el método Simplex. El total de variables N, es igual al número de renglones por el número de columnas de la imagen $N = n_r \times n_c$. La matriz de restricciones A tendrá M renglones y N columnas, donde M es el número de restricciones. Para cada una de las siete restricciones por píxel, de la ecuación (33) tendremos $M = 7n_r n_c$, así que la matriz A será de tamaño $N \times M = 7n_r^2 n_c^2$ y la matriz identidad será de tamaño $M \times M = 49 n_r^2 n_c^2$. En el caso de una imagen pequeña de 256×256 píxeles el total de memoria, suponiendo que cada elemento de la matriz se almacena en un flotante de 4 bytes será de 112 GB para la matriz de restricciones y 784 GB para la identidad de los cuales menos del 0.01 % son diferentes de cero. En este caso se requiere solucionar el problema utilizando matrices dispersas. Sin embargo el método Simplex tiene la desventaja de no mantener la dispersidad de la matriz. Otra alternativa de solución pueden ser los métodos de Punto Interior (Terlaky, 2013), con el manejo de matrices dispersas, ya que este método tiene un mejor desempeño que el método Simplex. Adicionalmente el método simplex puede en el peor caso visitar todos los vértices de la región factibilidad lo que significa complejidad $O(2^N)$ haciendolo poco viable para solucionar (33)
.

La solución para (33) se calculó utilizando la función NMaximize de Wolfram Mathematica para imágenes de hasta 512×512 píxeles de manera muy eficiente. El código desarrollado en Wolfram Mathematica se muestra en la Figura 3.

```
F1 = Abs[ImageData[ImageConvolve[I1, h]]];
F2 = Abs[ImageData[ImageConvolve[I2, h]]];
EP = Sum[Sum[Py,x (F1[[y, x]] - F2[[y, x]]), {x, 1, nc}], {y, 1, nr}];
var = Flatten[Table[Py,x, {y, 1, nr}, {x, 1, nc}]];
rest = Flatten[{Table[Py,x ≥ 0, {y, 1, nr}, {x, 1, nc}],
    Table[Py,x ≤ 1, {y, 1, nr}, {x, 1, nc}],
    Table[Py,x - Py,x-1 ≤ e, {y, 1, nr}, {x, 2, nc}],
    Table[Py,x-1 - Py,x ≤ e, {y, 1, nr}, {x, 2, nc}],
    Table[Py,x - Py-1,x-1 ≤ e, {y, 2, nr}, {x, 2, nc}],
    Table[Py-1,x-1 - Py,x ≤ e, {y, 2, nr}, {x, 2, nc}],
    Table[Py,x - Py-1,x ≤ e, {y, 2, nr}, {x, 1, nc}],
    Table[Py-1,x - Py,x ≤ e, {y, 2, nr}, {x, 1, nc}]}];
sol = NMaximize[{EP, rest}, var];
PStar = Table[Py,x /. Last[sol], {y, 1, nr}, {x, 1, nc}];
p = Binarizar[PStar];
```

Figura 3: Solución de (33) utilizando Wolfram Mathematica

4. Solución por ventanas, Algoritmo CLI-V

Dada la complejidad para calcular la matriz de pesos P sobre todos los píxeles de la imagen, decidimos dividir la imagen en un conjunto de ventanas de tamaño $w \times w$ y calcular para cada ventana una submatriz de pesos $p_{y,x}$ con origen en las coordenadas (y,x). La relación que guardan los valores originales de la matriz de pesos P con la submatriz de pesos $p_{y,x}$ pueden calcularse mediante (36).

$$P(y+k, x+l) = p_{y,x}(k,l)$$
$$\forall (k,l) \in \{0, \cdots, w-1\} \times \{0, \cdots, w-1\} \tag{36}$$

El procedimiento de optimización dado por (33) pude ser reformulado sobre una ventana de tamaño $w \times w$ y planteado como una nueva función a la que denominamos E_w dada por (37):

$$Max\ E_w(p_{y,x}) = \sum_{k=0}^{w-1} \sum_{l=0}^{w-1} \Delta F(y+k, x+l) p_{y,x}(k,l)$$

s. a

$$\begin{gathered} p_{y,x}(k,l) - p_{y,x}(k-1,l) < \epsilon \\ p_{y,x}(k-1,l) - p_{y,x}(k,l) < \epsilon \\ p_{y,x}(k,l) - p_{y,x}(k,l-1) < \epsilon \\ p_{y,x}(k,l-1) - p_{y,x}(k,l) < \epsilon \\ p_{y,x}(k-1,l-1) - p_{y,x}(k,l) < \epsilon \\ p_{y,x}(k,l) - p_{y,x}(k-1,l-1) < \epsilon \\ p_{y,x}(k,l) \leq 1 \\ p_{y,x}(k,l) \geq 0 \end{gathered} \tag{37}$$

$$\forall (k,l) \in \{0, \cdots, w-1\} \times \{0, \cdots, w-1\}$$

En general podemos considerar que la ventana tendrá un tamaño $1 \leq w \leq n_r$ y $1 \leq w \leq n_c$, en el caso de $w = n_r$ y $w = n_c$ se realiza la optimización sobre una ventana que abarca todos los píxeles de la imagen y en caso de que $w = 1$ entonces la optimización se realiza de manera individual para cada píxel. Podemos maximizar E_w para tantas submatrices $p_{y,x}$ como píxeles hay en la imagen, por lo cual proponemos calcular la matriz de pesos P para todos los píxeles en la imagen como el promedio de las soluciones de acuerdo con (38).

$$P(y,x) = \frac{q(y,x)}{N_w(y,x)}$$

$$q(y,x) = \sum_{k=1}^{w} \sum_{l=1}^{w} p_{y-k,x-l}(k,l) \tag{38}$$

$$N_w(y,x) = \sum_{k=1}^{w} \sum_{l=1}^{w} 1$$
$$\forall (y-k, x-l) \in L$$

donde $N_w(y,x)$ es el número de ventanas donde participa el píxel (y,x). En el caso de que la imagen se divida en un número de ventanas de tamaño $w \times w$ sin traslape, el valor $N_w(y,x)$ será uno.

El Algoritmo 2, que denominamos Combinación Lineal de Imágenes por Ventanas (CLI-V), muestra el procedimiento para el cálculo de pesos sobre todos los píxeles de la imagen con soluciones parciales sobre ventanas de tamaño $w \times w$ con desplazamiento Δ. Para maximizar E_w se utilizó el método Simplex, ya que la solución se llevó a cabo sobre ventanas de tamaño muy inferior a las dimensiones de la imagen.

Algoritmo 2 CLI-V(I_1, I_2 , ϵ, w, Δ)

Calcular F_1 y F_2 con (26)
Calcular $\Delta F = F_1 - F_2$
$q(y,x) \leftarrow 0$, $N_w(y,x) \leftarrow 0$, $\forall (y,x) \in L$
$y \leftarrow 1$
mientras $y \leq n_r - w$ **hacer**
  $x \leftarrow 1$
  mientras $x \leq n_c - w$ **hacer**
    Calcular $p^{\star}_{y,x} = \underset{p_{y,x}}{\text{argmax}}\ E_w(p_{y,x})$ dada por (37)
    para $k = 0$ **hasta** w **hacer**
      para $l = 0$ **hasta** w **hacer**
        $q(y+k, x+l) \leftarrow q(y+k, x+l) + p^{\star}_{y,x}(k,l)$
        $N_w(y+k, x+l) \leftarrow N_w(y+k, x+l) + 1$
      fin para
    fin para
    $x \leftarrow x + \Delta$
  fin mientras
  $y \leftarrow y + \Delta$
fin mientras
$P(y,x) = q(y,x)/N_w(y,x) \qquad \forall (y,x) \in L$
Calcular p binarizando P (34)
Construir I_F de acuerdo con (20)
devolver I_F

5. Algoritmo Simplificado de Optimización (CLI-S)

Para las restricciones de coherencia espacial dadas por (32) si consideramos que el valor de ϵ tiende a cero los valores de $p_{y,x}$ tenderán a un valor constante al que denominaremos $q_{y,x}$. Bajo esta consideración podemos replantear las restricciones de coherencia espacial como (39).

$$\begin{gathered} q_{y,x} \equiv p_{y,x}(k,l) \approx p_{y,x}(k-1,l) \\ q_{y,x} \equiv p_{y,x}(k,l) \approx p_{y,x}(k,l-1) \\ q_{y,x} \equiv p_{y,x}(k,l) \approx p_{y,x}(k-1,l-1) \end{gathered} \tag{39}$$

Por lo que maximizar $E_w(p_{y,x})$ pude reformularse asumiendo que el peso en cada valor $p_{y,x}(k,l)$ es constante para todos los valores en la ventana. Estas nuevas condiciones no violan las restricciones de coherencia espacial, por lo cual (37) pude simplificarse y expresarse por (40).

$$\begin{gathered} Max\ E_s(q_{y,x}) = q_{y,x} S(y,x) \\ \text{s.a} \\ 0 \leq q_{y,x} \leq 1 \end{gathered} \tag{40}$$

donde $S(y,x)$ se calcula sobre una ventana de tamaño $w \times w$, centrada en las coordenadas (y,x) y está dada por (41).

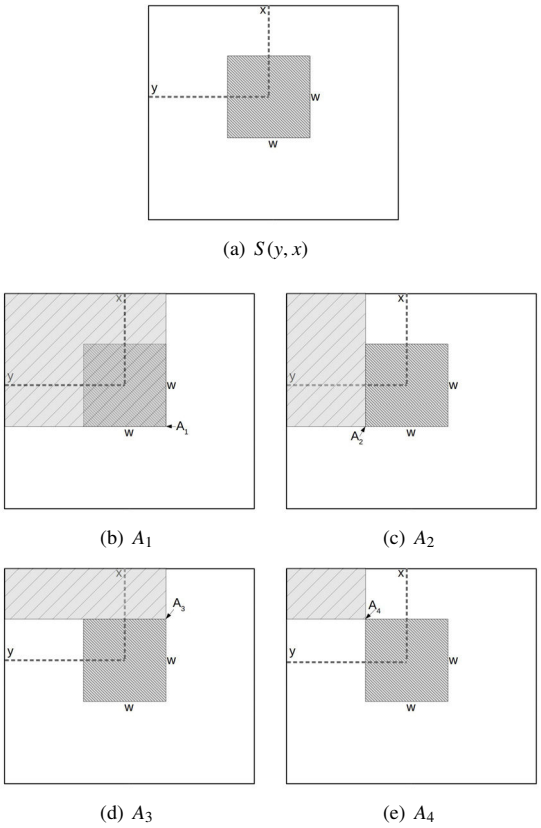
$$S(y,x) = \sum_{k=-w/2}^{w/2} \sum_{l=-w/2}^{w/2} \Delta F(y+k, x+l) \tag{41}$$

Para resolver este problema no será necesario aplicar el método Simplex o algún procedimiento de maximización, note que si el valor de $S(y,x)$ es positivo el valor que maximiza la función $E_s(q_{y,x})$ será $q_{y,x} = 1$; de lo contrario será $q_{y,x} = 0$. De

acuerdo con esto, solamente será necesario calcular el signo de la suma $S(y, x)$ para cada uno de los píxeles.

Una manera eficiente para calcular $S(y, x)$ es utilizar imágenes integrales o también conocidas como imágenes incrementales, procedimiento propuesto por Viola y Jones (Viola and Jones, 2001). Las imágenes incrementales son utilizadas en procesamiento digital de imágenes para hacer la convolución de una imagen con un kernel. La complejidad para el calculo de la imagen incremental es de orden $O(N)$, donde N es el número de píxeles. La imagen incremental está definida por (42).

$$\Delta F_{inc}(y, x) = \sum_{k=1}^{y} \sum_{l=1}^{x} \Delta F(k, l) \tag{42}$$



(a) $S(y, x)$

(b) A_1 (c) A_2

(d) A_3 (e) A_4

Figura 4: Calculo $S(y, x)$ en una ventana utilizando imágenes incrementales.

Una vez calculada la imagen incremental ΔF_{inc}, para calcular la suma $S(y, x)$ sobre una ventana de tamaño $w \times w$ con centro en la posición (y, x) tal como se muestra en la Figura 4(a), se deberán obtener cuatro valores de la imagen incremental. Cada uno de estos valores corresponden a la sumatoria dentro de una región en la imagen original ΔF de acuerdo con (42), los cuales son: $A_1 = \Delta F_{inc}(y + w/2, x + w/2)$, $A_2 = \Delta F_{inc}(y+w/2, x-w/2-1)$, $A_3 = \Delta F_{inc}(y-w/2-1, x+w/2)$ y $A_4 = \Delta F_{inc}(y-w/2-1, x-w/2-1)$. Esta condición se ilustra con las Figuras 4(b), 4(c), 4(d) y 4(e) y vistos como áreas sobre la imagen original ΔF, podemos decir que el valor de $S(y, x)$ se puede calcular como $A_1 - A_2 - A_3 + A_4$. Calculado de esta forma el valor de $S(x, y)$ (43) la complejidad será $O(N)$ independientemente del tamaño de la ventana.

$$\begin{aligned} S(y, x) = &\Delta F_{inc}(y + w/2, x + w/2) \\ &-\Delta F_{inc}(y + w/2, x - w/2 - 1) \\ &-\Delta F_{inc}(y - w/2 - 1, x + w/2) \\ &+\Delta F_{inc}(y - w/2 - 1, x - w/2 - 1) \end{aligned} \tag{43}$$

Note que la suma $S(y, x)$ sigue un rol muy parecido a lo implementado por Cao *et al.* (Cao et al., 2015) en la forma en que calcula sus pesos $W(i, j)$ y utiliza el filtro de mayoría, de acuerdo con (13) y (14), respectivamente. Sin embargo, el método que planteamos determina el signo de la suma $\Delta F(y, x)$ mientras Cao solamente toma el signo. La implementación de Cao *et al.* tendrá tiempos mucho mayores ya que al menos debe calcular la DCT la cual tiene complejidad $O(Nlog(N))$.

El Algoritmo 3, al que denominamos Combinación Lineal de Imágenes Simple (CLI-S), muestra el procedimiento para implementar la solución de Optimización Simplificada dada por (40). En este algoritmo, la imagen incremental se calcula una sola vez y los valores por ventana corresponden a una simple suma. La solución para p ya es binaria, por lo cual no es necesario el procedimiento de binarización, como en el caso de los algoritmos CLI y CLI-V.

Algoritmo 3 CLI-S(I_1, I_2, w)

Calcular F_1 y F_2 con (26)
Calcular $\Delta F = F_1 - F_2$
Calcular ΔF_{inc} con (42)
para $y = 1$ **hasta** n_r **hacer**
  para $x = 1$ **hasta** n_c **hacer**
    Calcular $S(y, x)$ con (43)
    si $S(y, x) \geq 0$ **entonces**
      $p(y, x) = 1$
    si no
      $p(y, x) = 0$
    fin si
  fin para
fin para
Construir I_F utilizando (20)
devolver I_F

6. Resultados

En esta sección se presentan los resultados de aplicar los tres algoritmos sobre un conjunto de 5 pares de imágenes. Uno de los experimentos se realizó sobre un par sintético de imágenes y los cuatro restantes sobre pares de imágenes reales utilizados, en otras publicaciones, para mostrar el desempeño de algoritmos de fusión de imágenes multi foco. Los detalles de estos experimentos se explican en las siguientes subsecciones.

6.1. Experimento con imágenes sintéticas

Para este experimento se utilizó un par de imágenes sintéticas de tamaño 512×512, correspondientes a un par de lobos. Estas imágenes, Figuras 2(c) y 2(d), fueron creadas a partir de

(4). Usando la matriz de pesos p_0 (Figura 2(b)), con la cual se calcularon las imágenes sintéticas, podemos determinar el porcentaje de aciertos de la matriz de pesos estimada con cada uno de los tres algoritmos. Cabe recordar que la relación entre p_0 y p es inversa y está dada por (23).

En la Tabla 1 se muestran los resultados de los tres algoritmos aplicados a las imágenes de los lobos con los parámetros que dieron el porcentaje más alto de aciertos. Para esta tabla, la primer columna corresponde al nombre del algoritmo, la segunda a los parámetros utilizadas en la implementación, la tercera al tiempo en segundos y la cuarta al porcentaje de acierto entre la matriz de pesos original y la calculada. Los parámetros se seleccionaron manualmente hasta obtener una solución lo suficientemente buena para cada uno de los algoritmos. Podemos notar, que el porcentaje de aciertos es muy similar para los tres algoritmos, sin embargo el Algoritmo CLI-S tiene los resultados en centésimas de segundo cuando los otros dos se llevan a cabo en minutos. Cabe señalar que en el caso del algoritmo CLI-S el tiempo es constante a pesar de cambiar el tamaño de la ventana dado que se implementaron las operaciones utilizando imágenes incrementales. En la Figura 5 se muestran los resultados de la matriz de pesos calculada por los tres algoritmos usando los parámetros de la Tabla 1. Note que se tienen matrices de pesos muy similares en los tres casos, pero el algoritmo CLI-S es miles de veces más rápido.

(a) CLI (b) CLI-V (c) CLI-S

Figura 5: Comparativo de las matrices de pesos p calculada con los tres Algoritmos para las imágenes de los lobos

Tabla 1: Prueba de los tres algoritmos para la imagen de los lobos

Alg.	Parámetros	tiempo s.	% de aciertos
CLI	$\epsilon = 0.03$	5641.36	98.38
CLI-V	$\epsilon = 0.01$ $w = 9$ $\Delta = 3$	1271.99	97.10
CLI- S	$w = 31$	0.16	98.39

6.2. *Comportamiento de la coherencia espacial*

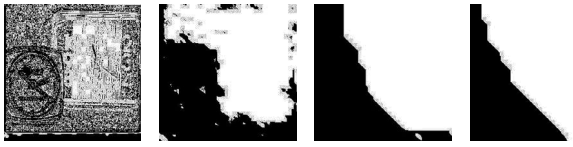
Para este experimento se utilizó un par de imágenes ampliamente utilizados en la literatura, las cuales corresponden a dos relojes que se muestran en la Figura 6. Intentamos explicar mediante los resultados para estas imágenes, como se comportan los parámetros para cada uno de los tres algoritmos. En general en los tres algoritmos los parámetros controlan la coherencia espacial a través de ϵ y w. Además de estos parámetros, para el algoritmo CLI-V se utilizó el parámetro $\Delta = w$, con el objetivo simplificar la explicación de los parámetros de coherencia

(a) Reloj 1 (b) Reloj 2

Figura 6: Imágenes de los relojes

espacial, ya que así no existe traslape entre ventanas. En los tres algoritmos se utilizaron los mismos valores de ϵ y w para clarificar el comportamiento de la coherencia. Los valores de coherencia espacial ϵ utilizados son 1, 0.1, 0.01 y 0.001, y un tamaño de ventana de w igual a 1, 3, 5 y 7.

En la Figura 7 se muestra la matriz p obtenida con el Algoritmo CLI cuando se hacen cambios en el parámetro ϵ. Para cada una de las figuras en la parte de abajo se muestra el tiempo en segundos de cada corrida y el valor de ϵ. Podemos notar que la mejor solución, de acuerdo con esta Figura, estará entre 0.1 y 0.01 con un tiempo de ejecución alrededor de 300 s.



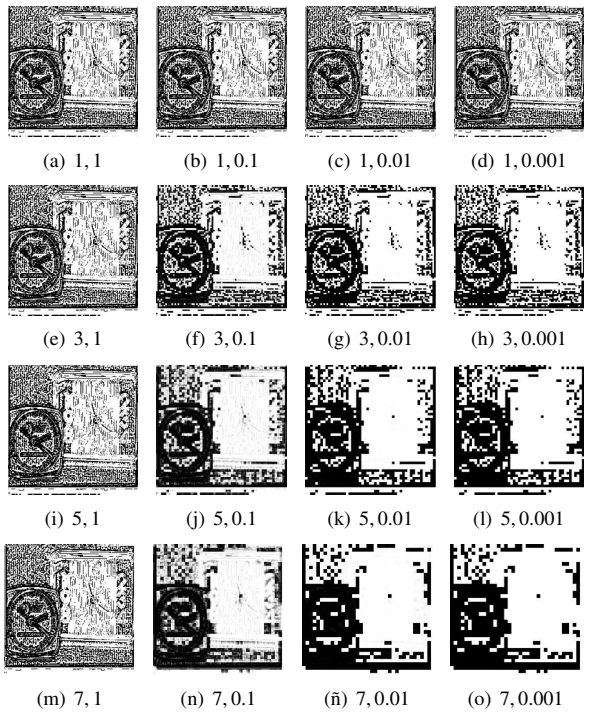
(a) 129.730 s. con $\epsilon = 1$, (b) 299.06 s. con $\epsilon = 0.1$ (c) 472.795 s. con $\epsilon = 0.01$ (d) 436.528 s. con $\epsilon = 0.001$

Figura 7: Matriz de pesos encontrada con el Algoritmo CLI

Para el caso del algoritmo CLI-V los resultados se muestran en la Figura 8 al variar los parámetros w y ϵ. En esta Figura, el primer renglón corresponde a la solución con $w = 1$, el segundo a $w = 3$ y así sucesivamente para $w = 5$ y $w = 7$ variando ϵ entre 1 y 0.001 para cada tamaño de ventana.

En general, podemos notar que al aumentar el tamaño de la ventana w y/o reducir el valor de la restricción de coherencia espacial ϵ obtenemos regiones menos granuladas. Así este algoritmo tiene la capacidad de mejorar la coherencia espacial de dos formas. En particular los parámetros de coherencia espacial con los que se obtuvieron los mejores resultados están por debajo de 0.1. Esto justifica la hipótesis que dio lugar al Algoritmo CLI-S ya que cuando ϵ tiende a cero tendremos regiones con poca granularidad.

La Figura 9 muestra los resultados para el Algoritmo CLI-S al variar el tamaño de la ventana w para valores iguales a 1, 3, 5 y 7. Los tiempos de ejecución para el algoritmo CLI-S, utilizando las imágenes del reloj en tamaño de 256×256 fue de 0.048 s., en promedio. Las parejas de valores, bajo cada una de las imágenes son los parámetros w y ϵ utilizados.



(a) 1, 1 (b) 1, 0.1 (c) 1, 0.01 (d) 1, 0.001

(e) 3, 1 (f) 3, 0.1 (g) 3, 0.01 (h) 3, 0.001

(i) 5, 1 (j) 5, 0.1 (k) 5, 0.01 (l) 5, 0.001

(m) 7, 1 (n) 7, 0.1 (ñ) 7, 0.01 (o) 7, 0.001

Figura 8: Matrices de pesos encontradas con el Algoritmo CLI-V.

(a) $w = 1$ (b) $w = 3$ (c) $w = 5$ (d) $w = 7$

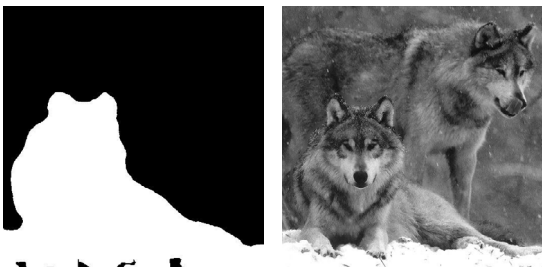
Figura 9: Matriz de pesos encontrada con el Algoritmo CLI-S.

6.3. *Imágenes fusionadas obtenidas con el Algoritmo CLI-S*

En esta subsección presentamos los resultados para los cinco pares de imágenes, con el objetivo de que el lector tenga una idea clara de las mejores soluciones encontradas y pueda comparar cualitativamente en el caso de las imágenes reales. En el caso de la imágenes reales, no hacemos una comparación cuantitativa, ya que hasta donde conocemos el estado del arte, no existe un enfoque que busque y/o reporte una función binaria de pesos. En todos los experimentos presentados en esta sección se utilizaron valores diferentes de tamaño de ventana los cuales fueron seleccionados manualmente, hasta obtener una calidad aceptable. Suponemos que la variación en el parámetro w puede ser explicada en base al nivel de detalle que existe entre los bordes de las diferentes regiones de las imágenes dadas.

En el caso de las imágenes sintéticas, correspondientes a los lobos, presentadas en la Figuras 2(c) y 2(d), la solución para la matriz de pesos y la imagen fusionada se presentan en las Figuras 10(a) y 10(b), respectivamente, para una ancho de ventana de $w = 31$.

Para el caso de la imágenes de los relojes, en la Figura 11 se muestra la solución obtenida, a la izquierda podemos ver la matriz de pesos p calculada para un valor de $w = 37$ y a la derecha la imagen fusionada. El tiempo para calcular la solución fue de 0.048 segundos para imágenes de tamaño 256×256 .



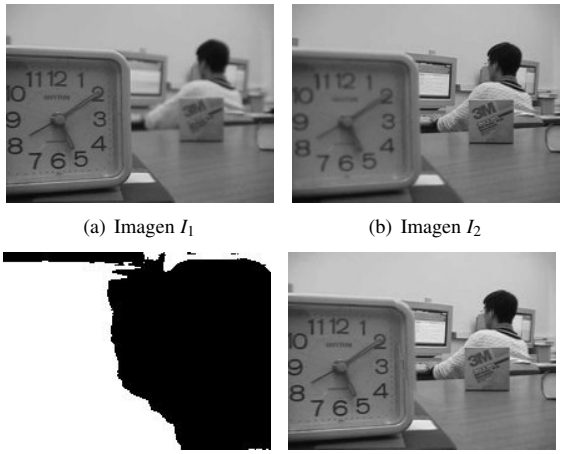
(a) matriz de pesos calculada (b) imagen fusionada

Figura 10: Resultados para las imágenes del lobo

(a) matriz de pesos calculada (b) imagen fusionada

Figura 11: Resultados para la imagen de los dos relojes

En la Figura 12 se muestra la solución para una imagen en la que aparece un reloj y una persona al fondo trabajando en una computadora. En la Figura 12(a) se muestra la imagen enfocada en el reloj y en la Figura 12(b) enfocada en la persona. La matriz de pesos y la imagen fusionada se muestran en las figuras 12(c) y 12(d) respectivamente. El resultado para este par de imágenes de 159×215 píxeles, es lo suficientemente bueno para una ventana de $w = 51$, con un tiempo de solución de 0.032 segundos.



(a) Imagen I_1 (b) Imagen I_2

(c) matriz de pesos calculada p (d) imagen fusionada I_F

Figura 12: Solución para las imágenes de reloj

Otro par de imágenes utilizadas en la literatura se presentan en la Figura 13. Las Figuras 13(a) y 13(b) muestran una lata de refresco y una caja con un código de barras donde en la primera se enfocó en la lata y en la segunda en la caja. La matriz de pesos se presenta en la Figura 13(c) y la imagen fusionada en la 13(d). Para este ejemplo el tiempo de solución fue de 0.091 segundos para imágenes de tamaño 571×571 y con $w = 141$.

(a) Imagen $I1$ (b) Imagen I_2

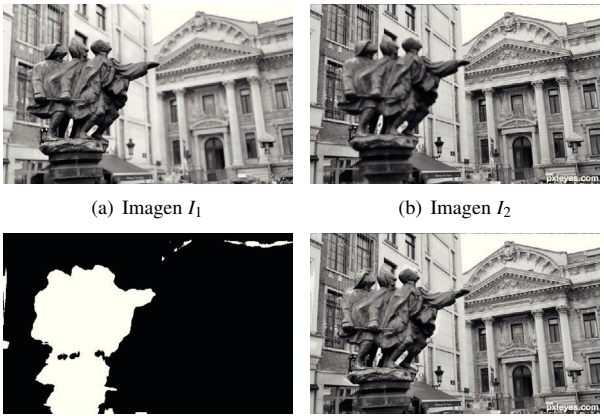
(c) matriz de pesos calculada p (d) imagen fusionada I_F

Figura 13: Solución para las imágenes de caja y refresco

Finalmente tomamos un par de imágenes de la red donde se muestra la imagen de una escultura. En la Figura 14(a) se muestra la imagen enfocada sobre la escultura de tres personajes y en la Figura14(b) la imagen está enfocada en el edificio del fondo. Las imágenes de las Figuras 14(c) y 14(d) muestran la matriz de pesos y la imagen fusionada, respectivamente. Escogimos este par de imágenes por el detalle obtenido en la matriz de pesos calculada ya que ésta se aproximan muy bien al contorno de la escultura. La solución se calculó en un tiempo de 0.078 segundos para una tamaño de imagen de 373×600 y $w = 21$.

7. Conclusiones

En este trabajo se presentaron tres algoritmos para efectuar la fusión de imágenes multi foco. De los resultados obtenidos con el par sintético debemos hacer notar que la exactitud alcanzada fue del 98.39 %. Para los casos de las imágenes reales no existen soluciones similares a la que planteamos y menos una matriz de pesos como la calculada. Sin embargo, podemos ver de manera cualitativa que se obtiene una matriz de pesos que se apega a los bordes de los objetos. Consideramos que en los cuatro casos reales presentados tenemos resultados cualitativamente buenos comparados con los resultados reportados en el estado del arte para el mismo conjunto de imágenes. El tiempo de ejecución del algoritmo CLI-S para la fusión de imágenes



(a) Imagen I_1 (b) Imagen I_2

(c) matriz de pesos calculada p (d) imagen fusionada I_F

Figura 14: Solución para las imágenes de la escultura

fue del orden de centésimas de segundo en todos los ejemplos mostrados, lo que nos permite plantear éste algoritmo como un algoritmo que permite la fusión de imágenes en tiempo real, más aún dado que el tiempo consumido por el algoritmo es de orden lineal, podemos decir que el tiempo necesario para realizar la fusión será linealmente proporcional al tamaño de la imagen, por lo que el algoritmo puede funcionar de manera adecuada tanto para imágenes pequeñas como para imágenes de alta resolución. A diferencia de muchos algoritmos planteados en la literatura en los que se realizan procesos de orden cuadrático o exponencial que hacen poco práctica su aplicación con imágenes de alta resolución.

En la redacción de éste artículo mencionamos que la selección del tamaño de ventana se hizo de forma manual hasta encontrar la mejor solución, dicha decisión es importante y una mala decisión podría ocasionar un resultado indeseado. Actualmente estamos trabajando en automatizar la selección del tamaño de ventana para evitar esa situación.

Dada la velocidad del algoritmo CLI-S se puede generalizar su uso para fusionar conjuntos con más de 2 imágenes, para ello se extraen dos imágenes del conjunto y se remplazan por la imagen fusionada resultante. El procedimiento se repite hasta que sólo se tenga una imagen, la cual será el resultado de la fusión de todas las imágenes. No presentamos la fusión considerando más de dos imágenes dado que no es común encontrarlas en la literatura y que actualmente trabajamos en esa dirección.

English Summary

Multi Focus Image Fusion based on Linear Combination of Images using Incremental Images

Abstract

This article presents three algorithms to determinate multifocus image fusion. These algorithms are based on a linear combination of two images with different focus distances. The three algorithms maximize a linear function with spatial coherence constrains. We present these algorithms in sequence to

show how we devised a fast and simple algorithm. The first algorithm, CLI (for its acronym in spanish Combinación Lineal de Imágenes) was implemented using Wolfram Mathematica, but given the number of variables to optimize, the solution takes a lot of computing time. The second algorithm, CLI-V (for its acronym in spanish Combinación Lineal de Imágenes por Ventanas) is an application of algorithm CLI on image regions to improve the time performance and being able to implement it through the Simplex method. The third algorithm, CLI-S (for its acronym in spanish Combinación Lineal de Imágenes Simple), is a simplification on CLI-V. This last algorithm is much faster exhibiting results of similar quality to the previous two, with a performance comparable to the results presented in the state of the art. CLI-S was implemented using the concept of integral images. This fact allows the algorithm to produce results in hundredth of a second for the test images analized. The results of the three algorithms are compared using one set of synthetic and four sets of real images. The real images are commonly used by the state of the art proposal; they were so that the reader can make a qualitative comparison of results. The synthetic images are reconstructed with 98 % accuracy in 0.080 s. and the image size is 512×512, this situation allows us to say that CLI-S can be used as a real-time algorithm of multifocus image fusion and we have not found a similar proposal in the state of art.

Keywords: Linear Programming, muitifocus images fusion, high pass filters, integral images

Referencias

Alonso, J. R., Fernández, A., Ayubi, G. A., Ferrari, J. A., Apr 2015. All-in-focus image reconstruction under severe defocus. Opt. Lett. 40 (8), 1671–1674.

Bae, S., Durand, F., 2007. Defocus magnification. Computer Graphics Forum 26 (3), 571–579.

Burt, P., Adelson, E., Apr 1983. The laplacian pyramid as a compact image code. Communications, IEEE Transactions on 31 (4), 532–540.

Burt, P., Kolczynski, R., May 1993. Enhanced image capture through fusion. In: Computer Vision, 1993. Proceedings., Fourth International Conference on. pp. 173–182.

Cao, L., Jin, L., Tao, H., Li, G., Zhuang, Z., Zhang, Y., Feb 2015. Multi-focus image fusion based on spatial frequency in discrete cosine transform domain. Signal Processing Letters, IEEE 22 (2), 220–224.

Chai, Y., Li, H., Guo, M., 2011. Multifocus image fusion scheme based on features of multiscale products and {PCNN} in lifting stationary wavelet domain. Optics Communications 284 (5), 1146 – 1158.

Elder, J., Zucker, S., Jul 1998. Local scale control for edge detection and blur estimation. Pattern Analysis and Machine Intelligence, IEEE Transactions on 20 (7), 699–716.

Gonzalez, R. C., Woods, R. E., 2008. Digital image processing. Prentice Hall, Upper Saddle River, N.J.

Kuthirummal, S., Nagahara, H., Zhou, C., Nayar, S., Jan 2011. Flexible depth of field photography. Pattern Analysis and Machine Intelligence, IEEE Transactions on 33 (1), 58–71.

Li, S., Kwok, J. T., Wang, Y., 2001. Combination of images with diverse focuses using the spatial frequency. Information Fusion 2 (3), 169 – 176.

Li, S., Kwok, J. T., Wang, Y., 2002. Multifocus image fusion using artificial neural networks. Pattern Recognition Letters 23 (8), 985 – 997.

Li, S., Yang, B., 2008. Multifocus image fusion using region segmentation and spatial frequency. Image and Vision Computing 26 (7), 971 – 979.

Luenberger, D., 1973. Introduction to Linear and Nonlinear Programming. Addison-Wesley Publishing Company.

Orozco, R. I., 2013. Fusión de imágenes multifoco por medio de filtrado de regiones de alta y baja frecuencia. Master's thesis, División de Estudios de Postgrado. Facultad de Ingeniería Eléctrica. UMSNH, Morelia Michoacan Mexico.

Pagidimarry, M., Babu, K. A., 2011. An all approach for multi-focus image fusion using neural network. Artificial Intelligent Systems and Machine Learning 3 (12), 732–739.

Pajares, G., de la Cruz, J. M., 2004. A wavelet-based image fusion tutorial. Pattern Recognition 37 (9), 1855 – 1872.

Redondo, R., Šroubek, F., Fischer, S., Cristóbal, G., 2009. Multifocus image fusion using the log-gabor transform and a multisize windows technique. Information Fusion 10 (2), 163 – 171.

Riaz, M., Park, S., Ahmad, M., Rasheed, W., Park, J., 2008. Generalized laplacian as focus measure. In: Bubak, M., van Albada, G., Dongarra, J., Sloot, P. (Eds.), Computational Science ? ICCS 2008. Vol. 5101 of Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 1013–1021.

Rivera, M., Ocegueda, O., Marroquin, J., Dec 2007. Entropy-controlled quadratic markov measure field models for efficient image segmentation. Image Processing, IEEE Transactions on 16 (12), 3047–3057.

Terlaky, T., 2013. Interior point methods of mathematical programming. Vol. 5. Springer Science & Business Media.

Viola, P., Jones, M., 2001. Rapid object detection using a boosted cascade of simple features. In: Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on. Vol. 1. pp. I–511–I–518 vol.1.

Wiener, N., 1964. Extrapolation, interpolation, and smoothing of stationary time series : with engineering applications. M.I. T. paperback series. Cambridge, Mass. Technology Press of the Massachusetts Institute of Technology, first published during the war as a classified report to Section D2, National Defense Research Committee.

Zhang, Q., long Guo, B., 2009. Multifocus image fusion using the nonsubsampled contourlet transform. Signal Processing 89 (7), 1334 – 1346.