Un algoritmo de restauración de imágenes digitales basado en medidas de contraste y técnicas de agrupamiento

D. Paternain, A. Jurio, I. Sarobe, C. Marco-Detchart, and H. Bustince

Departamento de Automática y Computación and Institute of Smart Cities, Universidad Pública de Navarra, Campus Arrosadia sn, 31106 Pamplona, daniel.paternain@unavarra.es

Resumen En este trabajo presentamos un nuevo algoritmo de restauración de imágenes digitales (*inpainting*) basado en algoritmos de agrupamiento (*clustering*) y medidas de contraste. El algoritmo propuesto comienza con la clasificación, mediante el algoritmo k-medias, de todas las subventanas (patrones) de una imagen utilizando sus características de contraste. En segundo lugar, el algoritmo restaura la intensidad de los píxeles desconocidos buscando el patrón más parecido dentro de una única clase, aquella que contiene las subventanas más parecidas a la subventana del píxel a restaurar. El uso de las técnicas de clustering y de las medidas de contraste permite reducir el número de comparaciones y obtener imágenes restauradas de mejor calidad.

Keywords: Inpainting, clustering, medidas de contraste, contraste local

1. Introducción

El *inpainting* o restauración de imágenes digitales consiste en modificar una imagen digital de forma que el observador que no conozca la imagen original no lo consiga detectar [1,5,8]. Un ejemplo clásico de inpainting es la restauración de imágenes que se han deteriorado a lo largo del tiempo, como imágenes antiguas con fallos de revelado. En este caso, el objetivo del *inpainting* es averiguar la intensidad original de los píxeles deteriorados. Otro ejemplo de aplicación bien diferente es la eliminación de objetos indeseados de una imagen. En este caso, el objetivo del inpainting es predecir el valor de intensidad que dichos píxeles tendrían si el objeto no estuviese en la imagen. Es decir, averiguar la intensidad de las zonas ocultas por el objeto.

Las técnicas de *inpainting* se dividen principalmente en dos clases. La primera clase engloba a los algoritmos basados en la interpolación de los píxeles circundantes conocidos (ver, por ejemplo, [7,8]). Estos algoritmos son computacionalmente sencillos, pero solo son útiles en imágenes donde las zonas a restaurar son pequeñas y homogéneas en términos de intensidad, debido a que producen un efecto de suavizado (*blurring*) y no son capaces de recuperar las texturas de la zona a restaurar. La segunda clase de algoritmos se basa en el ánalisis de texturas

para restaurar zonas suficientemente grandes dando sensación de continuidad en los objetos. Un ejemplo de algoritmo bien conocido es el presentado en [5], que se basa en buscar en la zona conocida de la imagen patrones (o subventanas) que sean muy parecidos a la zona que rodea al píxel a restaurar. Este procedimiento es muy similar al de los algoritmos de filtrado no local ([3]).

Los algoritmos basados en búsqueda de patrones presentan dos problemas. El primero es el elevado coste computacional. Esto es debido a que el número de comparaciones es muy grande. De hecho, para restaurar el valor de un píxel debemos comparar la subventana alrededor de dicho píxel con todas las subventanas (del mismo tamaño) disponibles en la zona conocida de la imagen. El segundo problema es cómo establecer la medida de comparación entre dos subventanas de la imagen. Si únicamente tenemos en cuenta la intensidad de las subventanas podríamos llegar a perder la continuidad en la forma de los objetos.

Para mejorar las técnicas de *inpainting* basadas en búsqueda de patrones, en este trabajo proponemos un nuevo algoritmo de restauración de imágenes digitales que realiza una clasificación previa de las subventanas disponibles en la zona conocida de la imagen. De esta manera, agrupamos en una clase (o *cluster*) todas las subventanas que tengan una distribución de intensidades lo más parecida posible. La clasificación la realizamos mediante el algoritmo de agrupamiento (o *clustering*) de las k-medias (k-*means*), utilizando las medidas de contraste (ver [2]) para analizar la distribución de las intensidades de cada subventana. La fase de restauración de cada píxel se divide en dos pasos: en el primero analizamos la distribución de intensidades del entorno conocido del píxel y lo comparamos con el representante (centroide) de cada clase. Una vez encontrada la clase que contiene las ventanas más parecidas (en términos de distribución de intensidades), buscamos el patrón más parecido únicamente en las subventanas asiganadas a dicha clase.

Las ventajas de nuesto algoritmo son dos. En primer lugar, somos capaces de reducir el coste computacional debido a la significativa reducción en el número de comparaciones. En lugar de comparar con todas las subventanas disponibles, lo hacemos únicamente con aquellas que son morfológicamente más parecidas. Además, al clasificar las zonas conocidas utilizando la distribución de intensidades (característica de contraste) hacemos que los píxeles restaurados mantengan mejor la continuidad y forma de las zonas restauradas.

Este trabajo se divide en las siguientes secciones: en la sección 2 recordamos los conceptos básicos en los que se basa nuestro algoritmo: *inpainting*, medidas de contraste y *clustering*. En la sección 3 presentamos el algoritmo de *inpainting* propuesto. En la sección 4 mostramos algunos resultados experimentales y finalizamos con conclusiones y líneas futuras en la sección 5.

2. Preliminares

En este trabajo denotamos una imagen en escala de grises de $M \times N$ píxeles como una aplicación $Q : \{1, \ldots, M\} \times \{1, \ldots, N\} \to \{0, \ldots, L-1\}$, siendo L el número de niveles de la escala de grises de la imagen (generalmente $L = 256$).

2.1. *Inpaiting*

Las técnicas de restauración digital o *inpainting* consisten en restaurar zonas dañadas o degradas de una imagen o eliminar objetos indeseados de esta. En más detalle, dada una imagen Q de $M \times N$ píxeles y una región (subconjunto de píxeles) dentro de ella $\Omega \subset M \times N$, el probema de inpainting consiste en dar valor a los píxeles de Ω de forma que los nuevos objetos formados sean visualmente plausibles (ver Figura 1).

Figura 1. Restauración de una imagen digital utilizando técnicas de inpainting

Una de las técnicas de *inpainting* más utilizada es la basada en buscar patrones semejantes dentro de la zona conocida de la imagen [5]. Esta técnica trata de rellenar cada píxel desconocido $(x, y) \in \Omega$ con el valor de intensidad del píxel conocido $(x', y') \in \Phi = M \times N \setminus \Omega$ cuyo vecindario sea el más parecido posible al vecindario del píxel desconocido. En detalle, dado un píxel desconocido $(x, y) \in \Omega$ y su vecindario $\Psi((x, y)) = \{(x - i, y - j) | - n \leq i, j \leq n \text{ y } n \in \mathcal{N}\}$,

$$I(x, y) = I(\arg \min_{(x',y')} d(\Psi((x, y)), \Psi((x', y'))))$$

donde d es una medida de lo diferentes que son los vecindarios $\Psi((x, y))$ y $\Psi((x', y'))$ (únicamente evaluado en aquellos píxeles conocidos). Por ejemplo, si tomamos como medida la suma de las diferencias al cuadrado tenemos

$$d(\Psi((x, y)), \Psi((x', y'))) = \sum_{(i,j) \in \Psi((x,y)) \cap \Phi} (I(i, j) - I(x' - x + i, y' - y + j))^2.$$

Las técnicas de *inpainting* basadas en patrones permiten recuperar características de textura en las zonas dañadas, a diferencia de las técnicas de *inpainting* basadas en interpolación [8], que producen efectos de suavizado (o *blurring*).

2.2. Medidas de contraste

En [2] se presenta el concepto de medida de contraste de una relación difusa. El concepto de contraste trata de medir la variación de los grados de pertenencia en una región específica de una relación difusa. Teniendo en cuenta el hecho de que una imagen en escala de grises puede verse como una relación difusa, podemos ver la medida de contraste como una forma de medir la variación de intensidades en una región específica.

Definición 1 *Sean* $X = \{1, \ldots, M\}$ *e* $Y = \{1, \ldots, N\}$. *Un contraste local es una función real de* $X \times Y$ *tal que:*

(LC1) $0 \leq C_l(x, y) \leq 1$ *para todo* $(x, y) \in X \times Y$*;*
(LC2) Si el grado de pertenencia de todos los elementos de una submatriz centrada en (x, y) *son iguales, entonces* $C_l(x, y) = 0$. *Es decir, si* $R(x - i, y - j) = q_0$ *con* $q_0 \in [0, 1]$ *constante para todo* $i, j = -n, \ldots, 0, \ldots, n$, *entonces* $C_l(x, y) = 0$*;*
(LC3) Si en la submatriz centrada en (x, y) *existe al menos un elemento con pertenencia nula y existe al menos un elemento con pertenencia igual a uno, entonces* $C_l(x, y) = 1$*;*
(LC4) El contraste local de (x, y) *no cambia si para todo* $i, j = -n, \ldots, 0, \ldots, n$ *tomamos* $1 - R(x - i, y - j)$ *en lugar de* $R(x - i, y - j)$.

Nota 1 *Nótese que en la definición original dada en [2], la medida de contraste local está asociada a una negación fuerte. En este trabajo hemos modificado la propiedad (LC4) por simplicidad.*

Ejemplo 1 *La función* $C_l(x, y) = \text{máx}_{-n \leq i,j \leq n} R(x-i, y-j) - \text{mín}_{-n \leq i,j \leq n} R(x-i, y-j)$ *es una medida de contraste local.*

Teniendo en cuenta la Definición 1 y el hecho de que una imagen puede verse como una relación difusa (normalizando las intensidades al intervalo $[0, 1]$), podemos obtener una imagen de contrastes locales simplemente aplicando una función de contraste local sobre cada píxel de la imagen. Cabe recordar que las medidas de contraste también han sido ampliamente estudiadas en la literatura como un ejemplo de textura (ver, por ejemplo [4]).

2.3. *Clustering*

Las técnicas de agrupamiento o *clustering* consisten en agrupar o clasificar un conjunto de objetos de tal forma que los objetos que forman parte de un mismo grupo (o *cluster*) sean muy similares entre sí y sean, a su vez, muy diferentes de los objetos que forman parte de otros grupos.

Uno de los algoritmos de clustering más conocidos es el algoritmo k-medias (o *k-means*) [6], que trata de minimizar la varianza intra-cluster. Dados $x_1, \ldots, x_n \in \mathcal{R}^m$ objetos y K número de clusters, el objetivo es encontrar una partición de los objetos $x_1, \ldots, x_n$ en K conjuntos $S_1, \ldots, S_K$ tal que se minimice la función de coste

$$\sum_{k=1}^{K} \sum_{x \in S_k} ||x - \mu_k||^2,$$

donde $\mu_k \in \mathcal{R}^m$ es la media aritmética de los elementos pertenecientes al conjunto S_k. A los objetos $\mu_k \in \mathcal{R}^m$ se les denomina centros o centroides y actúan como representantes o prototipos de los conjuntos $S_1, \ldots, S_K$.

El algoritmo *k-means* viene dado por los siguientes pasos:

1. Inicializar aleatoriamente los centros $\mu_1, \ldots, \mu_K \in \mathcal{R}^m$.

2. Para cada objeto x_i con $i = 1, \ldots, n$, asignarlo al conjunto S_j tal que

$$j = \arg \min_{k \in \{1,\ldots,K\}} ||x_i - \mu_k||^2.$$

3. Recalcular los centros de cada conjunto S_i como la media aritmética de los datos del conjunto

$$\mu_i = \frac{1}{|S_i|} \sum_{x_i \in S_i} x_i.$$

4. Repetir los pasos 2-3 hasta que los centros no cambien.

3. Algoritmo de restauración propuesto

En esta sección explicamos nuestra propuesta de algoritmo de restauración de imágenes utilizando algoritmos de *clustering* y características de contraste. Este algoritmo se divide en dos fases bien diferenciadas. La primera fase, o fase de preprocesamiento, consiste en realizar una clasificación de las ventanas no deterioradas de una imagen en diferentes clases atendiendo a las características de contraste de las propias ventanas. Una vez obtenida esta clasificación, la segunda fase, o fase de restauración, restaura la intensidad de cada píxel deteriorado utilizando la clasificación obtenida en la primera fase. Para ello, el algoritmo extrae las características no dañadas de la ventana que rodea al píxel deteriorado y realiza una búsqueda de la ventana más parecida dentro de una única clase (aquella que contenga las ventanas que tienen las mismas características de intensidad).

3.1. Fase de preprocesamiento

La primera fase parte de la imagen deteriorada y extrae todas aquellas ventanas de $n \times n$ (n siendo número entero) píxeles de la imagen en las que todos sus píxeles sean conocidos, es decir, todas aquellas ventanas que no tengan píxeles deteriorados. El objetivo del algoritmo es clasificar estas ventanas en grupos de tal forma que las ventanas pertenecientes a un mismo grupo tengan características similares.

Uno de los problemas de este algoritmo es elegir qué características utilizar para clasificar cada subventana. Aunque a primera vista parece razonable utilizar las propias intensidades (cada ventana es transformada en un elemento n^2-dimensional), enseguida nos dimos cuenta de que no es lo más útil. Esto puede verse en el ejemplo de la Figura 2. Supongamos que queremos restaurar el píxel marcado en rojo en la Figura 2(a) y elegimos como patrón aquella subventana más similar en términos de intensidad. De entre las subventanas (b) y (c) la más similar es esta última, por lo que reconstruiremos el píxel como muestra en (d).

Siguiendo con el ejemplo anterior, supongamos ahora que obtenemos, de cada subventana disponible, su información de contraste y la utilizamos para encontrar el patrón más parecido. En (e) mostramos el mismo píxel a restaurar pero habiendo extraído sus características de contraste. A continuación, (f) y (g)

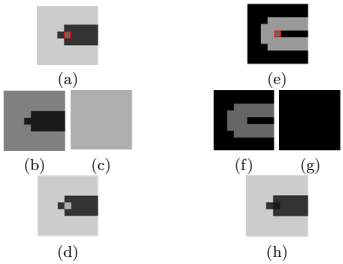

Figura 2. Ejemplo de ventanas deterioradas y su posible restauración utilizando características de intensidad o características de contraste

Algoritmo de preprocesamiento

Entrada: Imagen I de $M \times N$ píxeles; $\Omega \subset M \times N$ zona a restaurar; Medida de contraste local C_l; Tamaño de ventana n (impar); Número de clases K.

Salida: Conjuntos (clases) $S_1, \ldots, S_K$ de patrones, cada uno con $\mu_1, \ldots, \mu_K$ centroides asociados.

1: Obtener la zona conocida (disponible) de la imagen como $\Phi = M \times N \setminus \Omega$
2: Obtener la información de contraste de la zona conocida de I : $I_C(x, y) = C_l(I(x, y))$ para todo $(x, y) \in \Phi$
3: Obtener todas las subventanas $n \times n$ de I_C
4: Clasificar las subventanas del paso anterior en K clases $S_1, \ldots, S_K$ y obtener el centroide de cada clase $\mu_1, \ldots, \mu_K$.

corresponden con la información de contrastes de (b) y (c) respectivamente. Podemos observar que ahora el patrón más parecido es (f), ya que estructuralmente son más similares. Por tanto, la imagen reconstruida queda como se ve en (h). Nótese que la forma de restaurar la intensidad del píxel es tomar directamente la intensidad del píxel central de la subventana más parecida.

Nota 2 *Es importante recordar que cuando clasificamos cada ventana utilizando la información de contraste es necesario almacenar, además, la subventana original, ya que la restauración se hace tomando la intensidad original y no la información de contraste.*

3.2. Fase de restauración

La segunda fase del algoritmo parte de la clasificación obtenida en la primera fase y realiza la restauración, píxel a píxel, de Ω. En resumen, una vez elegido el píxel a restaurar $(x, y) \in \Omega$, obtenemos la información de contraste de su entorno $\Psi_C((x, y))$ y buscamos la subventana (patrón) más parecida. Para ello, en lugar de comparar con todas las subventanas obtenidas en el paso 3 del algoritmo de

preprocesamiento, buscamos el centroide (representante) $\mu_1, \ldots, \mu_K$ más parecido a $\Psi_C((x,y))$. Una vez obtenido, buscamos la subventana más parecida de entre las asignadas a la clase correspondiente. De esta manera, se logra agilizar el proceso de restauración ya que no es necesario buscar sobre toda la base de información, sino solamente sobre una fracción de ella.

Para elegir cuál es el primer píxel a restaurar, elegimos aquel que tenga más información correcta a su alrededor. Para ello, contamos cuántos píxeles conocidos (pertenecientes a Φ) hay en una ventana de $n \times n$ píxeles alrededor de cada píxel a restaurar. Elegiremos aquél que tenga mayor número de píxeles conocidos. Este orden prioriza los píxeles *a priori* más sencillos de restaurar ya que, una vez restaurados, pasarán a formar parte de Φ y serán utilizados para restaurar futuros píxeles de Ω.

Algoritmo de restauración

Entrada: Imagen I de $M \times N$ pixels; $\Omega \subset M \times N$ zona a restaurar; Medida de contraste local C_l; Tamaño de ventana n (impar); Conjuntos $S_1, \ldots, S_K$ y centroides $\mu_1, \ldots, \mu_K$.

Salida: Imagen I restaurada ($\Phi = M \times N$)

1: Para cada píxel $(x,y) \in \Omega$ obtener la ventana $n \times n$ centrada en él $\Psi((x,y)$ y calcular el número de píxeles conocidos en $\Psi((x,y)) \cap \Phi$
2: Elegir el píxel (x',y') con mayor número de píxeles conocidos
3: Obtener la información de contraste $\Psi_C((x',y'))$
4: Comparar $\Psi_C((x',y'))$ con $\mu_1, \ldots, \mu_K$ y elegir la clase asociada al centroide más parecido
5: Comparar $\Psi((x',y'))$ con todas las subventanas de la clase seleccionada en el paso anterior
6: Asignar a $I(x',y')$ el valor del píxel central de la subventana más parecida seleccionada en el paso anterior y eliminarlo de Ω
7: Si $card(\Omega) > 0$ volver al paso 2

Una vez seleccionado el píxel a restaurar $(x,y) \in \Omega$, tomamos una ventana de $n \times n$ píxeles alrededor, que llamamos $\Psi((x,y))$. Utilizando únicamente la zona conocida de $\Psi((x,y))$, calculamos su información de contraste $\Psi_C((x,y))$ utilizando la misma función de contraste local C_l que en la fase de preprocesamiento. Como hemos comentado anteriormente, la idea es restringir el número de comparaciones para encontrar el patrón más parecido, así que comparamos $\Psi_C((x,y))$ con los centroides $\mu_1, \ldots, \mu_K$. Elegiremos la clase asociada al centroide más parecido. El último paso es encontrar la subventana (patrón) más parecido de aquellas que pertencen a la clase seleccionada. Recordamos que en este último paso, utilizaremos de nuevo las características de intensidad. El último paso consiste en asignar el valor de intensidad al pixel (x,y) como el valor del píxel central de la subventana más parecida dentro de la clase.

4. Resultados experimentales

En esta Sección presentamos algunos resultados obtenidos por nuestro algoritmo de restauración. Para ello, hemos fijado los siguientes parámetros: Tamaño de ventana $n = 9$; Medida de contraste local C_l dada en Ejemplo 1; Número de clases $K = 20$.

La primera prueba consiste en eliminar un objeto no deseado en una imagen. En la Figura 3 mostramos una imagen original (a) y en blanco (b) delimitamos el objeto que queremos eliminar, en este caso la figura de la mujer. En (c) mostramos el resultado obtenido por el algoritmo propuesto. Para comparar nuestros resultados, en (d) mostramos la restauración obtenida aplicando el mismo algoritmo de restauración pero sin fase de preprocesamiento y, por consecuencia, utilizando únicamente características de intensidad (y no de contraste).

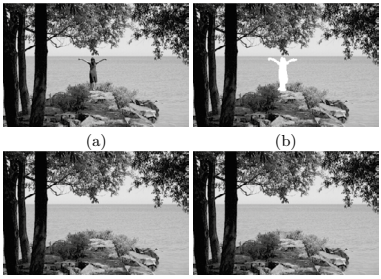

Figura 3. (a) Imagen original. (b) Imagen con los píxeles que vamos a eliminar marcados en blanco. (c) Imagen restaurada mediante el algoritmo de la Sección 3. (d) Imagen restaurada teniendo en cuenta sólo las intensidades.

Como podemos comprobar visualmente, en la zona del agua ambos resultados son bastante parecidos. Sin embargo, en los arbustos cercanos a las piernas de la mujer, nuestro algoritmo obtiene unos contornos mucho más definidos que la propuesta que sólo se basa en intensidades. De esta forma, nuestra solución mantiene unos bordes correctos que separan perfectamente la piedra de las hierbas, y aproxima de manera muy satisfactoria las texturas de ambas superficies.

A continuación trabajamos con una imagen en la que tenemos zonas perdidas y queremos que el algoritmo restaure la información original. En este caso, vamos a trabajar con la imagen Lena a la que le vamos a eliminar dos conjuntos de píxeles diferentes (marcados en blanco), en las Figuras 4(a) y 5(a).

En las Figuras 4 y 5 mostramos los resultados de este experimento obtenidos por nuestro algoritmo (b), así como los obtenidos cuando hacemos la restaura-

(a) (b) (c)

Figura 4. (a) En blanco se muestran las máscaras de píxeles que vamos a recontruir. (b) Reconstrucción utilizando características de contraste. (c) Reconstrucción utilizando solo intensidades.

ción teniendo en cuenta sólo las intensidades de los píxeles (c). Como se puede ver, nuestro algoritmo presenta imágenes menos difuminadas, y por lo tanto restauraciones más integradas en la imagen final. En la Figura 5, al trabajar con los contrastes conseguimos que el mechón de pelo tenga continuidad a lo largo de la zona restaurada, efecto que no se consigue con el otro algoritmo. Por tanto, podemos asegurar que visualmente el uso de contrastes locales mejora significativamente el resultado final frente al uso únicamente de intensidades.

(a) (b) (c)

Figura 5. (a) En blanco se muestran las máscaras de píxeles que vamos a recontruir. (b) Reconstrucción utilizando características de contraste. (c) Reconstrucción utilizando solo intensidades.

Por último, es importante hacer notar que, en una imagen de tamaño $M \times N$, el número de ventanas de tamaño $n \times n$ es $(M-n+1) \times (N-n+1)$. Por tanto, el número de comparaciones sin utilizar nuestra fase de preprocesamiento es muy elevado. De hecho, aun asumiendo el tiempo consumido por el algoritmo k-medias, los tiempos obtenidos con nuestro algoritmo disminuyen en un 70 % respecto a la propuesta sin fase de preprocesamiento.

Además, para el tamaño de ventana elegido ($n = 9$) hemos considerado un número de clases $K = 20$. En la Figura 6 se muestran los centroides obtenidos para la imagen de la Figura 4 (a). Podemos observar que los centroides obtenidos son suficientemente significativos y que para dicho tamaño de ventana no es necesario aumentar el número de clases. Sin embargo, creemos que conforme se aumente el tamaño de la ventana también será necesario aumentar el número de clases para no reducir la cantidad de información disponible.

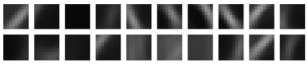

Figura 6. Centroides obtenidos de la clasificación de las ventanas disponibles en la imagen de Figura 4 (a)

5. Conclusiones y líneas futuras

En este trabajo hemos presentado un algoritmo de restauración en dos fases: fase de preprocesamiento y fase de restauración. La novedad del algoritmo es la propia fase de preprocesamiento, en la que realizamos una clasificación de las ventanas disponibles utilizando características de contraste en lugar de intensidad.

Los primeros resultados obtenidos muestran que el algoritmo propuesto no solo mejora el tiempo de computación debido a la clasificación previa, sino que también mejora visualmente los resultados por la utilización del contraste. Aunque esta propuesta está en fase inicial de desarrollo, creemos que podemos mejorar los resultados utilizando diferentes configuraciones (contrastes locales, medidas de similitud/comparación). Además, queremos realizar una comparativa más exhaustiva con otros métodos presentes en la literatura.

Agradecimientos

Este trabajo ha sido parcialmente financiado por el proyecto TIN2013-40765-P.

Referencias

1. Bertalmio, M., Sapiro, G., Caselles, V., Ballester, C. Image inpainting. In Proc. SIGGRAPH'00, New Orleans, USA, 417-–424 (2000).
2. Bustince, H., Barrenechea, E., Fernandez, J., Pagola, M., Montero, J., Guerra, C. Contrast of a fuzzy relation. Information Sciences 180, 1326–1344 (2010).
3. Buades, A. A non-local algorithm for image denoising. Computer Vision and Pattern Recognition 2, 60–65 (2005).
4. Chamorro-Martinez, J., Martinez-Jimenez, P.M., Soto-Hidalgo, J.M., Prados-Suarez, B. Perception-based fuzzy sets for visual texture modelling. Soft Computing 18, 2485–2499 (2014).
5. Criminisi, A., Pérez P., Toyama, K.: Region filling and object removal by exemplar-based image inpainting. IEEE Transactions on Image Processing 13, 1200–1212 (2004).
6. Hartigan, J. A., Wong, M. A.: Algorithm AS 136: A K-Means Clustering Algorithm. Journal of the Royal Statistical Society, Series C (Applied Statistics) 28, 100–108 (1979).
7. Kokaram, A.C., Morris, R.D., Fitzgerald, W.J., Rayner P.J.W. Interpolation of missing data in image sequences. IEEE Transactions on Image Processing 11, 1509–1591 (1995).
8. Ogden, J.M., Adelson, E.H., Bergen, J.R., Burt, P.J. Pyramid-based computer graphics. RCA Engineer, 30, 4—15 (1985).