CAPÍTULO 4.
DETECCIÓN DE PÍXELES CANDIDATOS.

En este capítulo vamos a centrarnos en describir tanto la teoría utilizada (descrita en el apartado 4.1) como la herramienta desarrollada en C para encontrar los píxeles candidatos a ser microcalcificación (descrita en el apartado 4.2).

Una adecuada selección de estos puntos resulta de vital importancia, ya que será sobre ellos sobre los que nos apoyaremos para determinar si existe, o no, una microcalcificación en su entorno. Las reglas de decisión para este fin serán objeto de estudio en el apartado 4 de este capítulo.

4.1 PREDICCIÓN LINEAL. ALGORITMO DE BURG.

La predicción lineal es un tema de enorme importancia en el tema de procesado digital de imágenes y tiene una gran cantidad de aplicaciones prácticas. Se trata de predecir valores desconocidos de una cierta señal a partir de otros valores de misma que sí se conocen.

Todas las técnicas de predicción se basan en el concepto de correlación entre los diferentes valores de la señal que se está prediciendo, es decir, se supone la existencia de una cierta similitud entre los valores conocidos de la señal y aquéllos que se pretende predecir. Los valores predichos se parecerán más a los reales en tanto en cuanto la correlación de la señal sea más alta.

Desde el punto de vista de la predicción, las señales se tratan como procesos aleatorios como única manera de contemplar la incertidumbre existente en determinados valores de la señal que, en principio, podrían ser cualesquiera dentro de un determinado rango. Lógicamente sólo tendrá sentido plantearse la predicción de señales estacionarias ya que, en cualquier otro caso, no parece lógico pensar que puedan inferirse con éxito ciertos valores de la señal a partir de otros, ya que se desconoce su dependencia estadística.

Dentro del campo de las telecomunicaciones el concepto de predicción se aplica en innumerables áreas, tales como el tratamiento digital de señales, tratamiento de la voz o transmisión de datos. En este sentido, y según la aplicación que se esté considerando, se habla de predicción en una dimensión (1D), de dos dimensiones (2D), de tres dimensiones (3D); predicción hacia delante y predicción hacia detrás.

Para nuestro proyecto nos centraremos en la predicción lineal, que es aquélla en la cual el valor predicho de la señal se obtiene como combinación lineal otros valores conocidos.

4.1.1 Predicción lineal 1D

Desarrollaremos aquí, brevemente, la base teórica que subyace bajo el concepto de predicción lineal 1D, que nos ayudará a comprender la predicción en más dimensiones posteriormente. Consideraremos el problema de predecir linealmente el valor de un proceso estacionario aleatorio tanto hacia delante como hacia atrás en el tiempo (la variable aleatoria podría ser cualquiera aunque lo habitual al trabajar en una sola dimensión es hacerlo con el tiempo).

4.1.1.1 Predicción lineal 1D hacia delante.

Se trata de predecir linealmente un valor futuro de un proceso aleatorio estacionario a partir de la observación de valores pasados del proceso. En particular, consideraremos el predictor lineal hacia delante de una muestra, que forma la predicción del valor x(n), donde n indica el instante de tiempo considerado en el proceso aleatorio x, mediante una combinación lineal ponderada de los valores pasados x(n-1), x(n-2),…, x(n-p). Así, el valor predicho de x(n) es:

$$x_p(n) = -\sum_{k=1}^{p} a_p x(n-k) \qquad \textbf{(4.1)}$$

donde los $\{a_p(k)\}$ representan los pesos de la combinación lineal. Estos pesos se denominan coeficientes de predicción del *predictor lineal hacia delante de una muestra*, ya que sólo se predice una, de orden p (hemos utilizado p muestras pasadas para predecirla).

La diferencia entre el valor x(n) y $x_p(n)$ predicho se denomina error de predicción hacia delante y se denota por $f_p(n)$:

$$f_p(n) = x(n) - x_p(n) = x(n) + \sum_{k=1}^{p} a_p(k)x(n-k) = \sum_{k=0}^{p} a_p(k)x(n-k), a_p(0) = 1 \qquad \textbf{(4.2)}$$

Veamos una representación gráfica de las ecuaciones anteriores, donde se describe el proceso de obtención de los errores de predicción a partir de los datos de entrada y viceversa.

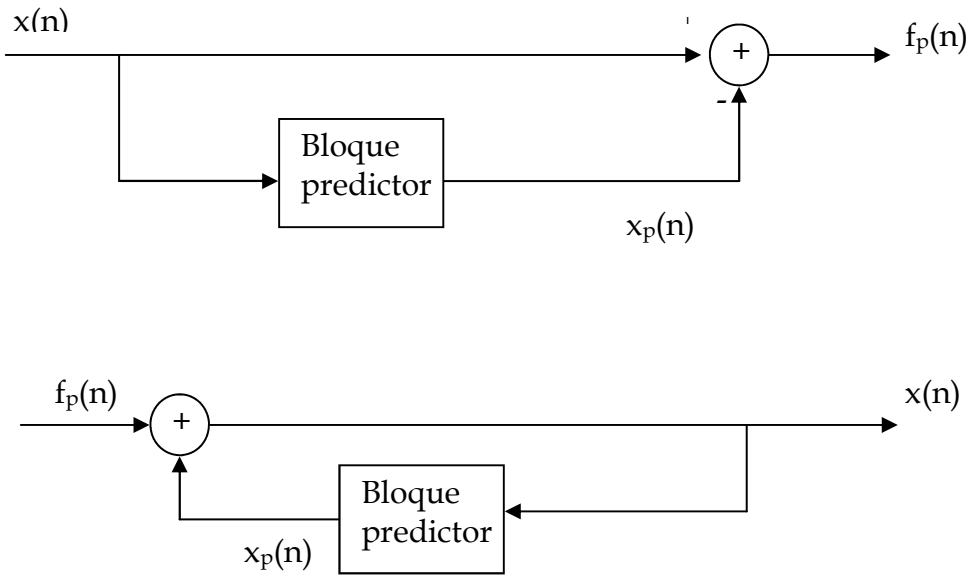

Figura 4.1. Esquema general de predicción hacia delante

Se llama filtro de error de predicción hacia delante a aquél que, dada la secuencia de entrada {x(n)}, ofrece como secuencia de salida {$f_p(n)$} y cuya función de transferencia es:

$$A_p(z) = \sum_{k=0}^{p} a_p(k) z^{-k}, a_p(0) = 1 \qquad \textbf{(4.3)}$$

Este filtro FIR tiene la siguiente estructura:

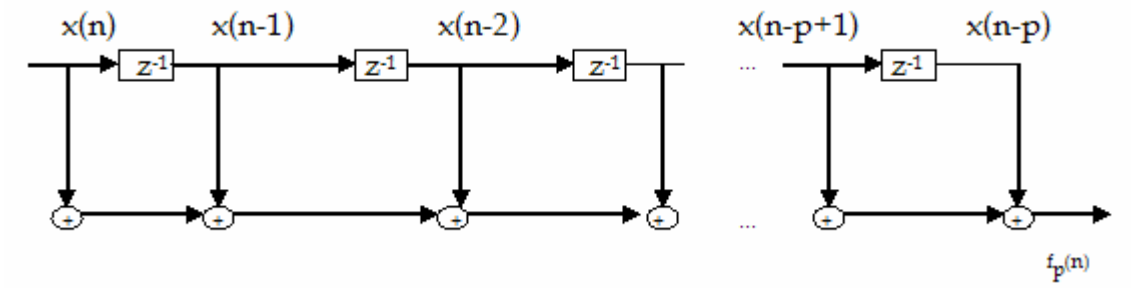

Figura 4.2. Filtro de error de predicción hacia delante.

Si se quiere tener una buena predicción de x(n), deberá minimizarse de alguna forma el valor del error de predicción $f_p(n)$. En concreto, el criterio más usado es el de minimización del valor cuadrático medio del error $E[|f_p(n)|^2]$, cuya expresión es la que sigue:

$$\varepsilon_p^f = E\left[\left|f_p(n)\right|^2\right] = r_{xx}(0) + 2\,\mathrm{Re}\left[\sum_{k=1}^{p} a_p^*(k) r_{xx}(k)\right] + \sum_{k=1}^{p}\sum_{k=1}^{p} a_p^*(1) a_p(k) r_{xx}(1-k) \qquad \textbf{(4.4)}$$

donde r_{xx} es la función de autocorrelación del proceso aleatorio x. La ecuación 4.4 se obtiene directamente de aplicar l fórmula del error cuadrático medio

sobre la ecuación 4.2. Este criterio se denomina **MMSE** (Minimum Mean Square Error).

La minimización de este valor cuadrático medio supone el cálculo de su gradiente y la igualación a cero y el resultado conduce al siguiente conjunto de ecuaciones lineales:

$$r_{xx}(l) = -\sum_{k=1}^{p} a_p(k) r_{xx}(l-k)\text{, } con\ l=1,\ 2,\ \ldots,\ p \qquad \textbf{(4.5)}$$

Estas ecuaciones se denominan *ecuaciones normales* para los coeficientes del predictor lineal. El mínimo error cuadrático de predicción es, simplemente:

$$\min\left[\varepsilon_p^f\right] \equiv E_p^f = r_{xx}(0) + \sum_{k=1}^{p} a_p(k) r_{xx}(-k) \qquad \textbf{(4.6)}$$

Es muy habitual utilizar una representación matricial de las ecuaciones normales de la forma: $\mathbf{R_p a_p = -r_p}$, donde los vectores de la matriz se definen como sigue:

$$\mathbf{R_p} = \begin{bmatrix} r_{xx}(0) & r_{xx}^*(1) & \ldots & r_{xx}^*(p-1) \\ r_{xx}(1) & r_{xx}(0) & \ldots & r_{xx}^*(p-2) \\ \ldots & \ldots & \ldots & \ldots \\ r_{xx}(p-1) & r_{xx}(p-2) & \ldots & r_{xx}(0) \end{bmatrix};\ \mathbf{a_p} = \begin{bmatrix} a_p(1) \\ a_p(2) \\ \ldots \\ a_p(p) \end{bmatrix};\ \mathbf{r_p} = \begin{bmatrix} r_{xx}(1) \\ r_{xx}(2) \\ \ldots \\ r_{xx}(p) \end{bmatrix} \qquad \textbf{(4.7)}$$

Nótese que en esta definición se ha considerado la propiedad de la función de autocorrelación por la que $r_{xx}(j) = r^*_{xx}(-j)$ Vj. Se cumple que $R_p(i,j) = r_{xx}(i-j)$, es decir, la matriz de autocorrelación es una matriz de Toeplitz. Como además, $R_p(i,j) = R^*_p(j,i)$, la matriz también es hermítica.

4.1.1.2 Predicción lineal 1D hacia atrás.

En este caso se supondrán conocidos una serie de valores del proceso estacionario x(n), x(n-1),…, x(n-p+1) y el objetivo es, a partir de ellos, predecir el valor inmediatamente anterior de ese mismo proceso. En este caso, se utilizará un *predictor lineal hacia atrás de una muestra,* ya que se predice sólo una, de orden p, ya que se utilizan p muestras conocidas para predecir el anterior. El valor predicho linealmente será:

$$x_p(n-p) = -\sum_{k=1}^{p} b_p x(n-k) \qquad \textbf{(4.8)}$$

En esta ocasión son los valores $b_p(k)$ los coeficientes de predicción hacia atrás. La diferencia entre el valor real x(n-p) y el valor estimado $x_p(n-p)$ se denomina error de predicción hacia atrás y se denota por $g_p(n)$:

$$g_p(n) = x(n-p) - x_p(n-p) = x(n-p) + \sum_{k=1}^{p-1} b_p(k)x(n-k) = \sum_{k=0}^{p} b_p(k)x(n-k), b_p(0) = 1 \quad \textbf{(4.9)}$$

El filtro que proporciona el error del valor predicho se denomina filtro de error de la predicción hacia atrás, y su función de transferencia es:

$$B_p(z) = \sum_{k=0}^{p} b_p(k)z^{-k}, b_p(p) = 1 \quad \textbf{(4.10)}$$

Como se puede ver, la estructura de este filtro FIR es idéntica a la del filtro de predicción hacia delante.

De nuevo se aplicará el criterio MMSE al filtro de predicción hacia atrás, obteniéndose en este caso nuevas ecuaciones normales que, en forma matricial, son: $\mathbf{R_p b_p = -r_p^{b*}}$, donde la matriz de autocorrelación es la misma que en el caso de predicción hacia delante y donde:

$$\mathbf{b_p} = \begin{bmatrix} b_p(1) \\ b_p(2) \\ \dots \\ b_p(p) \end{bmatrix}; \mathbf{r_p^{b*}} = \begin{bmatrix} r_{xx}^{*}(1) \\ r_{xx}^{*}(2) \\ \dots \\ r_{xx}^{*}(p) \end{bmatrix} \quad \textbf{(4.11)}$$

4.1.1.3 Solución de las ecuaciones normales. El algoritmo de Levinson-Durbin.

Tal y como se vio anteriormente, la minimización del valor cuadrático medio del error de predicción hacia delante da lugar a lo que se denominan ecuaciones normales y a un cierto valor mínimo de dicho error. Ambas ecuaciones pueden unificarse en lo que se llama *ecuaciones normales aumentadas,* que son las siguientes:

$$\sum_{k=0}^{p} a_p(k)r_{xx}(l-k) = \begin{cases} E_p^f, l = 0 \\ 0, l = 1,2,\dots,p \end{cases} \quad \textbf{(4.12)}$$

Interesa resolver estas ecuaciones y obtener de esa forma los coeficientes de predicción hacia delante. Un algoritmo bastante eficiente para encontrar la solución es el llamado **algoritmo de recursión de Levinson-Durbin**, que explota la simetría espacial de la matriz de autocorrelación $\mathbf{R_p}$ y que pasamos a describir a continuación someramente.

La clave de la solución del método propuesto, que aprovecha la propiedad Toeplitz de la matriz, consiste en proceder recursivamente, empezando con un predictor de orden m=1 (un coeficiente) y después ir aumentando el orden recursivamente usando las soluciones de orden menor para obtener la solución al siguiente orden superior. Así, la solución al predictor de primer orden obtenida resolviendo la ecuación 4.12 es:

$$a_1(1) = -\frac{r_{xx}(1)}{r_{xx}(0)} \tag{4.13}$$

Y el MMSE resultante es:

$$E_1^f = r_{xx}(0) + a_1(1) r_{xx}(-1) = r_{xx}(0)\left[1 - |a_1(1)|^2\right] \tag{4.14}$$

El siguiente paso consiste en resolver para los coeficientes {$a_2(1)$, $a_2(2)$} del predictor de segundo orden y expresar la solución en función de $a_1(1)$. Las dos ecuaciones obtenidas a partir de 4.12 son:

$$\begin{aligned} a_2(1) r_{xx}(0) + a_2(2) r_{xx}^*(1) &= -r_{xx}(1) \\ a_2(1) r_{xx}(1) + a_2(2) r_{xx}(0) &= -r_{xx}(2) \end{aligned} \tag{4.15}$$

Si usamos la solución de la ecuación 4.13 para eliminar $r_{xx}(1)$, se obtiene:

$$\begin{aligned} a_2(2) &= -\frac{r_{xx}(2) + a_1(1) r_{xx}(1)}{r_{xx}(0)\left[1 - |a_1(1)|^2\right]} = -\frac{r_{xx}(2) + a_1(1) r_{xx}(1)}{E_1^f}; \\ a_2(1) &= a_1(1) + a_2(2) a_1^*(1); \end{aligned} \tag{4.16}$$

De esta forma, se han obtenido los coeficientes del predictor hacia delante de segundo orden. Procediendo de esta manera, se pueden expresar los coeficientes del predictor de orden m en función de los coeficientes del predictor de orden m-1. Esto permite escribir el vector de coeficientes de predicción de orden m $\mathbf{a}_m$ como la suma de dos vectores:

$$\mathbf{a}_m = \begin{bmatrix} a_m(1) \\ a_m(2) \\ \cdots \\ a_m(m) \end{bmatrix} = \begin{bmatrix} a_{m-1} \\ \cdots \\ 0 \end{bmatrix} + \begin{bmatrix} d_{m-1} \\ \cdots \\ K_m \end{bmatrix} \tag{4.17}$$

donde $\mathbf{a}_{m-1}$ es el vector de coeficientes de predicción del predictor de orden m-1 y el vector $\mathbf{d}_{m-1}$ y el escalar $\mathbf{K}_m$ tienen que ser determinados.

Se realiza ahora una división de la matriz de autocorrelación del proceso estacionario x, de orden m x m, Γ_m como:

$$\Gamma_m = \begin{bmatrix} \Gamma_{m-1} & \left(r_{m-1}^b\right)^* \\ \left(r_{m-1}^b\right)^t & r_{xx}(0) \end{bmatrix} \tag{4.18}$$

donde $\left(r_{m-1}^b\right)^t = [r_{xx}(m-1)\ r_{xx}(m-2)\ \ldots\ r_{xx}(1)]$, el asterisco * denota el complejo conjugado y r_m^t denota la traspuesta de r_m. El superíndice b en r_{m-1} denota el vector $r^t_{m-1} = [r_{xx}(1)\ r_{xx}(2)\ \ldots\ r_{xx}(m-1)]$ con los elementos tomados en orden inverso.

Las ecuaciones normales vistas matricialmente $\Gamma_m\mathbf{a}_m=-\mathbf{r}_m$ presentan la siguiente forma:

$$\begin{bmatrix}\Gamma_{m-1} & r_{m-1}^{b*}\\ r_{m-1}^{bt} & r_{xx}(0)\end{bmatrix}\left\{\begin{bmatrix}a_{m-1}\\ 0\end{bmatrix}+\begin{bmatrix}d_{m-1}\\ K_m\end{bmatrix}\right\}=-\begin{bmatrix}r_{m-1}\\ r_{xx}(m)\end{bmatrix} \tag{4.19}$$

Esta ecuación puede subdividirse en dos:

$$\begin{aligned}&\Gamma_{m-1}a_{m-1}+\Gamma_{m-1}d_{m-1}+K_m r_{m-1}^{\ b*}=-r_{m-1}\\ &r_{m-1}^{\ bt}a_{m-1}+r_{m-1}^{\ bt}\Gamma_{m-1}+K_m r_{xx}(0)=-r_{xx}(m)\end{aligned} \tag{4.20}$$

Como se cumple la ecuación normal $\Gamma_{m-1}\mathbf{a}_{m-1}=-\mathbf{r}_{m-1}$ entonces, de la primera de las dos ecuaciones anteriores:

$$d_{m-1}=-K_m\Gamma_{m-1}^{\ -1}r_{m-1}^{\ b*}=K_m a_{m-1}^{\ b*}=K_m\begin{bmatrix}a_{m-1}^{*}(m-1)\\ a_{m-1}^{*}(m-2)\\ \cdots\\ a_{m-1}^{*}(1)\end{bmatrix} \tag{4.21}$$

Usamos ahora la ecuación 4.21 para obtener K_m. Si eliminamos $\mathbf{d}_{m-1}$ en 4.20, obtenemos:

$$\begin{aligned}&K_m\left[r_{xx}(0)+r_{m-1}^{bt}a_{m-1}^{b*}\right]+r_{m-1}^{bt}a_{m-1}=-r_{xx}(0)\\ &K_m=-\frac{r_{xx}(m)+r_{m-1}^{bt}a_{m-1}}{r_{xx}(0)+r_{m-1}^{bt}a_{m-1}^{b*}}\end{aligned} \tag{4.22}$$

Por lo tanto, sustituyendo las soluciones 4.21, 4.22 en 4.17, se tiene la recursión deseada par los coeficientes predictores en el algoritmo de Levinson-Durbin como:

$$\begin{aligned}&a_m(m)=K_m=-\frac{r_{xx}(m)+r_{m-1}^{bt}a_{m-1}}{r_{xx}(0)+r_{m-1}^{bt}a_{m-1}^{b*}}=-\frac{r_{xx}(m)+r_{m-1}^{bt}a_{m-1}}{E_m^f}\\ &a_m(k)=a_{m-1}(k)+K_m a_{m-1}^{*}(m-k)=a_{m-1}(k)+a_m(m)a_{m-1}^{\ *}(m-k)\\ &k=1,2,\ldots,m-1\\ &m=1,2,\ldots,p\end{aligned} \tag{4.23}$$

Finalmente, puede determinarse la expresión para el MMSE, que en el caso de un predictor de orden m será:

$$E_1^f=r_{xx}(0)+a_1(1)r_{xx}(-1)=r_{xx}(0)\left[1-|a_1(1)|^2\right] \tag{4.24}$$

donde $E^f_0=r_{xx}(0)$. Como los coeficientes $a_m(m)$ satisfacen la propiedad de que $|a_m(m)|\leq 1$, el MMSE para la secuencia de predictores satisface la condición:

$$E_0^f\geq E_1^f\geq E_2^f\geq\ldots\geq E_p^f \tag{4.25}$$

Es decir, el valor medio del error cuadrático de predicción disminuye a medida que aumenta el orden del predictor, o sea, a medida que se utiliza un mayor número de muestras anteriores para predecir la muestra actual.

Esto concluye la derivación del algoritmo Levinson-Durbin para resolver las ecuaciones normales $\Gamma_m \mathbf{a}_m = -r_m$, para m=0,1,…,p. Se ha llevado a cabo la aplicación del algoritmo para obtener los coeficientes del filtro de predicción hacia delante. Sin embargo, es conocida la relación de éstos con los coeficientes de predicción hacia atrás: $\mathbf{a}_m = \mathbf{b}^{b^*}_m$, de manera que, a la postre, encontrar la solución del algoritmo implicará conocer tanto los coeficientes de predicción hacia delante como hacia atrás.

4.1.1.4 El método de Burg para los coeficientes de predicción.

Como se ha visto en el punto anterior, el algoritmo de Levinson-Durbin permite determinar el valor de los coeficientes de predicción a partir de las funciones autocorrelación. En este apartado se estudia un método alternativo llamado algoritmo de Burg, en el cual dichos coeficientes se calculan a partir de los errores de predicción hacia delante y hacia atrás. La principal ventaja radica en que se evita el cálculo de las funciones autocorrelación que, en primer lugar, son sólo una aproximación (y no demasiado buena) al valor real de la autocorrelación estadística y, en segundo lugar, supone un gasto computacional importante.

Este método ideado por Burg en el año 1968 para estimar los coeficientes de predicción, se puede ver como un método en celosía con minimización cuadrática recursiva en el orden, basada en la minimización de los errores hacia delante y hacia atrás en los predictores lineales, con la restricción de que dichos coeficientes de predicción satisfacen la recursión Levinson-Durbin.

Para derivar el estimador supóngase que se tienen los datos x(n) con n=0,1,…,N-1, y considérense las estimas de predicción lineal hacia delante y hacia detrás de orden m dadas por:

$$\hat{x}(n) = -\sum_{k=1}^{m} a_m(k)\,x(n-k)$$
$$\hat{x}(n-m) = -\sum_{k=1}^{m} a_m^{*}(k)\,x(n+k-m) \qquad \textbf{(4.26)}$$

y los errores hacia delante y hacia atrás definidos como $f_m(n) = x(n) - \hat{x}(n)$ y $g_m(n) = x(n-m) - \hat{x}(n-m)$ donde $a_m(k)$, k=0, …,m-1; m=1, …,p. El error de mínimos cuadrados es:

$$\varepsilon_m = \sum_{n=m}^{N-1} \left[|f_m(n)|^2 + |g_m(n)|^2 \right] \qquad \textbf{(4.27)}$$

Este error se va a minimizar seleccionando los coeficientes de predicción, sujetos a satisfacer la recursión de Levinson-Durbin dada por:

$$a_m(k) = a_{m-1}(k) + K_m a_{m-1}^*(m-k) \qquad \textbf{(4.28)}$$
$$\text{con} \quad \begin{matrix} 1 \le k \le m-1 \\ 1 \le m \le p \end{matrix}$$

donde $K_m = a_m(m)$ es el coeficiente de reflexión m-ésimo en la realización en celosía del filtro predictor. Cuando se sustituye la ecuación 4.28 en las expresiones para $f_m(n)$ y $g_m(n)$, el resultado es el siguiente par de ecuaciones recursivas en el orden para los errores de predicción hacia delante y hacia atrás:

$$\begin{aligned} f_m(n) &= f_{m-1}(n) + K_m g_{m-1}(n-1) \\ g_m(n) &= g_{m-1}(n-1) + K^*_m f_{m-1}(n) \end{aligned} \qquad \textbf{(4.29)}$$
$$\text{con m=1, ..., p}$$

con la inicialización $f_0(n) = g_0(n) = x(n)$.

La siguiente figura ilustra el filtro en celosía de p etapas en forma de diagrama de bloques junto con una etapa típica que muestra los cálculos dados por las ecuaciones 4.29.

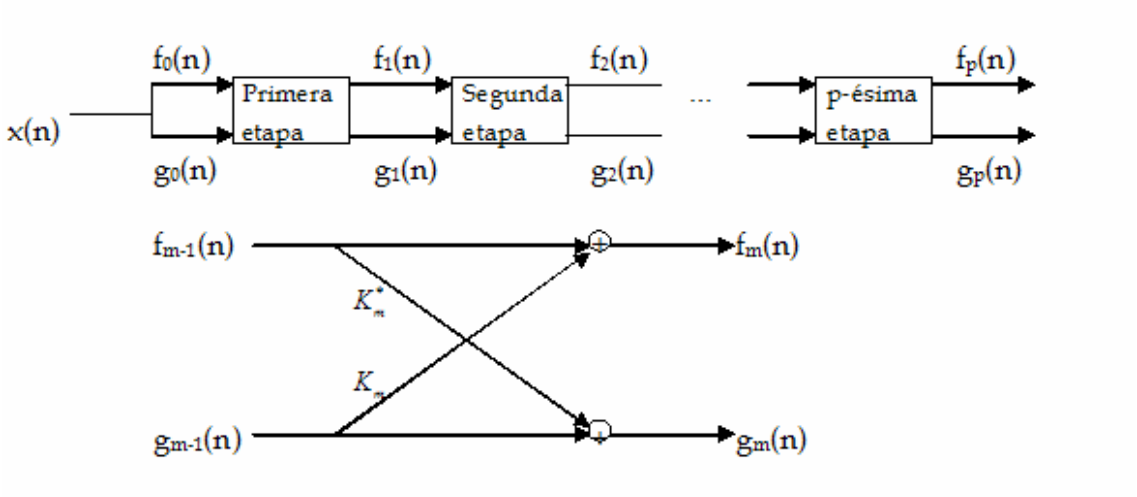

Figura 4.3. Filtro de errores de predicción en celosía. Etapa m-ésima.

Ahora si se sustituye 4.29 en 4.28 y se realiza la minimización de ε_m con respecto a los coeficientes de reflexión complejos K_m, se obtiene:

$$\hat{K}_m = \frac{-\sum_{n=m}^{N-1} f_{m-1}(n) g^*_{m-1}(n-1)}{\frac{1}{2}\sum_{n=m}^{N-!}\left[\left|f_{m-1}(n)\right|^2 + \left|g_{m-1}(n-1)\right|^2\right]} \text{, con m=1, 2, …,p} \qquad \textbf{(4.30)}$$

El término del numerador en la ecuación 4.30 es una estima de la correlación cruzada entre los errores de predicción hacia delante y hacia atrás. Con los factores de normalización del denominador de la ecuación 4.30, queda claro que $|K_m|<1$, así que el modelo todo polos obtenido de los datos es estable.

Cabe destacar que el denominador 4.30 es simplemente una estima de mínimos cuadrados de los errores de predicción E^f_{m-1} y E^b_{m-1}. Por lo tanto, la ecuación 4.30 se puede expresar como:

$$\hat{K}_m = \frac{-\sum_{n=m}^{N-1} f_{m-1}(n) g^*_{m-1}(n-1)}{\frac{1}{2}\left[\hat{E}^f_{m-1} + \hat{E}^b_{m-1}\right]} \qquad \textbf{(4.31)}$$

donde $\hat{E}^f_{m-1} + \hat{E}^b_{m-1}$ es una estima del error cuadrático medito total E_m.

Puede observarse que el denominador en la ecuación 4.31 se puede calcular de forma recursiva de acuerdo con la relación:

$$\hat{E}_m = \left(1 - \left|\hat{K}_m\right|^2\right) + \hat{E}_{m-1} - \left|f_{m-1}(m-1)\right|^2 - \left|g_{m-1}(m-2)\right|^2 \qquad \textbf{(4.32)}$$

donde $\hat{E}_m \equiv \hat{E}^f_m + \hat{E}^b_m$ es el error total de mínimos cuadrados.

Resumiendo, el algoritmo de Burg calcula los coeficientes de reflexión de la estructura en celosía equivalente como se especifica en las ecuaciones 4.31 y 4.32, y el algoritmo de Levinson-Durbin se usa para obtener finalmente los coeficientes de predicción lineal según la ecuación 4.28.

La principal ventaja del algoritmo de Burg par estimar los coeficientes de predicción es que, además de producir un modelo AR estable, presenta una importante eficiencia computacional frente al algoritmo de Levinson-Durbin ya que evita el cálculo de las funciones de autocorrelación.

4.1.2 Predicción lineal aplicada al tratamiento de imágenes.

En el punto anterior se ha repasado la predicción unidimensional (1D), en la cual existía una señal que variaba su valor respecto a una cierta variable y

el objetivo era predecir el valor de una cierta muestra de la señal a partir del valor de dicha señal en instantes anteriores o posteriores. Pero, ¿qué ocurre cuando trabajamos con imágenes digitalizadas?

Una imagen digital está constituida por un conjunto de píxeles ordenados en filas y columnas, es decir, estas imágenes tienen un *carácter bidimensional*. Hablar de predicción para una imagen supone ser capaz de predecir el valor que tendrá un cierto píxel de la misma a partir de los valores de otros píxeles ya conocidos. Ahora bien, en el caso más general, un píxel de una imagen tiene píxeles vecinos a su izquierda, a su derecha, arriba y abajo y, en todas las diagonales. ¿A partir de cuales de ellos se hace la predicción?

Una primera opción sería considerar la predicción 1D ya estudiada, de forma que un cierto píxel sería predicho bien a partir de otros que se encuentran situados en su misma horizontal, bien a partir de otros que se encuentran en su misma vertical dentro de la imagen. Esta idea queda reflejada en la siguiente figura:

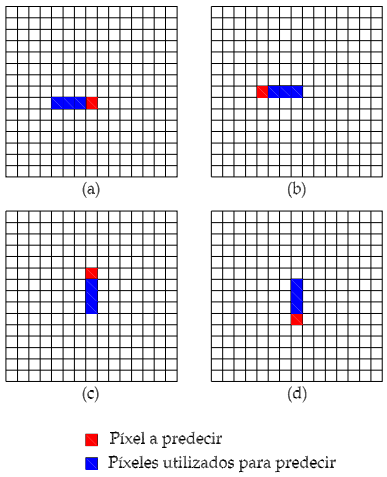

Figura 4.4. Predicción lineal 1D en imágenes digitales.

Denotando una cierta imagen digitalizada como I(x,y) donde x es el índice de la fila e y el índice de la columna, la predicción 1D sería:

$$\begin{aligned}
(a) &\longrightarrow I(x,y) = \sum_{k=1}^{N} a_N(k) I(x, y-k) \\
(b) &\longrightarrow I(x,y) = \sum_{k=1}^{N} a_N(k) I(x, y+k) \\
(c) &\longrightarrow I(x,y) = \sum_{k=1}^{N} a_N(k) I(x+k, y) \\
(d) &\longrightarrow I(x,y) = \sum_{k=1}^{N} a_N(k) I(x-k, y)
\end{aligned} \qquad \textbf{(4.33)}$$

Sin embargo, la predicción 1D no es, ni mucho menos, la más indicada cuando se trabaja con imágenes ya que parece lógico pensar que el valor de un cierto píxel dependerá, en la inmensa mayor parte de los casos, de todos los píxeles que le rodean en esa imagen y no sólo de los que están en su misma vertical u horizontal. Con carácter general será mucho más aproximada al valor real de un píxel aquella predicción que se haya realizado considerando todo el *entorno* de ese píxel y no aquélla en que sólo se considere una cierta dirección. De esta forma, se llega al concepto de predicción 2D, que matemáticamente se expresa:

$$I(x,y) = \sum_i \sum_j a(i,j) I(x-i, y-j), [i,j] \in ROS \qquad \textbf{(4.34)}$$

donde ROS (del inglés *region of summation*) abarcar la región de la imagen que se utilizará para realizar la predicción. La elección de la ROS es un proceso delicado y su extensión y forma dependerá de la aplicación concreta con la que se trabaje dentro del campo de la predicción.

Una posible aplicación es la siguiente: en ocasiones al recibir una cierta imagen, y por errores en la transmisión, aparecen ciertos píxeles aislados con valores muy diferentes a los del resto de píxeles bien recibidos; éstos se llaman **píxeles fuera de rango** y para eliminarlos se suele desechar su valor y sustituirlo por el resultado de la predicción a partir de los píxeles que lo rodean. En este caso, la ROS sería un *anillo* alrededor del píxel fuera de rango en cuestión.

Por último, hablaremos de la predicción 3D, muy utilizada en el caso de *imágenes en movimiento*. Éstas no son más que una ilusión, en realidad son una seria de imágenes fijas que se suceden a una velocidad suficientemente rápida como para que el ojo humano no lo perciba y se tenga la sensación de estar observando figuras que se mueven. Naturalmente, para que esto sea así, dos imágenes consecutivas en la secuencia deberán ser lo suficientemente parecidas para que no se noten saltos bruscos, sino variaciones suaves en la posición.

Se tiene entonces que existe una relación entre el valor de los píxeles en una imagen y el valor de los mismos en imágenes anteriores de la secuencia. Esta circunstancia puede ser aprovechada a la hora de realizar una predicción, de forma que ahora un píxel no se predecirá sólo a partir del valor de otros píxeles cercanos de la misma imagen, sino que también se utilizará el valor de esos mismos píxeles y del propio píxel a predecir en imágenes anteriores de la

secuencia de imágenes que crea la ilusión de imagen en movimiento. La formulación matemática de este tipo de predicción es:

$$I(x,y,n) = \sum_{k}\sum_{i}\sum_{j} a(i,j,k)I(x-i,y-j,n-k), [i,j] \in ROS \quad y\ k=0,1,2,\ldots \qquad \textbf{(4.35)}$$

donde el índice n indica el instante de tiempo considerado en la secuencia de imágenes, de manera que k valdrá 0 si no se usan imágenes anteriores a la actual para predecir; k valdrá 1 si se usa para predecir la imagen fija justamente anterior; k valdrá 2 si se usan las 2 imágenes fijas anteriores y así sucesivamente hasta llegar al orden deseado. Gráficamente:

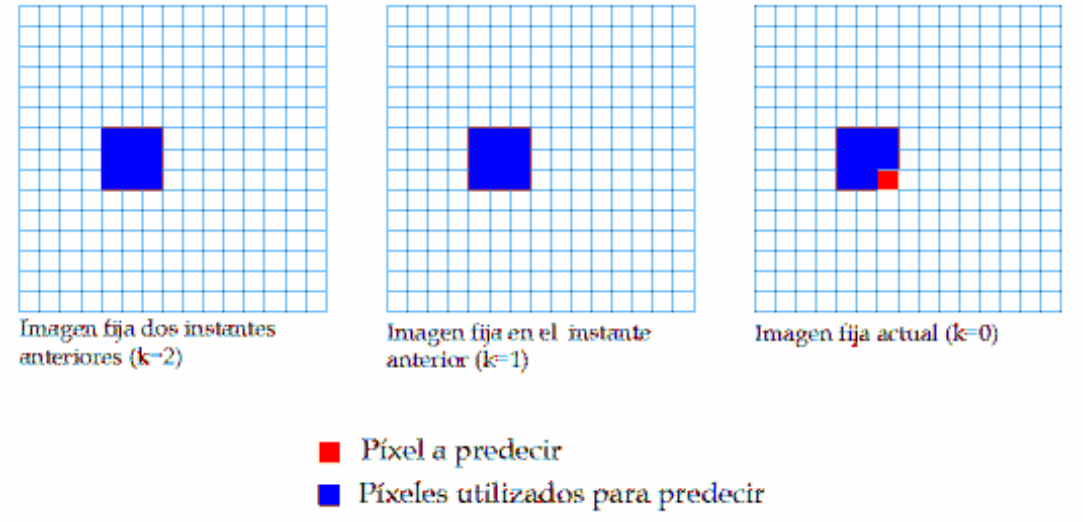

Figura 4.5. Predicción lineal 3D en imágenes digitales.

En resumidas cuentas, la predicción lineal 1D no es apropiada para tratar imágenes y la predicción 3D se aplica a imágenes en movimiento, hecho que escapa al objetivo de este proyecto. Por esta razón, será la predicción lineal 2D la que ocupará el grueso del proyecto y a la que se dedicará un estudio en profundidad para su aplicación a la detección de puntos singulares.

4.1.3 Algoritmo para el cálculo directo de coeficientes de predicción lineal 2D.

A lo largo de este apartado nos plantearemos el problema de la predicción lineal de los píxeles de una imagen digital.

En primer lugar se describe el método tradicional de predicción, que viene dado por una extensión a dos dimensiones del algoritmo de Levinson-Durbin descrito en apartados anteriores, para proponer posteriormente un algoritmo alternativo que permite llevar a cabo los cálculos de una forma más directa, con el consiguiente ahorro de tiempo y coste de almacenamiento.

Este segundo algoritmo resulta de una interpretación multicanal del algoritmo de Burg y permite calcular los coeficientes de predicción lineal 2D y los errores de predicción directamente a partir de los píxeles de la imagen.

4.1.3.1 El problema de la predicción lineal en imágenes digitales.

La situación de partida es la existencia de una cierta imagen digital que, desde el punto de vista matemático, puede verse como una matriz bidimensional donde los elementos son los niveles de gris asociados a cada uno de los píxeles de la imagen (consideramos únicamente imágenes en escala de grises). Cualquier imagen vendrá, por tanto, representada por el conjunto de números:

$$x[f,c];\ \{f,c\} = \{0, 1, \ldots, F-1;\ 0, 1, \ldots, C-1\} \qquad \textbf{(4.36)}$$

donde F y C son, respectivamente, el número de filas y columnas de la imagen.

El objetivo que se persigue es poder predecir el valor de un determinado píxel de la imagen a partir de los valores de otros píxeles de la misma. Para ello será necesario la aplicación de algún algoritmo de predicción, donde la hipótesis de partida es siempre que cualquier píxel puede predecirse a partir del valor de píxeles que le rodean.

Sin embargo, esta suposición inicial no tiene por qué ser cierta con carácter general y se requiere el cumplimiento de algunas condiciones tales como la estacionariedad y la correlación de píxeles. Por esta razón es habitual llevar a cabo una división de la imagen completa en una serie de bloques de menor tamaño, de manera que el algoritmo de predicción se aplicará a cada uno de los bloques por separado, independientemente de los demás. De nuevo, dichos bloques serán matrices bidimensionales:

$$x[m,n];\ \{m,n\} = \{0, 1, \ldots, M-1;\ 0, 1, \ldots, N-1\};\ M \leq F,\ N \leq C \qquad \textbf{(4.37)}$$

donde ahora M y N son respectivamente el número de filas y columnas de un cierto bloque.

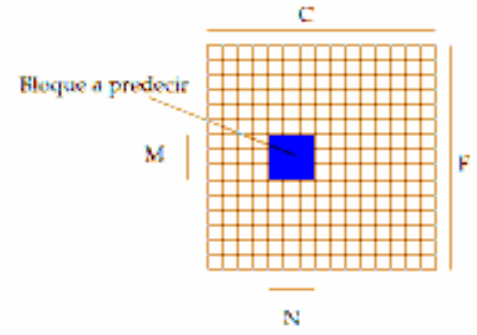

Figura 4.6. Bloque a predecir dentro de una imagen completa.

El tamaño del bloque considerado (M y N) es un parámetro cuyo valor óptimo dependerá de la aplicación concreta que se haga del algoritmo de predicción y de las prestaciones que se requieran del mismo.

En el caso de predicción lineal, el valor predicho para un cierto píxel del bloque se obtiene como una combinación lineal de los valores de otros píxeles del mismo:

$$\hat{x}[m,n] = -\sum_{[i,j]\in ROS}\sum a[i,j]x[m-i,n-j] \qquad \textbf{(4.38)}$$

$$\text{con} \quad \begin{matrix} 0 \le m \le M \\ 0 \le n \le N-1 \end{matrix}$$

donde ROS es la región de suma que se contempla en la predicción y a[i,j] son los coeficientes de predicción lineal 2D. El problema estará resuelto en el momento en que se conozcan los coeficientes a[i,j].

En este caso la ROS será un bloque bidimensional de tamaño (p_1 x p_2)-1, formado por píxeles que preceden en el bloque al píxel que se quiere predecir. Es decir, se considera $0 \le i < p_1$ y $0 \le j < p_2$ (careciendo de sentido el valor a[0,0]). Veamos gráficamente el concepto de ROS dentro de un bloque:

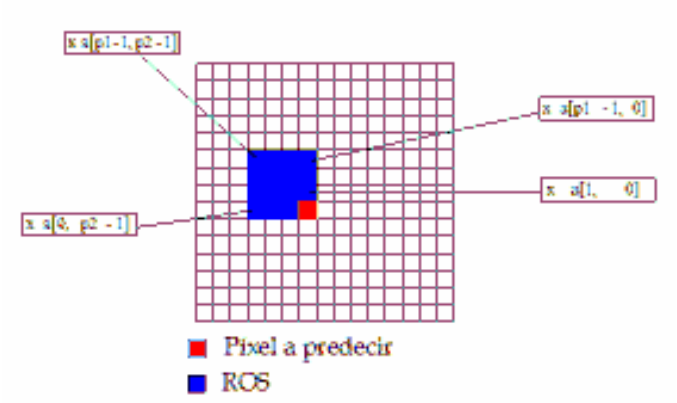

Figura 4.7. Predicción a partir de los píxeles de la ROS.

Se considera *error de predicción lineal* por píxel x[m,n] al valor:

$$e[m,n] = x[m,n] - \hat{x}[m,n] = x[m,n] - \sum_{[i,j]\in ROS}\sum a[i,j]x[m-i,n-j] \qquad \textbf{(4.39)}$$

Los coeficientes de predicción a[i,j] se eligen de manera que se minimice el valor cuadrático medio de dichos errores de predicción (criterio MMSE). La aplicación del criterio MMSE conduce a las llamadas ecuaciones normales, en este caso en 2D:

$$\sum_{[i,j]\in ROS}\sum a[i,j]r_{xx}[k-i,l-j]=\begin{Bmatrix}0\rightarrow[k,1]\in ROS\\ \varepsilon^2\rightarrow[k,1]=[0,0]\end{Bmatrix} \quad \textbf{(4.40)}$$

donde $\varepsilon^2 = E\{e^2[m,n]\}$ y $r_{xx}[k,l]$ es la función autocorrelación 2D para el bloque.

La resolución de las ecuaciones normales proporciona los valores de los coeficientes a[i,j]. Para encontrarlos, en los apartados siguientes se proponen dos métodos: el algoritmo de Levinson-Durbin de dos dimensiones y la versión multicanal del algoritmo de Burg para computación directa de los coeficientes de predicción.

4.1.3.2 El algoritmo de Levinson-Durbin en dos dimensiones.

El algoritmo que aquí se describe no es más que una extensión a dos dimensiones del algoritmo Levinson-Durbin descrito en apartados anteriores y se basa en el cálculo de una cierta matriz de autocorrelaciones.

Las ecuaciones normales para dos dimensiones en formato matricial tienen la siguiente forma $R_{xx}a=\rho$, donde la matriz y los vectores valen:

$$R_{xx}=\begin{bmatrix}R_{xx}[0] & R_{xx}[-1] & \dots & R_{xx}[-p_1+1]\\ R_{xx}[1] & R_{xx}[0] & \dots & R_{xx}[-p_1+2]\\ \dots & \dots & \dots & \dots\\ R_{xx}[p_1-1] & R_{xx}[p_1-2] & \dots & R_{xx}[0]\end{bmatrix}\quad R_{xx}[i]=\begin{bmatrix}r_{xx}[i,0] & \dots & r_{xx}[i,-p_2+1]\\ \dots & \dots & \dots\\ r_{xx}[i,p_2-1] & \dots & r_{xx}[i,0]\end{bmatrix} \quad \textbf{(4.41)}$$

$$a=\begin{bmatrix}a[0]\\ a[1]\\ \dots\\ a[p_1-1]\end{bmatrix},\ a[j]=\begin{bmatrix}a[j,0]\\ a[j,1]\\ \dots\\ a[j,p_2-1]\end{bmatrix},\ \rho=\begin{bmatrix}\varepsilon^2\\ 0\\ \dots\\ 0\end{bmatrix} \quad \textbf{(4.42)}$$

En estas ecuaciones $r_{xx}[i,k]$ es la función autocorrelación de dos dimensiones de la variable bidimensional x, mientras que a[i,l] son los coeficientes que aparecen en la figura 4.9

Una forma de obtener la solución para estas ecuaciones normales es establecer una relación entre la predicción lineal 2D y la predicción lineal multicanal.

4.1.3.2.1 Predicción lineal multicanal.

Se consideran un total de p_2 canales (vectores de señal) numerados con j=0 ,…, p_2-1 de l a forma $x_j[m]$; m=0, 1, …, N_m-1. Lo que se pretende es predecir el valor de dichos canales en un cierto instante m a partir de valores anteriores de los mismos. Es importante resaltar que en este caso la predicción para un cierto canal no se realiza sólo a partir de los valores anteriores de ese mismo canal sino también de todos los demás canales considerados.

Los predictores lineales hacia atrás y hacia delante de orden p_1-1 para una señal multicanal como la que se ha descrito, vienen dados por:

$$\hat{X}[m] = -\sum_{i=1}^{p_1-1} A[i]X[m-i] \rightarrow \textit{predicción lineal hacia delante}$$

$$\hat{X}[m] = -\sum_{i=1}^{p_1-1} B[i]X[m+i] \rightarrow \textit{predicción lineal hacia atrás} \qquad \textbf{(4.43)}$$

donde las matrices A[i] y B[i], i=1, 2, …, p_1-1 son las matrices de coeficientes de predicción multicanal hacia delante y hacia atrás, y donde:

$$X[m]=[x_0[m]\ x_1[m]\ \ldots\ x_{p2-1}[m]]^T \qquad \textbf{(4.44)}$$

son los valores de los distintos canales en el instante m.

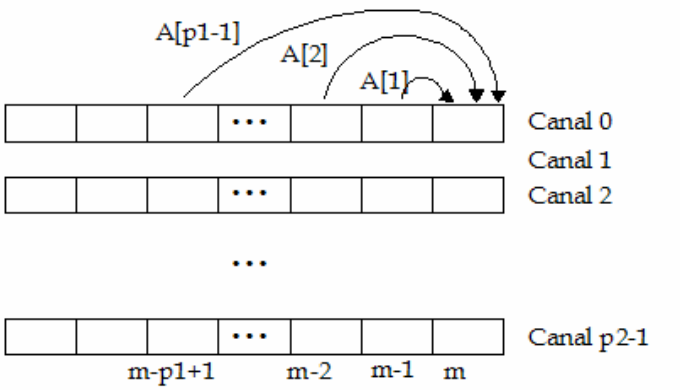

Figura 4.8. Predicción multicanal.

El error de predicción multicanal será, en el instante m:

$$e[m] = X[m] - \hat{X}[m] = X[m] - \sum_{i=1}^{p_1-1} A[i]X[m-i] \qquad \textbf{(4.45)}$$

Las matrices de coeficientes de predicción A[i] que minimizan la traza de la matriz de covarianza del error $P=E\{e[m]e^T[m]\}$, para los procesos multicanal

estacionarios en sentido amplio se obtienen resolviendo las ecuaciones normales $\mathbf{R}_m\mathbf{A}=\mathbf{P}$, donde:

$$R_m=\begin{bmatrix} R_m[0] & R_{xx}[-1] & \dots & R_m[p_1-1] \\ R_m[-1] & R_m[0] & \dots & R_m[p_1-2] \\ \dots & \dots & \dots & \dots \\ R_m[-p_1+1] & R_m[-p_1+2] & \dots & R_m[0] \end{bmatrix} \quad R_m[k]=\begin{bmatrix} r_{0,0}[k] & \dots & r_{0,p_2-1}[k] \\ \dots & \dots & \dots \\ r_{p_2-1,0}[k] & \dots & r_{p_2-1,p_2-1}[k] \end{bmatrix} \tag{4.46}$$

$$r_{ij}[k]=E\{x_i[m]x_j[m-k]\},\ \mathbf{A}=[I\ A[1]\ A[p_1-1]]^T,\ \mathbf{P}=[P\ O\ \dots\ O] \tag{4.47}$$

donde O es una matriz todo-cero de tamaño p_2xp_2.

A la vista queda la enorme similitud entre las ecuaciones normales del problema de predicción lineal 2D mediante el método de Levinson-Durbin y las ecuaciones normales generalas por la predicción multicanal. Esta similitud será aprovechada para realizar la predicción 2D de Levinson-Durbin utilizando herramientas proporcionadas por la predicción multicanal.

4.1.3.2.2 Predicción lineal 2D con herramientas multicanal.

La solución a la versión multicanal de las ecuaciones normales puede utilizarse para obtener los coeficientes de predicción 2D, tan sólo llevando a cabo la siguiente asociación:

$$r_{ij}[k]=r_{xx}[-k,i-j],\ i,j=0,1,\dots,p_2-1 \tag{4.48}$$

y eligiendo

$$\rho=P\ a[0] \tag{4.49}$$

de tal manera que se cumple:

$$a[k]=A^T[k]a[0],\ k=1,2,\dots,p_1-1 \tag{4.50}$$

Denotando la solución de orden n con el sufijo n, indicando con los superíndices f y b la predicción hacia delante y hacia atrás respectivamente, y utilizando la asociación realizada en la ecuación 4.38, se comprueba que:

$$\begin{aligned} P_{p_1}^f &= P_{p_1}^b = P_{p_1} \\ B_{p_1}[k] &= A_{p_1}^T[k];k=1,2,\dots,p_1-1 \end{aligned} \tag{4.51}$$

Una vez realizadas todas las consideraciones, el algoritmo de recursión puede ser modificado para **calcular los coeficientes de predicción 2D** de la siguiente forma:

Para n=0:

1) Inicializar $P_0=R_{xx}[0]$

Para n = 1, 2, …, p_1-1, repetir los pasos 2-4:

2) $A_i[i]=-(R_{xx}[i] + \Sigma A_{i-1}[k]R_{xx}[i-k])P^{-1}{}_{i-1}$

3) $A_i[k]=A_{i-1}[k] + A_i[i]A^T{}_{i-1}[i-k]$, k=1, 2, ..., i-1

4) $P_i=P_{i-1} - A_i[i]P_{i-1}A^T{}_i[i]$

Llegados a este punto ya se dispone de las siguientes dos matrices:

$$A=[I\ A_{p1-1}[1]\ A_{p1-1}[2]\ \dots\ A_{p1-1}[p_1-1]]^T$$

$$P=[P_{p1-1}\ O\ \dots\ O] \qquad \textbf{(4.52)}$$

Con ellas se podrán obtener los coeficientes de predicción 2D de orden p_1-1, calculando en primer lugar a[0] mediante la ecuación 4.49, para posteriormente utilizar la ecuación 4.50. Este algoritmo de cálculo, no obstante, presenta dos **problemas**:

1) En primer lugar, es un método basado en el cálculo de funciones de autocorrelación, necesarias para poder obtener el valor de las distintas matrices A_i. Esto hace que el algoritmo sea poco eficiente computacionalmente hablando, lo cual desemboca en tiempos de ejecución considerablemente largos e importante consumo de recursos de la máquina en la que se esté ejecutando. Además se trata sólo de una *estimación* de la función de autocorrelación.

2) Es necesario llevar a cabo suposiciones sobre el valor de hipotéticos píxeles que se encontrarían rodeando a la imagen real. Esto ocurrirá a la hora de predecir píxeles que se sitúan en zonas cercanas al borde de la imagen.

El valor concreto que se le supondrá a esos píxeles ficticios dependerá de la aplicación, pero en principio existen técnicas típicas como pueden ser la *técnica especular* sobre la imagen, fijar valores extremos (blancos o negros) o extensión de los píxeles que forman el borde.

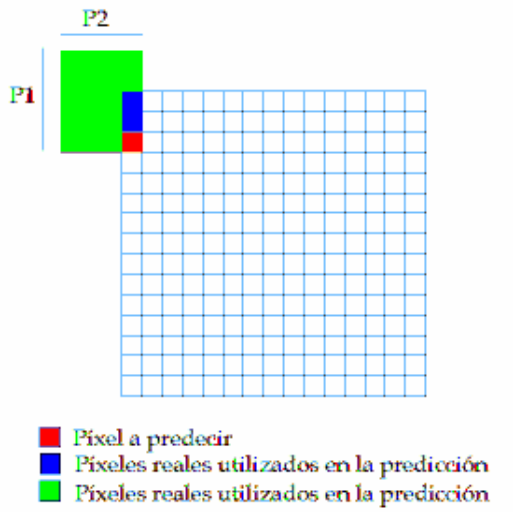

Figura 4.9. Predicción de píxeles situados en el borde de la imagen

4.1.3.2.3 Versión multicanal del algoritmo de Burg para predicción lineal 2D.

Como solución a los dos problemas principales del método descrito en el apartado anterior, describimos en este apartado un nuevo algoritmo propuesto por Rangaraj M. Rangayyan y Gopinath R. Kuduvalli con aplicación directa a la predicción lineal 2D para imágenes digitales (véase referencia 11). Este algoritmo nace a partir de una extensión multicanal del algoritmo de Burg y elimina la dependencia de las matrices A_i respecto de las matrices de autocorrelación $R_{xx}[k]$.

En este caso las matrices A_i originadas por la extensión multicanal, y en última instancia los coeficientes de predicción multicanal, se calculan a partir de los vectores de error de predicción lineal hacia delante y hacia atrás. Además, el proceso de predicción no implicará salir de los límites de la imagen, con lo cual no será necesario dar valores a píxeles ficticios.

Por último, decimos que se trata de un *algoritmo reversible*, es decir, en una primera fase permite obtener los coeficientes y errores de predicción a partir del valor de los píxeles de un cierto bloque de la imagen para, en una segunda fase, poder reconstruir la imagen a partir de dichos coeficientes y errores. A continuación detallamos ambas fases.

FASE 1. Obtención de los coeficientes y errores de predicción a partir de la imagen original.

El punto de partida en este caso es la existencia de un cierto bloque de la imagen de dimensiones M filas y N columnas. Lo que se pretende es calcular los coeficientes y errores de predicción para dicho bloque, considerando regiones de predicción de tamaño p_1 x p_2 tal y como aparece en la figura 4.9.

El primer paso es llevar a cabo un particionado del bloque en una seria de vectores de error. Existen vectores de error de predicción hacia delante (e^f_i) y hacia atrás (e^b_i), donde el subíndice i hace referencia al orden de predicción actual. Dichos valores son de tamaño p_2 y su distribución en el bloque que será predicho es la que aparece en la figura 4.10. En ella, $N_{máx}$ es el número total de vectores de error hacia delante o hacia atrás que caben en el bloque y $\lambda = \frac{N}{p_2}$. Al principio del algoritmo, dichos vectores se inicializan al valor de los píxeles del bloque (e^f_0, e^b_0).

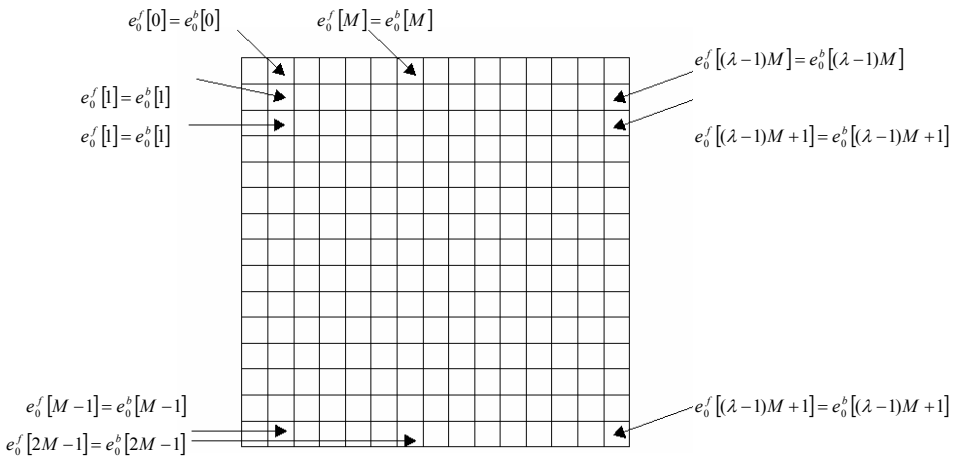

Figura 4.10. Distribución de los vectores de error en el bloque.

Una vez realizada la inicialización, comienza el **proceso de recursión de Burg,** presentando el algoritmo los siguientes pasos:

Paso 1. Repetir para i = 0, 1, p_1-2

1.1 Se calculan las matrices de covarianza del error hacia delante (E^f_i), hacia atrás (E^b_i) y hacia delante y atrás (E^{bf}_i), que son todas las matrices p_2xp_2, de la siguiente forma:

$$E_i^f = \sum_m e_i^f[m]\left(e_i^f[m]\right)^T$$
$$E_i^b = \sum_m e_i^b[m]\left(e_i^b[m]\right)^T \quad \textbf{(4.53)}$$
$$E_i^{bf} = \sum_m e_i^{bf}[m]\left(e_i^{bf}[m]\right)^T$$

1.2 Calcular la matriz de coeficientes de predicción $A_{i+1}[i+1]$ resolviendo la ecuación:

$$E_i^b A_{i+1}[i+1] + A_{i+1}[i+1]\, E_i^f = -2E_i^{bf} \quad \textbf{(4.54)}$$

Nótese que esta ecuación representa un sistema de ecuaciones de la forma AX+XB=C, que puede resolverse utilizando productos de Kronecker.

1.3 Se calculan los vectores de error de predicción hacia delante y hacia detrás de orden inmediatamente superior:

$$e_{i+1}^f[m] = e_i^f[m] + A_{i+1}[i+1]e_i^b[m-1]$$
$$e_{i+1}^b[m] = e_i^b[m] + A_{i+1}^T[i+1]e_i^f[m] \quad \textbf{(4.55)}$$

Una vez finalizado este primer paso, se dispone, pues, de la siguiente información de interés: los vectores de error de predicción hacia delante en el orden deseado p_1-1, así como las matrices de coeficientes de predicción hasta el mismo orden.

Nos disponemos ahora a calcular los coeficientes de predicción a partir de los vectores de error. Para ello, se considera una matriz denominada *matriz de errores de predicción hacia delante de orden cero* ($e_0[m,n]$), de tamaño M x N, formada por los vectores $e^f_{p1-1}[m]$ colocados exactamente en la misma posición en que se encontraban dentro del bloque al principio del paso 1. También se considera la *matriz de errores de predicción hacia atrás de orden cero* ($b_0[m,n]$), que se inicializa con los mismos valores que ($e_0[m,n]$).

Los siguientes pasos del algoritmo son un proceso iterativo de evolución de estas dos matrices de error.

Paso 2. Repetir para j = 0, 1, …, p_2-2

2.1 Se calculan las sumas cuadráticas de las matrices de error:

$$\sigma_{f,j} = \sum_{m=p_1}^{M-1}\sum_{n=j+1}^{N-1}\left|e_j[m,n]\right|^2$$
$$\sigma_{b,j} = \sum_{m=p_1}^{M-1}\sum_{n=j+1}^{N-1}\left|b_j[m,n-1]\right|^2 \qquad \textbf{(4.56)}$$
$$\sigma_{fb,j} = \sum_{m=p_1}^{M-1}\sum_{n=j+1}^{N-1} e_j[m,n]b_j[m,n-1]$$

2.2 Calcular la componente j+1 del vector de coeficientes de predicción de orden 0 (cuyas dimensiones son p_2x1) de la siguiente forma:

$$a[0,j+1] = \frac{-2\sigma_{fb,j}}{\sigma_{f,j}+\sigma_{b,j}} \qquad \textbf{(4.57)}$$

2.3 Calcular las matrices de error hacia delante y hacia atrás de orden inmediatamente superior:

$$b_{j+1}[m,n]=a[0,j+1]+e_j[m,n]+b_j[m,n-1] \text{ con } m=0, 1, \ldots, M-1$$
$$\textbf{(4.58)}$$
$$e_{j+1}[m,n]=e_j[m,n]+a[0,j+1]b_j[m,n-1] \; n=0, 1, \ldots, N-1$$

En estos momentos conocemos la matriz de errores de predicción hacia delante, el vector de coeficientes de orden 0 y la matriz **A**. El objetivo, a continuación, es hallar los coeficientes de predicción 2D del orden deseado (a[1], a[2], …, a[p_1-1]).

Paso 3. Calculamos las matrices $A_{p1-1}[1]$, $A_{p1-1}[2]$, …, $A_{p1-1}[p_1-2]$ mediante la siguiente recursión:

$$A_{i+1}[k]=A_i[k]+A_{i+1}[i+1]A^T_i[i+1-k] \text{ con } k=1, 2, \ldots, i \text{ e } i=1, 2, \ldots, p_1-2 \qquad \textbf{(4.59)}$$

Paso 4. Finalmente, ya es posible obtener los coeficientes de predicción lineal 2D para el orden deseado p_1-1: a[k], k=1, …, p_1-1. Para ello, basta con resolver el siguiente conjunto de ecuaciones:

$$a[k]=A^T_{p1-1}[k]a[0]; \; k=1, 2, \ldots, p_1-1 \qquad \textbf{(4.60)}$$

Llegados a este punto finaliza el algoritmo y se obtiene el resultado buscado: los coeficientes de error de predicción a[k], k=1, …, p_1-1 y la matriz de errores de predicción hacia delante. No obstante, hacemos a continuación unas **consideraciones de interés sobre el algoritmo** anteriormente descrito.

- En el paso 1, a la hora de calcular las matrices de covarianza de los errores de predicción hacia delante y hacia atrás, quedarán fuera del sumatorio aquellos vectores de error cuyo cálculo mediante los filtros de predicción implique salir fuera de los límites del bloque para un determinado orden i. Por este motivo, no se considerarán los vectores de error hacia atrás ubicados en la última fila, ni los vectores de error hacia delante situados en las i+1 primeras filas para dicho orden de predicción i.

- De la misma forma en las ecuaciones 4.55, para calcular los errores de predicción de un orden superior, el índice no se moverá en todo el rango posible excepto en aquellos valores donde el cálculo de los vectores de error implique salirse de los límites del bloque. Por esa razón, de nuevo no se consideran en estas ecuaciones los vectores de error que ocupan la última fila del mismo, ni los vectores de error de hacia delante que ocupan las i+1 primeras filas para el orden i.

- Como ya se ha indicado anteriormente, los vectores de error hacia delante y hacia atrás que se encuentran situados más a la derecha del bloque que se va a predecir, puede que no tengan una longitud p_2 (dependerá de que el número de columnas del bloque sea o no múltiplo entero de dicho valor). Sin embargo, a lo largo del algoritmo los vectores de error se han considerado todos del tamaño p_2 (de hecho han post-multiplicado a las matrices A_i que son de tamaño p_2x p_2). La forma de solucionar este problema está íntimamente ligada a la implementación computacional del algoritmo. Los vectores de error $e^f_i[m]$ y $e^b_i[m]$ deben situarse sobre el bloque tal y como indica la figura 4.10. Sin embargo, ¿qué ocurre si el número de columnas del bloque no es múltiplo entero de p_2?

Para solucionar este problema se define *tamb* como el mínimo múltiplo entero de p_2 mayor o igual que N, y se consideran como norma general bloques de la imagen de tamaño *tamb x tamb*. De esta forma,

considerando bloques de anchura *tamb* se podrá encajar un número entero de vectores $e^f_i[m]$ y $e^b_i[m]$ en cada fila.

Ahora bien, para no superar los límites de la imagen original, algunos bloques pueden que no lleguen a tener tamaño *tamb x tamb* sino inferior, y puede ocurrir que el número de columnas del mismo ya no sea múltiplo de p_2. ¿Cómo se sitúan entonces los vectores que hemos puesto antes $e^f_i[m]$ y $e^b_i[m]$?

La solución a este problema es utilizar los vectores $e^{fR}_i[m]$ y $e^{bR}_i[m]$ (vectores de error residual hacia delante y hacia atrás). En este caso lo que se hace es utilizar unas matrices de error $e_j[m,n]$ y $b_j[m,n]$ igualmente de tamaño *tamb x tamb* (con lo que podremos situar sobre ellas los vectores $e^f_i[m]$ y $e^b_i[m]$). No obstante, no se tomarán en consideración los vectores $e^f_i[m]$ y $e^b_i[m]$ que no caigan íntegramente del bloque real del tamaño inferior a *tamb,* de forma que los píxeles que queden como residuo (no incluidos en los $e^f_i[m]$ y $e^b_i[m]$ que se tengan en consideración) se incluirán en los vectores $e^{fR}_i[m]$ y $e^{bR}_i[m]$. El algoritmo de predicción se aplicará sobre los vectores $e^f_i[m]$ y $e^b_i[m]$, así como en los vectores $e^{fR}_i[m]$ y $e^{bR}_i[m]$, y al final del mismo las matrices de error tomarán valores de $e^f_i[m]$ y $e^b_i[m]$ o bien de $e^{fR}_i[m]$ y $e^{bR}_i[m]$, según donde corresponda.

El algoritmo de predicción se aplica tanto a los vectores $e^f_i[m]$ como a los $e^{fR}_i[m]$, sin embargo, sólo se considerarán en este caso concreto los vectores $e^{fR}_i[m]$ que caen dentro de las seis primeras columnas, con lo cual sobran las dos últimas, que serán tratadas a través de los vectores $e^{fR}_i[m]$. Al final del algoritmo, la matriz de errores $e_j[m,n]$ la formará los vectores $e^f_i[m]$ de esas seis primeras columnas colocadas en su posición correspondiente y las dos últimas columnas de los vectores $e^{fR}_i[m]$ (también en su posición adecuada).

De esta forma es como se hace la división de la imagen en bloques y como se soluciona el problema del tamaño de los mismos.

•Los pasos 3 y 4 del algoritmo serán necesarios sólo si se desea conocer expresamente el valor de todos los coeficientes de predicción a[k] para, por ejemplo, calcular los valores predichos según la ecuación 4.38. Pero si el objetivo es tan solo obtener la información necesaria para reconstruir el

bloque original, estos dos pasos pueden ahorrarse ya que tal y como se verán en el apartado siguiente, utilizando la versión multicanal del algoritmo de Burg basta con conocer la matriz de errores de predicción hacia delante y el vector de coeficientes de predicción de orden 0, así como las matrices $A_i[i]$. Además, gracias a ello, no será necesario hacer suposiciones sobre valores de hipotéticos píxeles que rodeen al bloque original, ya que no se aplica directamente la ecuación 4.39.

FASE 2. Reconstrucción de la imagen a partir de los coeficientes y errores de predicción.

La versión multicanal del algoritmo de Burg para la predicción lineal 2D R. M. Rangayyan y G. R. Kuduvalli (véase referencia 11) es de carácter reversible, es decir, una vez ejecutado el algoritmo descrito es posible realizar un proceso inverso para, a partir de los coeficientes y errores de predicción, poder reconstruir la imagen original (en realidad el bloque original, ya que este algoritmo de reconstrucción también se aplica sobre cada bloque de la imagen independientemente de los demás).

El punto de partida para el algoritmo consiste en conocer el vector de coeficientes de predicción de orden 0 a[0] (no se necesitan los coeficientes de orden superior), la matriz de errores de predicción hacia delante $e_{p_2-1}[m,n]$ y las matrices de coeficientes de predicción $A=[I\ A_1[1]\ \ldots\ A_{p1\text{-}1}[p_1\text{-}1]]^T$.

También en este caso existen los vectores de error de predicción hacia delante (e^f_i) y hacia atrás (e^b_i), donde de nuevo la i indica el orden de predicción actual, los cuales están ordenados en el bloque que deseamos reconstruir tal y como aparece en la figura 4.10. Todos estos vectores de error ordenados pueden verse como dos matrices de errores de predicción hacia delante y hacia atrás que se denominarán errf[m,n] y errb[m,n], siendo m=0,1,…,M-1 y n=0,1,…,N-1.

El algoritmo es un proceso iterativo que hace converger los errores de predicción a los valores reales de los píxeles que formaban el bloque original. Los pasos que presenta dicho algoritmo son los siguientes:

Paso 1. Inicializar los errores asociados a la primera columna del bloque en cuestión (errf[i,0] y errb[i,0] con i=0, …, M-1) con los errores de la primera columna $e_{p_2-1}[m,n]$.

Paso 2. Predecir los errores de las siguientes p_2-2 columnas de errf[m,n] y errb[m,n] e ir actualizando el vector de coeficientes de predicción de orden cero, a[0], de la siguiente forma:

Repetir para j=1, 2, …, p_2-2

2.1 $errf[i,j] = errb[i,j] = e_{p_2-1}[i,j] - \sum_{k=1}^{j} a[0,k]\,errf[i,j-k]$ con i=0,…,M-1 **(4.61)**

El resultado de esta operación es un número decimal que será redondeado al entero más cercano.

2.2 l=j, k=1

2.3 auxiliar = a[0,k]
a[0,k]=a[0,k]+a[0,j+1]a[0,1] **(4.62)**
a[0,1]=a[0,1]+a[0,j+1] · auxiliar

2.4 Incrementar k, decrementar l.

2.5 Volver a 2.3 si k<1.

2.6 Si k=1, entonces: a[0,k]=a[0,k]+a[0,j+1]a[0,1] **(4.63)**

Paso 3. Una vez obtenidos los errores de p_2-1 primeras columnas, predecir el resto de los valores de las matrices de error de la siguiente forma:

$$errf[i,j] = errb[i,j] = e_{p_2 1}[i,j] - \sum_{k=1}^{p_2-1} a[0,k]\,errf[i,j-k] \quad \textbf{(4.64)}$$

con i=0, …, M-1 y j=p_2-1, …, N-1

Nuevamente, si el resultado de esta operación resultase un número decimal, redondearíamos al entero más próximo.

Llegados a este punto ya tenemos inicializadas por completo las matrices de error de predicción errf[m,n] y errb[m,n]. Gráficamente, podemos observar el proceso seguido en los pasos 1, 2 y 3 en la siguiente figura:

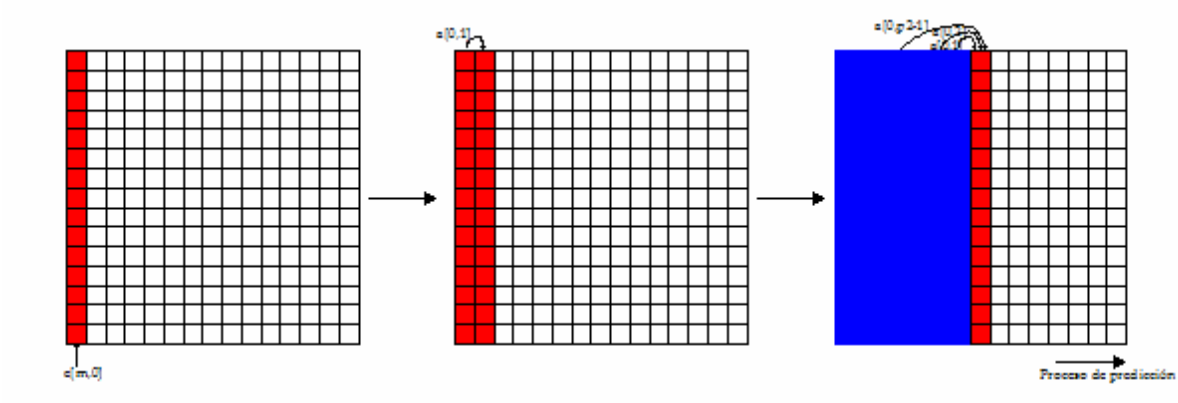

Figura 4.11. Predicción horizontal en el bloque.

Paso 4. En este paso haremos converger los errores de predicción hacia atrás a los valores iniciales de los píxeles en el bloque original. En primer lugar, se busca la correspondencia de las p_1-1 primeras filas de la matriz de error hacia atrás al mismo tiempo que se van calculando las diferentes matrices $A_i[j]$.

Repetir para j=1, 2, …, p_1-1

4.1 $$e_j^b[l] = e_0^f[l] - \sum_{k=1}^{j} A_j[k] e_{j-l}^b[l-k] \qquad \textbf{(4.65)}$$

En este punto, l toma los valores de manera que se consideran todos los vectores $e^b_j[l]$ situados en la fila j. Al igual que en apartados anteriores, las componentes de este vector se redondearán al valor entero más cercano.

4.2 Se calculan las matrices $A_{j+1}[k]$, k=1, …, j mediante:

$$A_{j+1}[k] = A_j[k] + A_{j+1}[j+1]\, A_j^T[j+1-k] \quad k=1, \ldots, j \qquad \textbf{(4.66)}$$

Paso 5. Finalmente se hace converger hacia los valores de los píxeles del bloque a los errores de predicción hacia atrás de la matriz de errores. Para ello, se trata de repetir para j=1, 2, …, p_1-1 el siguiente conjunto de ecuaciones:

$$e_j^b[l] = e_0^f[l] - \sum_{k=1}^{p_1-1} A_{p_1-1}[k] e_{j-l}^b[l-k] \qquad \textbf{(4.67)}$$

En este caso l se incrementa de uno en uno pero no toma aquellos valores que impliquen vectores de error $e^b_j[l]$ situados en la p_1-1 primeras filas del bloque.

Una vez finalizado el algoritmo la matriz errb[m,n] almacenará los píxeles del bloque que se quería reconstruir. Sobre el **algoritmo propuesto,** hacemos ciertas **consideraciones:**

•Se observa claramente cómo el proceso de convergencia (predicción) de los píxeles se lleva a cabo a partir de un rectángulo de dimensiones p_1xp_2. En primer lugar, y en el paso 3, se realiza un proceso de predicción de cada píxel a partir de los p_2-1 situados en su misma fila a la izquierda para, posteriormente, en el paso 5, continuar con la predicción de cada píxel pero esta vez a partir de los p_1-1 píxeles situados encima de él, en su misma columna. En otras palabras, existe un proceso de predicción horizontal de izquierda a derecha, seguido de un proceso vertical de arriba a abajo dentro del bloque.

•En este proceso de predicción son de especial interés aquellos píxeles situados dentro del bloque en los puntos que, o bien no tienen p_2-1 píxeles a su izquierda o bien no tienen p_1-1 píxeles por encima o ambas cosas. Precisamente para trabajar con este tipo de píxeles existen los puntos 2 y 4 del algoritmo. Nótese que en el punto 2 de cada píxel es actualizado a partir del valor de tantos píxeles como tenga situados a su izquierda, sin llegar a p_2-1 píxeles y sin salirse del bloque. He aquí una de las principales ventajas de este algoritmo de reconstrucción: se ha podido llevar a cabo la predicción sin necesidad de dar valores a posibles píxeles ficticios que se encontrarían rodeando a la imagen.

•De nuevo aparece aquí el problema de que los vectores de error situados más a la derecha en el bloque pueden llegar o no a tener longitud p_2, mientras que a lo largo del algoritmo, todos los vectores de error se están considerando de esa longitud.

4.2 CÁLCULO DEL PARÁMETRO TR

Hasta ahora hemos utilizado el algoritmo de predicción lineal para obtener unos puntos, *pre-candidatos,* como posibles microcalcificaciones. No obstante, se ha implementado otro parámetro, el TR (del inglés *tail ratio*), que servirá para discriminar muchos de estos píxeles. Conseguiremos, pues, quedarnos con un conjunto mucho más reducido de puntos a los que de ahora en adelante llamaremos *candidatos*. Este apartado ha sido obtenido del artículo *Detection of microcalcifications in mammograms* escrito por B. Acha, C. Serrano, R. M. Rangayyan y J. E. Desautels (véase referencia 14).

Las microcalcificaciones representan pequeños puntos de gran intensidad. Consecuentemente, si analizamos la función de densidad de probabilidad (PDF) de los píxeles situados en los alrededores de una microcalcificación, tendrá la cola derecha más larga que la izquierda. Basándonos en este hecho, se ha desarrollado un parámetro, *TR*, relativo a la medida de la longitud de una cola de una PDF. De hecho, existen ya algunos descriptores que miden el peso de una cola de una PDF. Por ejemplo, el **coeficiente de kurtosis** es frecuentemente considerado como una medida del peso de una distribución relativa a una distribución normal. Su interpretación y uso están restringidos a distribuciones simétricas debido a su comparación intrínseca con la distribución simétrica normal. Otro inconveniente de utilizar este coeficiente es que es muy sensible a la dispersión en los datos, ya que está basado en momentos de éstos. Para cualquier función de distribución, *F*, con un número finito de momentos, el coeficiente kurtosis se define como:

$$k = \frac{E_F\left(X - E_F(X)\right)^4}{\left\{E_F\left(X - E_F(X)\right)^2\right\}^2} \qquad \textbf{(4.68)}$$

donde numerador y denominador representan al segundo y cuarto momento central de F respectivamente, y *X* representa los valores de los datos. Ciertos autores han presentado medidas más robustas de kurtosis, pero definidas sólo para las distribuciones simétricas o simplemente midiendo los picos en vez del peso de la cola.

Otros estudios solucionan los problemas mencionados anteriormente introduciendo algunas medidas del peso de la cola para distribuciones continuas que pueden ser aplicadas tanto a distribuciones simétricas como a distribuciones asimétricas. Definen medidas de cola derecha e izquierda como medidas de torsión que se aplican sobre una parte de masa de probabilidad a derecha o izquierda, respectivamente, de la mediana de F, que se denota como:

$$m_F = F^{-1}(0.5) \tag{4.69}$$

Sin embargo, no hemos encontrado que estas medidas sean capaces de solucionar el problema de la detección de microcalcificaciones porque son una estimación del peso de la cola pero no de su longitud. Las microcalcificaciones, como ya describimos antes, se caracterizan por tener una cola derecha larga en el histograma local. Por este motivo y con el propósito de caracterizar la cola derecha, hemos desarrollado el *TR*, definido como:

$$TR = \frac{x_{\max} - F^{-1}(0.5)}{F^{-1}(0.5) - x_{\min}} \tag{4.70}$$

donde $x_{\max}$ y $x_{\min}$ representan los valores máximos y mínimos de intensidad de la PDF y F^{-1} es la inversa de la función de distribución de probabilidad.

Debido a que el tamaño de las microcalcificaciones es variable, el número de píxeles situados alrededor de los pre-candidatos usados para calcular el histograma local debe ser adaptativo. De acuerdo con el tamaño de una microcalcificación, cuyo diámetro varía entre 0.1 y 1 mm, y de acuerdo con la resolución de las imágenes de la base de datos utilizada, las microcalcificaciones pueden ocupar desde 4 a 400 píxeles cada una. Por consiguiente, calculamos inicialmente el histograma y el parámetro TR para 3x3 píxeles cuadrados alrededor de cada pre-candidato. Si el *TR* es mayor que el umbral aplicado, el pre-candidato pasa a ser considerado como candidato para los sucesivos pasos de la detección. En otro caso, se incrementa el tamaño del área circundante y se recalcula el *TR*. El procedimiento se repite hasta que una de estas dos condiciones se satisfaga:

1. La caja de selección alcanza su máxima área (especificada como 20×20 píxeles en el presente proyecto).

2. El *TR* es mayor que el umbral aplicado.

4.3 IMPLEMENTACIÓN DE UNA APLICACIÓN EN C

4.3.1 Diagrama de la aplicación.

En esta sección explicaremos con un cierto grado de profundidad la estructura del programa en C que hemos realizado.

En primer lugar presentamos un diagrama del programa que nos servirá como guía en el transcurso del capítulo.

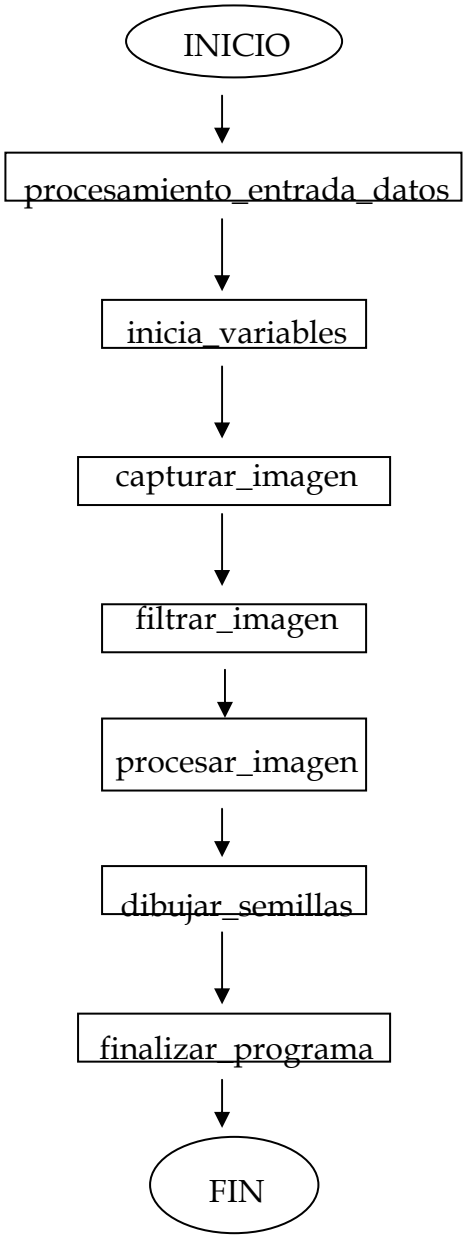

Figura 4.12. Diagrama de flujos del programa

Esta organización del código resulta totalmente novedosa con respecto a la versión anterior. Se ha pretendido realizar un programa con una idea siempre en mente: que en un futuro se pueda ampliar y mejorar muy fácilmente.

Pasemos, sin más preámbulos, a describir someramente cada una de las funciones llevadas a cabo.

4.3.2 Procesamiento de la entrada de datos.

Esta función tiene dos misiones claramente diferenciadas. Por una parte comprueba que se haya llamado correctamente al programa pasándole los parámetros adecuados. Asimismo, se encarga de abrir los ficheros en los que almacenaremos los resultados.

4.3.2.1 Parámetros del programa.

La llamada al programa debe hacerse escribiendo la siguiente línea de comando en el *símbolo de sistema* en el directorio en el que se encuentre el ejecutable: **semillas.exe nombre_imagen.lum**. Es importante resaltar que las extensiones de los ficheros deben ser escritas ya que el programa se encarga de generar los nombres de los ficheros de salida ignorando los cuatro últimos caracteres del nombre de la imagen que se desee procesar.

Cualquier tipo de llamada que sea distinta a ésta provocará una salida del programa.

4.3.2.2 Ficheros generados.

Son tres los ficheros que vamos a obtener tras la ejecución de nuestra aplicación:

1. En **sem_nombre_imagen.m** se almacenarán las coordenadas de los píxeles semillas obtenidos. Será en formato MatLAB ya que este fichero será utilizado posteriormente para un post-procesado con el fin de eliminar falsos candidatos.

2. El fichero **nombre_imagen_sem.bmp** es la imagen original con las semillas, almacenados en formato BMP.

3. Finalmente, **resultados_nombre_imagen.log** es un fichero que recopila los diversos resultados intermedios que se van obteniendo en el transcurso del programa. Aunque su existencia no aporta nada en cuanto al procesado de la imagen, resulta de una gran utilidad a la hora de depurar los posibles errores de código que se cometan en un futuro. Como dato significativo, incluye el tiempo total invertido por el programa para realizar el análisis.

4.3.3 Inicialización de las variables.

Al igual que ocurre en el apartado anterior, tenemos aquí dos tareas claramente diferenciadas.

Por un lado, agrupamos todas las reservas dinámicas realizadas. La razón de que no sean reservas estáticas radica en la existencia de una serie de parámetros cuyo ajuste permite que la sensibilidad del programa para detectar microcalcificaciones varíe.

Como segunda tarea rellenamos determinadas matrices con unos valores iniciales que serán fijos en cualquier ejecución. La elección de estos parámetros son fruto, en su mayoría, de una amplia experiencia empírica.

Se ha tratado de seguir un orden coherente, agrupando las distintas variables en bloques afines. Nos limitaremos a citar el nombre de los bloques creados:

1. Reserva e inicialización de los vectores y matrices de error.

2. Reserva e inicialización de las matrices A y A^t.

3. Reserva e inicialización del vector de coeficientes de predicción.

4. Reserva de memoria para las matrices de covarianza.

5. Reserva de memoria para los vectores y matrices auxiliares.

6. Reserva de memoria para las matrices de Kronecker.

4.3.4 Captura de la imagen.

Hasta ahora nos hemos estado limitando a sentar las bases. Es en este apartado cuando comienza, verdaderamente, a desarrollarse la aplicación.

El proceso que vamos a seguir es sencillo: iremos recorriendo la imagen obteniendo los parámetros de interés para nosotros. Así, nos posicionamos en el byte 806 de la imagen original (formato LUM) que es donde se encuentra el tamaño de la imagen; más concretamente, el número de filas y de columnas.

Seguidamente, nos colocamos en el byte 2048, posición donde comienza el listado de píxeles, los leemos y los almacenamos en una tabla (en la variable global *imagen_no_cod*). Cada elemento de la matriz resultante representará un píxel. Aunque una explicación más amplia del formato LUM se dio en el capítulo anterior, recordemos que cada píxel consta de 12 bits, por lo que se ha optado por coger una tipo de variable *short int* (16 bits).

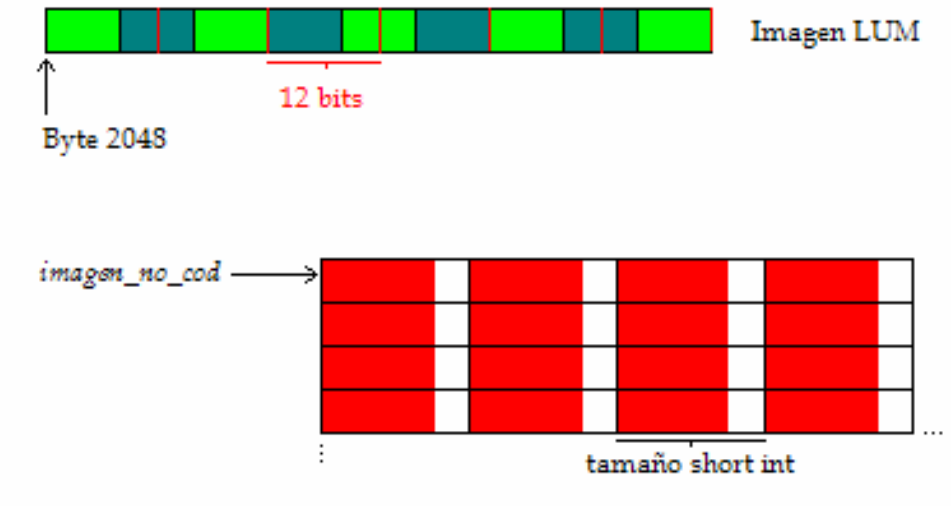

Figura 4.13. Captura de la imagen.

4.3.5 Filtrado de la imagen.

Quizá sea esta la función clave del programa. Un correcto filtrado de la imagen implicará una detección adecuada y por eso hemos dedicado la mayor parte de nuestro esfuerzo a elaborar un algoritmo óptimo. Varias han sido las opciones que hemos barajado en esta función. Uno de los principales problemas de los que partíamos radicaba en el excesivo número de píxeles candidatos que teníamos en un principio.

1. Para solventar este hecho probamos, en primera instancia, a eliminar el fondo negro de la imagen. En concreto, íbamos calculando la media parcial de la mamografía y cuando detectábamos un píxel con una desviación muy elevada, lo sobrescribíamos con el valor medio acumulado. Sin embargo, tras realizar estas operaciones nos dimos cuenta que conllevaba una alta carga computacional y que los resultados no eran tan buenos como los esperados. Por tanto, descartamos esta opción.

2. A continuación, pensamos en recortar la imagen utilizando un filtro paso de alta. Implementamos un filtro que daba unos resultados bastante aceptables y que detectaba los bordes de la mama. No obstante la precisión obtenida no era suficiente ya que para eliminar la parte inservible de la imagen tuvimos que hacer unos artificios del todo ineficientes. Una vez más, por su ineficacia, decidimos no adoptar esta solución.

3. Finalmente, probamos a declarar un umbral (*UMBRALMAMA*) y a calcular la media de la mamografía sólo con los píxeles que estaban por debajo de dicho umbral. La elección de éste ha sido absolutamente empírico y su ajuste nos ha proporcionado unos resultados tan buenos como inesperados. Tras analizar todas las imágenes disponibles en nuestra base de datos, hemos decidido tomar el valor de 2000. Recordemos que nuestro píxeles son de 12 bits y que por tanto su valor oscila entre 2^0 y 2^{12}-1, esto es, desde 0 a 4095.

El valor medio final de la imagen será almacenado en una variable global de nombre *medfil* y será el valor con el que compararemos en el algoritmo de detección de semillas.

4.3.6 Procesado de la imagen.

Nos disponemos ahora a aplicar el algoritmo de detección de semillas que hemos escogido para la realización de nuestro proyecto. Aunque un análisis bastante amplio y detallado ha sido hecho en los apartados 4.1 y 4.2, haremos un pequeño resumen basándonos concretamente en el código que hemos hecho.

En primer lugar, vamos dividiendo la imagen en bloques cuadrados de tamaño *tamb x tamb* píxeles (siendo *tamb* otra variable global que hemos fijado para un óptimo funcionamiento del programa).

A cada bloque le aplicamos el algoritmo de predicción lineal y, mediante la función *detecta_calcificaciones,* obtenemos los puntos de interés para nosotros. Será dentro de esta función donde calculemos el parámetro TR y obtengamos los píxeles candidatos que serán utilizados posteriormente en los programas realizados en el entorno MatLAB.

Una vez recorrida la imagen, tendremos una tabla, *sem,* de dimensiones (*microcalcificaciones, 2)* que almacena las coordenadas x e y pertenecientes a la imagen de los puntos candidatos.

4.3.7 Dibujar las semillas.

Aunque teóricamente podríamos acabar el programa en el apartado anterior, resulta de vital importancia poder tener los resultados en un formato típico de imagen y que todos los visores sean capaces de leer (a diferencia del formato LUM). Por este motivo, aquí nos centraremos en generar un fichero de tipo BMP en el que aparezcan las semillas dibujadas sobre la mamografía.

Tendremos que solventar un problema para hacer la conversión de un formato a otro. Dijimos que los píxeles del formato LUM eran de 12 bits. Nosotros vamos a generar un BMP de 24 bits. Aparentemente, tenemos más precisión pero esto es del todo incierto. Los 24 bits son, en realidad, tres bytes equivalentes al sistema RGB; pero al tratarse de imágenes en blanco y negro, el formato BMP obliga a que estos tres bytes tengan el mismo valor. Por tanto, pasamos de una precisión de 12 bits a una de 8. Para hacer esto, despreciamos los 4 bits menos significativos de la imagen original. Esta pérdida de precisión no resulta importante, ya que la sufrimos sólo al representar la imagen y no durante los cálculos.

4.3.8 Finalización del programa.

Nos encargamos, finalmente, de liberar las variables que fueron declarados dinámicamente y de cerrar los tres ficheros resultantes.