
VNIVERSIDAD NACIONAL AVTONOMA DE MEXICO

UNIVERSIDAD NACIONAL
AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

# ESQUEMA MULTIESCALA DE RELAJACIÓN ESTOCÁSTICA

# T E S I S

QUE PARA OBTENER EL TÍTULO DE:

**INGENIERO EN TELECOMUNICACIONES**

P R E S E N T A:

**ANGEL AUGUSTO TAMARIZ SÁNCHEZ**

DIRECTOR: DR. MIGUEL MOCTEZUMA

2007

*“Si la vida no es la realización de un proyecto, la inteligencia se convierte en una función puramente mecánica”*

Ortega y Gasset

# contenido

# 1. Introducción

## 1.1. Objetivo

Investigar sobre el análisis de texturas y técnicas de segmentación. Con fundamentos probabilísticos, definir un esquema estocástico para la segmentación de imágenes de percepción remota con una velocidad de procesamiento mayor a los esquemas convencionales.

## 1.2. Definición del Problema

La fotografía aérea de alta resolución permite realizar análisis detallados de estructuras urbanas y de elementos temáticos. Nuevos retos deben ser afrontados y nuevas técnicas deben ser planteadas a fin de resolver problemas de la alta definición como lo son la aparición de sombras. La información parcial proporcionada por las imágenes pancromáticas no permite una discriminación inmediatamente precisa sobre las escenas urbanas. Las áreas verdes por lo general comparten un rango similar en niveles de gris sobre otros elementos de las escenas. Por ello es difícil, a través de estas imágenes, distinguir entre los diversos usos de suelo y poder estudiar su evolución a través del tiempo. Sin embargo, segmentando imágenes de percepción remota, se aíslan las áreas de interés para el mejor estudio de terrenos determinados. Las tecnologías contemporáneas de procesamiento digital de imágenes y de percepción remota se sitúan como fuentes de datos útiles a los sistemas de información ambiental, ya que las conclusiones a las que se pueden llegar a partir de una fotografía original pueden llegar a ser incluso subjetivas.

## 1.3. Esquema de Análisis

Es en este contexto que en el presente tema de tesis, se pretende el estudio de técnicas bayesianas y estocásticas para la segmentación de imágenes de percepción remota. En datos de alta resolución, un problema a resolver consiste en la definición de un modelo rápido de procesamiento. En base a un esquema definido en tres niveles de resolución espacial, se plantea en esta tesis:

a) la definición de un esquema máximo a posteriori, que aplique criterios de procesos aleatorios en la segmentación de una representación en baja resolución,
b) la proyección de los resultados en el siguiente nivel de resolución, y
c) la repetición de los pasos anteriores hasta llegar al nivel de resolución original.

El sistema propuesto contempla la aplicación de un solo esquema de recocido simulado, en donde el proceso inicia en el nivel de baja resolución (en donde se aplicarán alrededor de 10 iteraciones), sus resultados se proyectarán en el siguiente nivel en donde continuarán las segmentaciones (con alrededor de 10 iteraciones), para finalmente proyectar los resultados intermedios en el nivel de alta resolución (donde se aplicarán alrededor de 20 iteraciones).

## 1.4. Propuesta de Solución

En el esquema propuesto, el algoritmo deberá tener un buen desempeño y lograr un rápido procesamiento en imágenes de gran tamaño. El nivel de baja resolución, por su tamaño, requerirá de un tiempo de procesamiento reducido, además de los efectos paso bajas que deberá eliminar parte del ruido de la imagen original. El procesamiento en el nivel de resolución original busca detectar en mejor forma las regiones de transición, es decir, las fronteras entre las distintas clases. El procesamiento por teoría de Campos de Markov deberá permitir la detección precisa de contornos. En ambos casos, el fundamento teórico reside en la teoría de Bayes y en la maximización de probabilidades a posteriori. Un estudio deberá emprenderse a fin de determinar la naturaleza del modelo de energía a emplear. Como aplicación, se pretende la segmentación de regiones urbanas del Distrito Federal y el empleo de imágenes de radar de apertura sintética.

## 1.5. Estructura de la Tesis

La tesis se divide básicamente en tres partes. La primera, distribuida entre los capítulos 2, 3, 4 y 5, presenta una breve descripción de la teoría probabilística empleada en el trabajo. En éstos se asume que el lector tiene conocimientos básicos de probabilidad que permitan el fácil entendimiento de los conceptos planteados ya que no se discuten fundamentos básicos a fondo. Los principales temas a discutir en estos cuatro capítulos son:

- Variables Aleatorias
- Teorema de Bayes
- Procesos Estocásticos
- Campos Aleatorios de Markov
- Recocido Simulado

Son éstas las herramientas básicas empleadas para la elaboración del algoritmo.

La segunda parte se desarrolla a lo largo del capítulo 6 (Esquema de Análisis), en el que se da la descripción de las tareas que el algoritmo realiza para alcanzar el objetivo planteado. Con un seguimiento detallado de los pasos a seguir, se utilizan los conocimientos presentados en los capítulos anteriores en conjunto con fundamentos de álgebra, estadística, estructura de datos, análisis de señales aleatorias y procesamiento digital de imágenes para dar forma al algoritmo final.

La tercera y última parte presenta una serie de resultados obtenidos en los distintos experimentos y el análisis de los mismos. Una vez mostrados éstos y analizados en conjunto, se crea una conclusión general que permite reflexionar sobre los logros alcanzados y los posibles retos a futuro en esta área.

El algoritmo fue programado utilizando MATLAB y su código se muestra al final de la tesis, en el apéndice.

# 2. Probabilidad

Antes de comenzar con la descripción formal de algunas definiciones de probabilidad, se resumirá brevemente la notación de conjuntos de la que se hará uso a lo largo de esta tesis.

Supóngase un *conjunto* S que consiste en los puntos identificados con 1, 2, 3 y 4. Esto se denota con S = {1, 2, 3, 4}. Si A = {1, 2} y B = {2, 3, 4}, entonces A y B son *subconjuntos* de S, lo cual se indica mediante A ⊂ S y B ⊂ S (B "está en" o "está contenido en" S). El hecho de que 2 *es un elemento* de A se denota con 2 ∈ A. La *unión* de A y B es el conjunto que consiste en todos los puntos que están en A, en B, o en ambos. Esto se representa mediante A ∪ B = {1, 2, 3, 4}. Si C = {4}, entonces A ∪ C = {1, 2, 4}. La *intersección* de dos conjuntos A y B es el conjunto que consta de todos los puntos que están tanto en A como en B y se representan mediante A ∩ B = AB = {2}. Por otra parte, AC = ∅, donde ∅ denota el conjunto vacío, que es el conjunto que no contiene puntos.

El *complemento* de A con respecto a S es el conjunto de todos los puntos en S que no están en A y se representa mediante $\bar{A}$. Para los conjuntos específicos anteriores $\bar{A}$ = {3, 4}. Se dice que dos conjuntos son *mutuamente excluyentes,* o, *disjuntos*, si no tienen puntos en común, como los conjuntos A y C anteriores.

## 2.1 Axiomas de Probabilidad

Una vez esclarecida la notación a emplear, podemos dar comienzo a la primera serie de definiciones.

Supóngase que un experimento tiene asociado un espacio muestral S. Una probabilidad es una función de valor numérico que asigna un número P(A) a cada evento A, de tal manera que son válidos los siguientes axiomas:

1) $P(A) \geq 0$

2) $P(S) = 1$

3) Si $A_1, A_2, \ldots$, es una sucesión de eventos mutuamente excluyentes, es decir, $A_i A_j = \varnothing$ para toda $i \neq j$, entonces

$$P\left[\bigcup_{i=1}^{\infty} A_i\right] = \sum_{i=1}^{\infty} P(A_i) \qquad (2.1\text{-}1)$$

El primero únicamente representa nuestro deseo de trabajar con números positivos. El axioma (2) reconoce que el espacio muestral en sí es un evento, y que, dado que es el evento que abarca a todos los demás, debería tener la probabilidad más alta, que es definida como la unidad. Por esta razón, S es conocido como el *evento seguro*. Alternamente, el conjunto nulo ∅ es un evento sin elementos, es conocido como el *evento imposible* y con probabilidad 0.

De acuerdo con el axioma (3) se llega a la conclusión de que si A y B son eventos mutuamente excluyentes,

$$P(A \cup B) = P(A) + P(B)$$

Es fácil ver ahora que si A ⊂ B entonces P(A) ≤ P(B). Para verlo, se escribe

$$B = A \cup A' B$$

de modo que

$$P(B) = P(A \cup A' B) = P(A) + P(A' B)$$

Como $P(A' B) \geq 0$, según el axioma (1), en consecuencia $P(A) \leq P(B)$. En especial, como A ⊂ S, para cualquier evento A, y P(S) = 1, entonces $P(A) \leq 1$.

De igual modo se puede demostrar que $P(\varnothing) = 0$. Como S y ∅ son disjuntos y S ∪ ∅ = S,

$$1 = P(S) = P(S \cup \varnothing) = P(S) + P(\varnothing)$$

La definición de probabilidad sólo señala los axiomas a los que debe apegarse esa función; no dice qué números asignar a eventos específicos. La asignación real de números se lleva a cabo, normalmente, de acuerdo a la evidencia empírica, o según consideraciones cuidadosas acerca del experimento.

## 2.2 Variables Aleatorias

La mayor parte de los experimentos generan resultados que se pueden interpretar en términos de números reales. Estos resultados numéricos, cuyos valores pueden cambiar de un experimento a otro, se llaman *variables aleatorias*. En sí, una variable aleatoria es una función de valor real cuyo dominio es un espacio muestral.

Las variables aleatorias se representarán mediante negritas, como por ejemplo **a**, **b** y **c**. Los valores numéricos reales que puede asumir una variable aleatoria se representarán mediante cursivas, como por ejemplo *a, b* y *c*. Se puede hablar de "la probabilidad de que **a** tome el valor *a*," $P(\mathbf{a} = a)$ y representarla mediante $p(a)$.

Se dice que una variable aleatoria **x** es *discreta* si puede tomar sólo un número finito, o un número infinito contable, de valores posibles *x*.

En este caso,

(1) $P(\mathbf{x} = x) = p(x) \geq 0$

(2) $\sum_x P(\mathbf{x} = x) = 1$, siendo la suma con respecto a todos los valores posibles de *x*.

A la función p(*x*) se le llama función de probabilidad de **x**.

A la función de probabilidad se le llama en ocasiones *función de masa de probabilidad* de **x** para dar la idea de que se apila una masa de probabilidad en puntos discretos.

La función de probabilidad *p*(*x*), para cualquier variable aleatoria, tiene dos propiedades:

1. $0 \leq p(x) \leq 1$ para toda x
2. $\sum_i p_i(x) = 1$, siendo la sumatoria en todos los valores posibles de x.

A veces se estudia el comportamiento de variables aleatorias recurriendo a sus *probabilidades acumuladas*. Esto es, para cualquier variable aleatoria **x** se puede ver $P\ \mathbf{x} \le x$ para todo número real *x*. Esta es la propiedad acumulada de **x** evaluada en *x*. Por lo tanto, se puede definir la *función de distribución* $F_x(x)$ de una variable **x** como

$$F_x\ x = P\ \mathbf{x} \le x \tag{2.2-1}$$

Si **x** es discreta,

$$F_x\ x = \sum_{x_i \le x} p\ x_i \tag{2.2-2}$$

siendo $p(x)$ la función de probabilidad.

A la función de distribución se le llama a veces *función de distribución acumulada* (f.d.a).

En el caso de una variable aleatoria *continua* **x**, su distribución de probabilidad está caracterizada por una función $f_x(x)$ que recibe el nombre de *función de densidad de probabilidad*. Esta función $f_x(x)$ no es la misma función de probabilidad que para el caso discreto. Como existe la probabilidad de que **x** tome el valor específico *x* es cero, la función de densidad de probabilidad no representa la probabilidad de que **x** = *x*. Más bien, ésta proporciona un medio para determinar la probabilidad de un intervalo $a \le \mathbf{x} \le b$.

La función $f_x(x)$, cuya gráfica es la curva límite que se obtiene para un número muy grande de observaciones y para una amplitud de intervalo muy pequeña, es la función de densidad de probabilidad para una variable aleatoria continua **x**, ya que la escala vertical se elige de manera que el área total bajo la curva es igual a uno. La función de densidad de probabilidad de una variable aleatoria continua **x** se define formalmente de la siguiente manera:

$$\begin{aligned} &f_x\ x \ge 0, \quad -\infty < x < \infty, \\ &\int_{-\infty}^{\infty} f_x\ x\, dx = 1, \\ &F_x\ a = \int_{-\infty}^{a} f_x\ \xi\ d\xi, \quad \text{y} \\ &P\ a \le \mathbf{x} \le b = \int_{a}^{b} f_x\ x\, dx \end{aligned} \tag{2.2-3}$$

para cualquiera *a* y *b*, entonces $f_x(x)$ es la función de densidad de probabilidad de la variable aleatoria continua **x**.

## 2.3 Teorema de Bayes

En la mayoría de las aplicaciones reales, las decisiones se actualizan conforme se obtiene nueva información o cuando existe un cambio de escenarios. El teorema de Bayes proporciona un método mediante el cual la probabilidad de cierto evento que ya es conocido (probabilidad *a priori* o previa) se

va actualizando conforme se obtiene nueva información. Una vez que la probabilidad ha sido actualizada se le llama probabilidad *a posteriori* (o probabilidad posterior).

La probabilidad condicional determina la forma en que un evento es influido o determinado, dada la presencia de otro evento. Por esta razón, las probabilidades a posteriori son probabilidades condicionales, pues han sido actualizadas por la presencia de un nuevo evento y al haberse obtenido mayor información. El teorema de Bayes se utiliza para obtener probabilidades más precisas que las probabilidades a priori, dada la presencia de un nuevo evento.

Si A y B son eventos cualesquiera, la *probabilidad condicional* de A dado B, que se representa mediante P(A | B), es

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

siempre que P(B) > 0.

Existen dos reglas básicas para manipular probabilidades: la *regla del producto* y la *regla de la suma*.

Regla del producto: $P(A \cap B|C) = P(A|C)P(B|A \cap C)$

Regla de la suma: $P(A \cup B|C) = P(A|C) + P(B|C) - P(A \cap B|C)$

La regla de la suma tiene un papel importante en el teorema de la probabilidad total (por simplicidad, se escribirá P(A) para P(A | C)). Este teorema es usado para evaluar P(B) en términos de $P(B | A_i)$ y $P(A_i)$.

El teorema de *probabilidad total* indica que dados *n* eventos mutuamente exclusivos $A_1,\ldots,A_n$ cuya unión es el evento seguro (de probabilidad igual a 1), es decir dado lo siguiente:

$$A_i \cap A_j = \varnothing \quad \forall\, i,j \;\; con \;\; i \neq j \;\; e \;\; i = 1,\ldots n$$

$$\bigcup_{i=1}^{n} A_i = \Omega \qquad (2.3\text{-}1)$$

La ecuación siguiente se cumple para cualquier evento arbitrario B:

$$P(B) = \sum_{i=1}^{n} P(B|A_i)P(A_i) \qquad (2.3\text{-}2)$$

Cabe aclarar que el teorema anterior es válido aún cuando la suma de los eventos $A_i$ no es el evento seguro si:

$$\bigcup_{i=1}^{n} A_i \subset B \qquad (2.3\text{-}3)$$

Una vez revisados los conceptos anteriores, se presenta el *Teorema de Bayes*, el cual se expresa de la siguiente manera:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{2.3-4}$$

La probabilidad P(A | B) es la *probabilidad a posteriori* y P(A) es la *probabilidad a priori* de A.

## 2.4 Función Gaussiana (normal)

Una variable aleatoria **x** es *gaussiana* o *normal* si su función de densidad tiene la forma

$$f_x(x) = \frac{1}{\sqrt{2\pi\sigma_x^2}} e^{-(x-a_x)^2/2\sigma_x^2} \tag{2.4-1}$$

donde $\sigma_x > 0$ y $-\infty < a_x < \infty$ son constantes reales. Esta función está trazada en la Figura 2.4-1(a). Su valor máximo $1/\sqrt{2\pi\sigma_x^2}$ ocurre en $x = a_x$. Su distribución alrededor del punto $x = a_x$ está relacionada con $\sigma_x$. La función decrece a 0.607 veces su valor máximo en el punto $x = a_x + \sigma_x$ y $x = a_x - \sigma_x$.

La densidad gaussiana es la de mayor importancia de todas las densidades y se utiliza en casi todas las áreas de ciencia e ingeniería. Esta importancia deriva de su descripción precisa de muchas cantidades prácticas y del mundo real, especialmente cuando dichas cantidades son el resultado de varios efectos aleatorios independientes actuando para generar la cantidad de interés.

La función de distribución se puede encontrar, como se mencionó anteriormente, integrando la función de densidad:

$$F_x(x) = \frac{1}{\sqrt{2\pi\sigma_x^2}} \int_{-\infty}^{x} e^{-(\xi-a_x)^2/2\sigma_x^2} d\xi \tag{2.4-2}$$

La integral no puede evaluarse en forma cerrada; sin embargo, se puede tabular $F(x; a_x, \sigma_x)$ como una función de $a_x$ y $\sigma_x$, lo que necesitaría una tabla para cada par de valores. Como existe un número infinito de valores para $a_x$ y $\sigma_x$, lo cual requeriría un número infinito de tablas, esta tarea es virtualmente imposible. Una mejor aproximación es posible si una sola tabla de $F_x(x)$ se desarrolla que corresponde a normalizar valores de $a_x$ y $\sigma_x$. Después se puede usar esa única tabla como el caso general donde $a_x$ y $\sigma_x$ pueden ser valores arbitrarios.

Se empieza seleccionando el caso normalizado donde $a_x = 0$ y $\sigma_x = 1$. Se indica la función de distribución con $F(x)$:

$$F(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\xi^2/2} d\xi \tag{2.4-3}$$

que es una función de *x* únicamente.

Para un valor negativo de *x* se usa la relación

$$F\ @x^{\alpha} = 1\ @F\ \ x^{\upsilon} \tag{2.4-4}$$

Para mostrar que la función general de distribución $F_x(x)$ de (2.4-2) puede encontrarse en términos de $F(x)$ de (2.4-3), se hace un cambio de variable

$$u = \frac{\xi\ @a_x}{\sigma_x} \tag{2.4-5}$$

en (2.4-2) para obtener

$$F_x\ \grave{}\ x^{a} = \frac{1}{p\ 2\pi} \int_{@1}^{x@a_x{}^{a}/\sigma_x} e^{@\frac{u^2}{2}}\ du \tag{2.4-6}$$

De (2.4-3), esta expresión es claramente equivalente a

$$F_x\ \grave{}\ x^{a} = F^{\Gamma}\ \frac{x\ @a_x}{\sigma_x}^{g}$$

La figura 2.4-1(b) representa el comportamiento de $F_x(x)$.

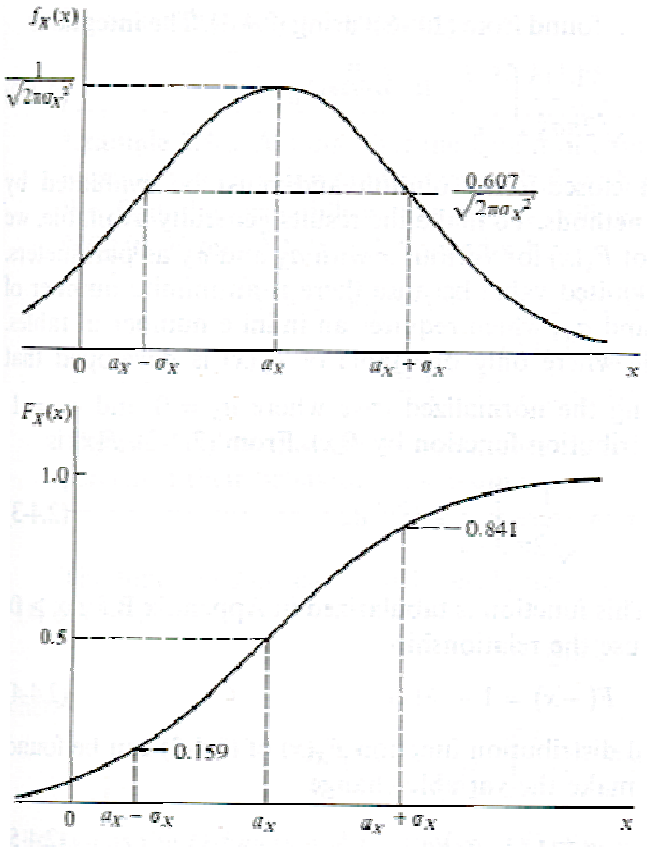



Figura 2.4-1

# 2.5 Teorema del Límite Central

Dadas *n* variables aleatorias independientes $\mathbf{x}_i$, se forma su suma

$$\mathbf{x} = \mathbf{x}_1 + \ldots + \mathbf{x}_n$$

Esta es una variable aleatoria con media $\eta = \eta_1 + \ldots + \eta_n$ y varianza $\sigma^2 = \sigma_1^2 + \ldots + \sigma_n^2$. El teorema del límite central dicta que bajo ciertas condiciones generales, la distribución $F(x)$ de $\mathbf{x}$ se aproxima a la distribución gaussiana con la misma media y varianza

$$F(x) \approx G\left(\frac{x-\eta}{\sigma}\right) \tag{2.5-1}$$

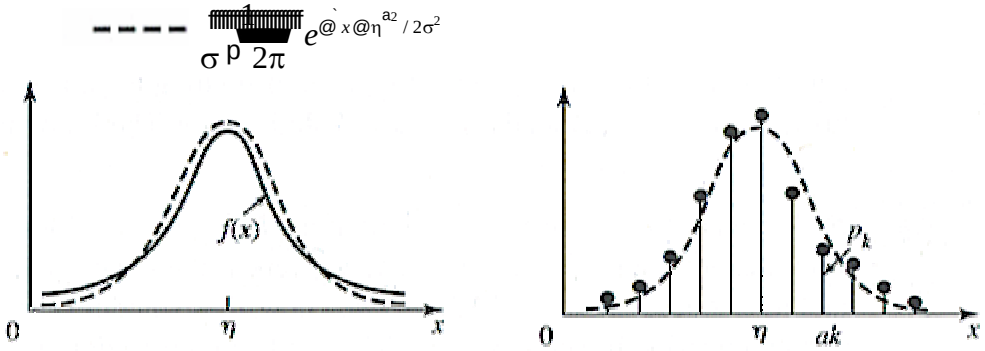
conforme *n* crece. Además, si las variables aleatorias $\mathbf{x}_i$ son de tipo continuo, la función de densidad $f(x)$ de $\mathbf{x}$ se aproxima a la función de densidad gaussiana (Figura 2.5-1(a))

$$f(x) \approx \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\eta)^2}{2\sigma^2}} \tag{2.5-2}$$

Este teorema importante puede definirse como un límite: Si $\mathbf{z} = (\mathbf{x}-\eta)/\sigma$ entonces

$$\lim_{n\to\infty} F_z(z) = G(z) \qquad \lim_{n\to\infty} f_z(z) = \frac{1}{\sqrt{2\pi}} e^{-z^2/2}$$

para el caso general y el continuo respectivamente.




(a) (b)

Figura 2.5-1

La naturaleza de la aproximación del teorema del límite central y el valor de *n* requerido para un rango de error determinado depende de la forma de las funciones de densidad $f_i(x)$. Si las variables aleatorias $\mathbf{x}_i$ son i.i.d. (independientes y distribuidas idénticamente), el valor de $n = 30$ es adecuado para la mayoría de las aplicaciones. De hecho, si las funciones $f_i(x)$ son suaves, valores tan pequeños como $n = 5$ pueden ser utilizados.

Para una variable aleatoria de tipo discreta, $F(x)$ es una función escalera aproximada a la distribución gaussiana. Sin embargo, las probabilidades $p_k$, que **x** iguala a valores específicos $x_k$ no están, en general, relacionadas a la densidad gaussiana. Existe una excepción para variables aleatorias llamadas de tipo Lattice: si las variables aleatorias $\mathbf{x}_i$ toman valores equidistantes $ak_i$, entonces **x** toma los valores $ak$ y para un valor grande de $n$, las discontinuidades $p_k = P(\mathbf{x} = ak)$ de $F(x)$ en los puntos $x_k = ak$ igualan las muestras de la densidad gaussiana (Figura 2.5-1(b))

$$P(\mathbf{x} = ak) \approx \frac{1}{\sigma\sqrt{2\pi}} e^{-(ak-\eta)^2/2\sigma^2} \tag{2.5-3}$$

## 2.5.1 Desigualdades de Markov y Chebyshev

Una medida de la concentración de una variable aleatoria alrededor de su media $\eta$ es su varianza $\sigma^2$. De hecho, así como el teorema que se muestra a continuación lo muestra, la probabilidad que $x$ esté fuera de un intervalo arbitrario $(\eta-\varepsilon, \eta+\varepsilon)$ es insignificante si la razón $\sigma/\varepsilon$ es suficientemente pequeña. Este resultado, conocido como la *desigualdad de Chebyshev*, es fundamental.

Desigualdad de Chebyshev

Para cualquier $\varepsilon > 0$,

$$P(|\mathbf{x}-\eta| \geq \varepsilon) \leq \frac{\sigma^2}{\varepsilon^2} \tag{2.5.1-1}$$

La demostración está basada en el hecho de que

$$P(|\mathbf{x}-\eta| \geq \varepsilon) = \int_{-\infty}^{\eta-\varepsilon} f(x)\,dx + \int_{\eta+\varepsilon}^{\infty} f(x)\,dx = \int_{|x-\eta|\geq\varepsilon} f(x)\,dx$$

En efecto

$$\sigma^2 = \int_{-\infty}^{\infty} (x-\eta)^2 f(x)\,dx \geq \int_{|x-\eta|\geq\varepsilon} (x-\eta)^2 f(x)\,dx \geq \varepsilon^2 \int_{|x-\eta|\geq\varepsilon} f(x)\,dx$$

y se tiene como resultado (2.5.1-1) debido a que la última integral es igual a $P(|\mathbf{x}-\eta| \geq \varepsilon)$.

La desigualdad de Chebyshev es muy importante porque permite determinar los límites de las probabilidades de variables aleatorias discretas o continuas sin tener que especificar sus funciones (densidades) de probabilidad.

Desigualdad de Markov

Si $f(x) = 0$ para $x < 0$, entonces, para cualquier $\alpha > 0$,

$$P\{\mathbf{x} \geq \alpha\} \leq \frac{\mu}{\alpha} \qquad (2.5.1\text{-}2)$$

La demostración está basada en el hecho de que

$$E\{\mathbf{x}\} = \int_0^{\infty} x f(x)\,dx \geq \int_{\alpha}^{\infty} x f(x)\,dx \geq \alpha \int_{\alpha}^{\infty} f(x)\,dx$$

y se tiene como resultado (2.5.1-2) debido a que la última integral es igual a $P\{\mathbf{x} \geq \alpha\}$.

## 2.5.2 Ley de los Grandes Números

Si la probabilidad de un evento *A* en un experimento dado es igual a *p* y el número de éxitos de *A* en *n* pruebas es igual a *k*, entonces

$$P\left\{\left|\frac{k}{n} - p\right| < \varepsilon\right\} \to 1 \qquad \text{conforme} \qquad n \to \infty \qquad (2.5.2\text{-}1)$$

Se reestablecerá este resultado como un límite de una secuencia de variables aleatorias. Para este propósito se presentan las siguientes variables aleatorias

$$\mathbf{x}_i = \begin{cases} 1 & \text{si A ocurre en la } i\text{-ésima prueba} \\ 0 & \text{en otro caso} \end{cases}$$

Mostraremos que el promedio de la muestra

$$\bar{\mathbf{x}}_n = \frac{\mathbf{x}_1 + \mathbf{x}_2 + \cdots + \mathbf{x}_n}{n}$$

de estas variables aleatorias tiende a *p* en probabilidad conforme *n* tiende a $\infty$.

Para demostrarlo sabemos que

$$E\{\mathbf{x}_i\} = E\{\bar{\mathbf{x}}_n\} = p \qquad \sigma_{x_i}^2 = pq \qquad \sigma_{\bar{x}_n}^2 = \frac{pq}{n}$$

Además, $pq = p(1-p) \leq 1/4$. Por lo tanto

$$P\{|\bar{\mathbf{x}}_n - p| < \varepsilon\} \geq 1 - \frac{pq}{n\varepsilon^2} \geq 1 - \frac{1}{4n\varepsilon^2} \to 1$$

Esto restablece (2.5.2-1) porque $\bar{\mathbf{x}}_n(\xi) = k/n$ si *A* ocurre *k* veces.

# 3.0 Procesos Estocásticos

Como se describió anteriormente, una variable aleatoria **x** es una regla para asignar a cada resultado $\xi$ de algún experimento *S* un *número* **x**($\xi$). Un proceso estocástico **x**(*t*) es una regla para asignar a cada $\xi$ una *función* **x**(t, $\xi$). Así, un proceso estocástico es una familia de funciones de tiempo dependiendo del parámetro $\xi$ o, lo que es lo mismo, una función de *t* y $\xi$. El dominio de $\xi$ es una serie de todos los resultados experimentales y el dominio de *t* es la serie *R* de números reales.

Si *R* es el eje real, entonces **x**(*t*) es un proceso de *tiempo continuo*. Si *R* es la serie de enteros, entonces **x**(*t*) es un proceso de *tiempo discreto*. Un proceso de tiempo discreto es, de esta manera, una secuencia de variables aleatorias. Tal secuencia se representará como $\mathbf{x}_n$, o, para evitar índices dobles, como **x**[*n*].

Se dirá que **x**(*t*) es un proceso de *estado discreto* si sus variables son contables. De otra forma, es un proceso de *estado continuo*.

Se utilizará la notación **x**(*t*) para representar un proceso estocástico omitiendo su dependencia de $\xi$. Así **x**(*t*) tiene las siguientes interpretaciones:

1. Es un grupo de funciones **x**(*t*,$\xi$). En esta interpretación *t* y $\xi$ son variables.
2. Es una *muestra* de un proceso dado. En este caso, *t* es una variable y $\xi$ está definida.
3. Si *t* esta definida y $\xi$ es una variable, entonces **x**(*t*) es una variable aleatoria igual al estado del proceso dado en el tiempo *t*.
4. Si *t* y $\xi$ están definidas, entonces **x**(*t*) es un *número*.

## 3.1 Estacionalidad

Un proceso estocástico es conocido como de *sentido estacionario estricto* (abreviado como SSS por sus siglas en inglés) si sus propiedades estadísticas son invariantes ante un cambio del origen. Esto significa que los procesos **x**(*t*) y **x**(*t*+*c*) tienen las mismas propiedades estadísticas para cualquier *c*.

Dos procesos **x**(*t*) y **y**(*t*) son de *estacionalidad conjunta* si las propiedades estadísticas conjuntas de **x**(*t*) y **y**(*t*) son iguales que las propiedades estadísticas de **x**(*t*+*c*) y **y**(*t*+*c*) para cualquier *c*.

Un proceso complejo **z**(*t*) = **x**(*t*) + *j***y**(*t*) es estacionario si los procesos **x**(*t*)y **y**(*t*) son de estacionalidad conjunta.

A partir de la definición puede decirse que la densidad de *n*-ésimo orden de un proceso SSS debe ser tal que

$$f\left(x_1, \ldots, x_n ; t_1, \ldots, t_n\right) = f\left(x_1, \ldots, x_n ; t_1 + c, \ldots, t_n + c\right) \tag{3.1-1}$$

para cualquier *c*.

A partir de esto se puede decir que $f(x; t) = f(x; t+c)$ para cualquier *c*. Por lo tanto la densidad de primer orden de **x**(*t*) es independiente de *t*:

$$f(x; t) = f(x) \tag{3.1-2}$$

De forma similar, $f(x_1, x_2 ; t_1+c, t_2+c)$ es independiente de *c* para cualquier *c*, en particular para $c = -t_2$. Esto nos lleva a la conclusión de que

$$f\left(x_1, x_2, t_1, t_2\right) = f\left(x_1, x_2; \tau\right) \qquad \tau = t_1 - t_2 \tag{3.1-3}$$

Así, la densidad conjunta de la variable aleatoria $\mathbf{x}(t + \tau)$ y $\mathbf{x}(t)$ es independiente de $t$ y es igual a $f\left(x_1, x_2; \tau\right)$.

Un proceso estocástico es conocido como de *sentido estacionario amplio* (abreviado como WSS por sus siglas en inglés) si su promedio es constante

$$E\left\{\mathbf{x}\left(t\right)\right\} = \eta \tag{3.1-4}$$

y su autocorrelación depende únicamente de $\tau = t_1 - t_2$:

$$E\left\{\mathbf{x}\left(t+\tau\right)\mathbf{x}^*\left(t\right)\right\} = R\left(\tau\right) \tag{3.1-5}$$

Ya que $\tau$ es la distancia desde $t$ a $t + \tau$, la función $R(\tau)$ puede escribirse en la forma simétrica

$$R\left(\tau\right) = E\left\{\mathbf{x}\left(t + \frac{\tau}{2}\right)\mathbf{x}^*\left(t - \frac{\tau}{2}\right)\right\} \tag{3.1-6}$$

Nótese en particular que

$$E\left\{\left|x\left(t\right)\right|^2\right\} = R\left(0\right)$$

## 3.2 Ergodicidad

Un problema importante en la aplicación de procesos estocásticos es la estimación de varios parámetros estadísticos en términos de datos reales. La mayoría de los parámetros pueden ser expresados como valores esperados de alguna función de algún proceso $\mathbf{x}(t)$ . El problema de estimar el promedio de un proceso $\mathbf{x}(t)$ dado es, por lo tanto, fundamental.

Para un valor de $t$ específico, $\mathbf{x}(t)$ es una variable aleatoria; su promedio $\eta\left(t\right) = E\{\mathbf{x}(t)\}$ puede, por lo tanto, estimarse de la siguiente manera: Se observan $n$ muestras $\mathbf{x}(t,\xi_i)$ de $\mathbf{x}(t)$ y se usa, como el punto estimado de $E\{\mathbf{x}(t)\}$, el promedio

$$\hat{\eta}\left(t\right) = \frac{1}{n}\sum_i \mathbf{x}\left(t, \xi_i\right)$$

Como se sabe, $\hat{\eta}\left(t\right)$ es un estimador consistente de $\eta\left(t\right)$; sin embargo, puede ser usado únicamente si un número grande de realizaciones $\mathbf{x}(t,\xi_i)$ de $\mathbf{x}(t)$ están disponibles. En muchas aplicaciones, se conoce una *única muestra* de $\mathbf{x}(t)$. ¿Se puede entonces estimar $\eta\left(t\right)$ en términos del tiempo promedio de una muestra dada? Esto no es posible si $E\{\mathbf{x}(t)\}$ depende de $t$. Sin embargo, si $\mathbf{x}(t)$ es un proceso estacionario regular, su promedio de tiempo tiende a $E\{\mathbf{x}(t)\}$ conforme la longitud de la muestra disponible tiende a $\infty$ . La ergodicidad es un tema que trata esta importante teoría.

A partir de un proceso estacionario real $\mathbf{x}(t)$ deseamos estimar su promedio $\eta = E\{\mathbf{x}(t)\}$. Para este fin, se crea el *promedio en el tiempo*

$$\eta_T = \frac{1}{2T}\int_{-T}^{T} \mathbf{x}(t)\,dt \tag{3.2-1}$$

Claramente, $\eta_T$ es una variable aleatoria con promedio

$$E\{\eta_T\} = \frac{1}{2T}\int_{-T}^{T} E\{\mathbf{x}(t)\}\,dt = \eta$$

Así $\eta_T$ es un estimador imparcial de $\eta$. Si su varianza $\sigma_T^2 \to 0$ conforme $T \to \infty$, entonces $\eta_T \to \eta$. En este caso, el promedio en el tiempo $\eta_T(\xi)$ obtenido de una única realización de $\mathbf{x}(t)$ es cercano a $\eta$ con probabilidad cercana a 1. Si esto es cierto, se dirá que el proceso $\mathbf{x}(t)$ es de *promedio ergódico*. Así, un proceso $\mathbf{x}(t)$ es de promedio ergódico si su promedio en el tiempo $\eta_T$ tiende al conjunto promedio $\eta$ conforme $T \to \infty$.

Para establecer la ergodicidad de un proceso, es suficiente encontrar $\sigma_T$ y examinar las condiciones bajo las cuales $\sigma_T \to 0$ conforme $T \to \infty$.

## 3.3 Procesos Aleatorios de Markov

Los procesos aleatorios de Markov representan la generalización más simple de procesos independientes, permitiendo que el resultado de cualquier instante dependa únicamente del resultado precedente y de ninguno anterior a eso. Por lo mismo, en un proceso aleatorio de Markov $\mathbf{x}(t)$, el pasado no tiene ninguna influencia en el futuro si el presente está especificado. Esto significa que si $t_{n-1} < t_n$, entonces

$$P\{\mathbf{x}(t_n) \le x_n \mid \mathbf{x}(t), t \le t_{n-1}\} = P\{\mathbf{x}(t_n) \le x_n \mid \mathbf{x}(t_{n-1})\} \tag{3.3-1}$$

De (3.3-1) se sabe que si $t_1 < t_2 < \ldots < t_n$, entonces

$$P\{\mathbf{x}(t_n) \le x_n \mid \mathbf{x}(t_{n-1}), \mathbf{x}(t_{n-2}), \ldots, \mathbf{x}(t_1)\} = P\{x(t_n) \le x_n \mid x(t_{n-1})\} \tag{3.3-2}$$

Un tipo especial de proceso de Markov es una cadena de Markov donde el sistema puede ocupar un número finito o infinito contable de estados $e_1, e_2, \ldots, e_j, \ldots$ tales que la evolución futura de los procesos, una vez que está en un estado dado, depende únicamente del estado presente y no en cómo llegó a ese estado. Ambos, los procesos de Markov y las cadenas de Markov pueden ser en tiempo discreto o continuo, dependiendo de si la serie de tiempo es discreto o continuo.

Los procesos de Markov fueron así llamados por A. A. Markov (1856 – 1922), quien introdujo este concepto para sistemas de parámetros discretos con un número finito de estados en 1907. La teoría para cadenas de números infinitos contables fue dada por Kolmogorov (1936) seguido por Doeblin (1937), Doob (1942), Levy (1951), y muchos otros.

Un proceso de Markov en tiempo continuo **x**(*t*) puede tomar de forma aleatoria una cantidad finita o infinita de estados $e_0, e_1, e_3, \ldots$ en un tiempo *t*. La situación del proceso en el tiempo *t* está descrito por **x**(*t*) y es igual al estado $e_j$ que el proceso toma en ese tiempo. Supóngase que el proceso **x**(*t*) está en el estado $e_i$ en el tiempo $t_0$. Para un proceso de Markov, de (3.2-2) la probabilidad de que un proceso pase al estado $e_j$ en el tiempo $t_0 + t$ está dado por

$$P\{\mathbf{x}(t_0 + t) = e_j \mid \mathbf{x}(t_0) = e_i\} \tag{3.3-3}$$

y esta probabilidad es *independiente* del comportamiento del proceso **x**(*t*) antes del instante $t_0$. Si **x**(*t*) es un *proceso homogéneo de Markov*, entonces esta probabilidad de transición del estado $e_i$ al estado $e_j$ no depende del tiempo inicial $t_0$ pero sí, únicamente, del tiempo transcurrido, *t*, entre las transiciones. Así, en el caso de una cadena homogénea de Markov (3.3-3) se reduce a

$$p_{ij}(t) = P\{\mathbf{x}(t_0 + t) = e_j \mid \mathbf{x}(t_0) = e_i\} \tag{3.3-4}$$

En particular se tiene que

$$p_{ij}(t) = P\{\mathbf{x}(t) = e_j \mid \mathbf{x}(0) = e_i\} \tag{3.3-5}$$

donde

$$p_i(0) = P\{\mathbf{x}(0) = e_i\} \tag{3.3-6}$$

representa la probabilidad de la distribución inicial de los estados. Para todos los estados $e_i$, $e_j$ se tiene

$$0 \le p_{ij}(t) \le 1 \qquad \sum_j p_{ij}(t) = 1 \tag{3.3-7}$$

y la probabilidad incondicional del evento "**x**(*t*) está en el estado $e_j$" está dado por

$$p_j(t) = P\{\mathbf{x}(t) = e_j\} = \sum_i P\{\mathbf{x}(t) = e_j \mid \mathbf{x}(0) = e_i\} P\{\mathbf{x}(0) = e_i\} = \sum_i p_i(0) p_{ij}(t) \tag{3.3-8}$$

De forma más general para *t* y *s* arbitrarias, se tiene que

$$\begin{aligned} p_{ij}(t+s) &= P\{\mathbf{x}(t+s) = e_j \mid \mathbf{x}(0) = e_i\} \\ &= \sum_k P\{\mathbf{x}(t+s) = e_j \mid \mathbf{x}(t) = e_k, \mathbf{x}(0) = e_i\} P\{\mathbf{x}(t) = e_k \mid \mathbf{x}(0) = e_i\} \\ &= \sum_k P\{\mathbf{x}(t) = e_k \mid \mathbf{x}(0) = e_i\} P\{\mathbf{x}(t+s) = e_j \mid \mathbf{x}(t) = e_k\} \\ &= \sum_k p_{ik}(t) p_{kj}(s) \end{aligned} \tag{3.3-9}$$

# 4.0 Campos Aleatorios de Markov

En la década de 1920, en su mayoría inspirados en el modelo de Ising, un tipo nuevo de procesos estocásticos apareció en la teoría de la probabilidad llamado *Campo Aleatorio de Markov* (MRF por sus siglas en inglés). Estos campos se convirtieron rápidamente en una herramienta ampliamente utilizada en problemas no sólo de tipo estadístico. Su uso en procesamiento de imágenes data de inicios de la década de 1970.

## 4.1 El Modelo Ising

De acuerdo con Ising, se considera una secuencia, 0, 1, 2,…, *n* en la línea. En cada punto hay un determinado giro hacia arriba o hacia abajo en cualquier momento dado (Figura 4.1-1).

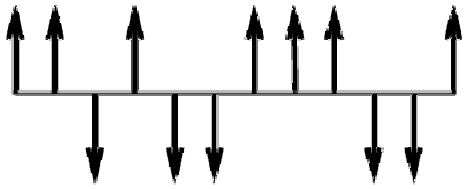


Figura 4.1-1 *Modelo de Ising unidimensional*

Ahora se define una medida de probabilidad en el conjunto $\Omega$ de todas las configuraciones posibles $\omega = \omega_0, \omega_1, \ldots, \omega_n$. En este contexto, cada giro es una función

$$\delta_i(\omega) = \begin{cases} 1 & \text{si } \omega_i \text{ está arriba} \\ -1 & \text{si } \omega_i \text{ está abajo} \end{cases} \tag{4.1-1}$$

Una *energía* $U(\omega)$ es asignada a cada configuración

$$U(\omega) = -J \sum_{i,j} \delta_i(\omega)\delta_j(\omega) - mH \sum_i \delta_i(\omega) \tag{4.1-2}$$

En la primera suma, Ising asumió la simplificación de que únicamente las interacciones de puntos con una unidad aparte necesitan ser tomadas en cuenta. Este término representa la energía ocasionada por la interacción de los giros. La constante *J* es una propiedad del material. Si $J > 0$, las interacciones tienden a mantener giros de vecindad en la misma dirección (*caso atractivo*). Si $J < 0$, los giros de vecindad con orientación opuesta son favorecidos (*caso repulsivo*). El segundo término representa la influencia de un campo magnético exterior de intensidad *H* y $m > 0$ es una propiedad el material. La medida de probabilidad en $\Omega$ está dado por

$$P(\omega) = \frac{e^{-U(\omega)/kT}}{Z} \tag{4.1-3}$$

donde T es la temperatura y *k* es la constante de Boltzmann. La constante *Z* está definida por

$$Z = \sum_{\omega \in \Omega} e^{-\frac{U(\omega)}{kT}} \tag{4.1-4}$$

La medida de probabilidad de (4.1-3) es llamada *distribución de Gibbs.* El modelo se puede extender a dos dimensiones de forma natural. Los giros están organizados en una red cristalina, y son localizados por dos coordenadas, teniendo cada punto cuatro vecinos, a menos que esté en la frontera. En el caso de dos dimensiones, la medida $P$ es inestable y existe una transición de fase. Considerando el *caso atractivo* y el campo externo $h$, la medida $P_h$ converge en $P^{(-)}$ si $h$ se va a cero a través de valores negativos pero converge en $P^{+} \neq P^{-}$ si $h$ se va a cero a través de valores positivos. Se ha demostrado que existe una *temperatura crítica* $T_C$ y debajo de esta temperatura siempre ocurre una transición de fase. La temperatura depende de los parámetros de interacción verticales ($J_1$) y horizontales ($J_2$).

Un ejemplo es el *modelo de árbol de Cayley*. En este caso, los puntos están colocados en un árbol (Figura 4.1-2). La raíz se llama el nivel 0. Desde la raíz, se tienen $q$ ramas ($q$ = 2 en la Figura 4.1-2). El caso $q$ = 1 simplemente da una cadena de Markov unidimensional. Una configuración en un árbol de $n$ niveles es una asignación de una etiqueta *arriba* o *abajo* para cada punto. Se puede definir una función de energía similar como para el *modelo Ising*.

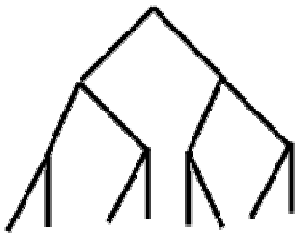


Figura 4.1-2 *modelo de árbol de Cayley*

Otra extensión del modelo Ising a más de dos estados por punto es el *modelo Potts*. El problema es observar el modelo Ising como un sistema de giros interactuando que pueden ser paralelos o antiparalelos. De forma más general, consideramos un sistema de giros, cada giro apuntando hacia alguna de las direcciones $q$ igualmente espaciadas. Estos vectores son la combinación lineal de $q$ cantidad de vectores unitarios apuntando hacia las direcciones simétricas de un híper-tetraedro en $q - 1$ direcciones. Para $q$ = 2, 3, 4 se muestran ejemplos en la Figura 4.1-3. La función de la energía del modelo de Potts puede ser escrito como

$$U(\omega) = \sum_{i,j} J(\Theta_{ij}) \tag{4.1-5}$$

donde $J(\Theta)$ es $2\pi$ periódico y $\Theta_{ij}$ es el ángulo entre dos giros vecinos en $i$ y $j$. El caso $q$ = 2 es equivalente al modelo Ising.

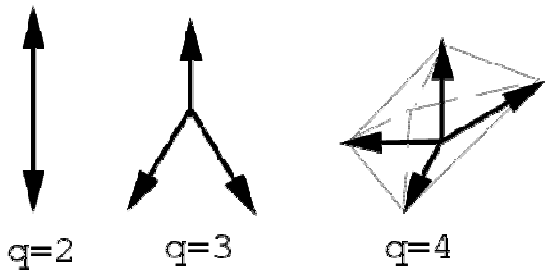



Figura 4.1-3 *Modelo de Potts*

## 4.2 Vecindades y Cliques

La forma más natural para definir campos aleatorios de Markov relacionados a modelos de imágenes es definirlos en una red. Sin embargo, aquí definiremos los CAM de forma más general, en gráficas. Sea $G = (S,E)$ una gráfica donde $S = \{s_1, s_2, \ldots s_n\}$ es una serie de vértices y $E$ es una serie de orillas.

Dos puntos $s_i$ y $s_j$ son *vecinos* si existe una orilla $e_{ij} \in E$ que los conecte. La serie de puntos que sean vecinos en un sitio $s$ (la vecindad de $s$) es denotada por $V_s$.

$V = \{V_s \mid s \in S\}$ es un *sistema de vecindad* para $G$ si

(i) $s \notin V_s$
(ii) $s \in V_r \Leftrightarrow r \in V_s$

Para cada sitio de la gráfica asignamos una etiqueta $\lambda$ de una serie finita de etiquetas $\Lambda$. Tal asignación es llamada una configuración $\omega$ teniendo probabilidad $P(\omega)$. La notación para un subconjunto $T \subset S$ es denotada por $\omega_T$ y $\omega_S \in \Lambda$. En lo siguiente nos interesaremos en las medidas de probabilidad asignadas a la serie $\Omega$ de todas las posibles configuraciones. Primero definamos las características locales como probabilidades condicionales $P(\omega_s \mid \omega_r, r \neq s)$.

$X$ es un Campo Aleatorio de Markov con respecto a $V$ si

(i) para toda $\omega \in \Omega$: $P(X = \omega) > 0$,
(ii) para toda $s \in S$ y $\omega \in \Omega$: $P(X_s = \omega_s \mid X_r = \omega_r, r \neq s) = P(X_s = \omega_s \mid X_r = \omega_r, r \in V_s)$

Para continuar con nuestra discusión acerca de medidas de probabilidad en $\Omega$, el concepto de *cliques* será de mucha utilidad.

Un subconjunto $C \subseteq S$ es un *clique* si todo par de distintos sitios en $c$ son vecinos. $C$ denota el conjunto de cliques.

Usando la definición anterior, podemos definir una medida de Gibbs en $\Omega$. Sea $V$ un *potencial* que asigna un número a cada configuración $\omega_T$. $V$ define una *energía* $U(\omega)$ en $\Omega$ mediante

$$U(\omega) = \sum_{T} V_T(\omega) \tag{4.2-1}$$

## 4.3 Esquemas Espaciales de Red

En esta sección trataremos con una subclase particular de Campos Aleatorios de Markov que son los más usados en los esquemas de procesamiento de imágenes. En este caso, consideramos *S* como una red *L* de tal forma que $\forall s \in S : s = (i, j)$ y definimos los llamados *n-ésimos sistemas de vecindad homogéneos* como

$$V^n = \{V^n_{i,j} : (i, j) \in L\}, \tag{4.3-1}$$

$$V^n_{i,j} = \{(k, l) \in L : (k - i)^2 + (l - j)^2 \le n\}. \tag{4.3-2}$$

Obviamente, sitios cercanos a la frontera tendrán menos vecinos que los del interior. Además, $V^0 = S$ y para toda $n \ge 0 : V^n \subset V^{n+1}$. La figura 4.3-1 muestra una vecindad de primer orden correspondiente a *n* = 1. Los cliques son {(i, j)}, {(i, j), (i, j+1)}, {(i, j), (i+1, j)}. En la práctica, más que sistemas de orden dos son raramente utilizados debido a que la función de energía sería demasiado complicada y requeriría mucha computación.

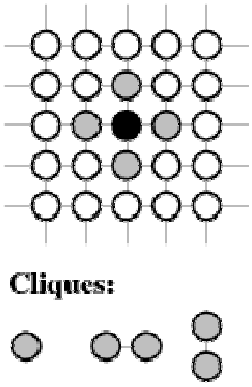



Figura 4.3-1
*Sistema de vecindad de primer orden*

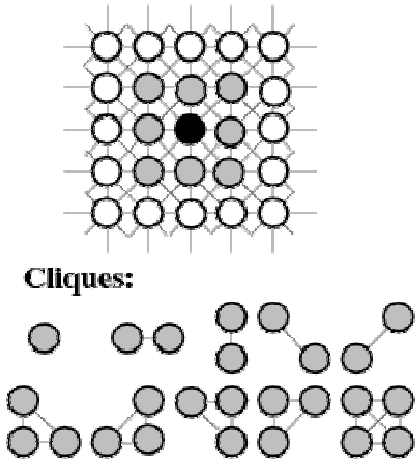



Figura 4.3-2
*Sistema de vecindad de segundo orden*

# 5.0 Optimización

El conocido ejemplo siguiente nos ayudará a entender de mejor forma los conceptos que se tratarán a lo largo de este capítulo:

*Ejemplo:* Dadas $N$ ciudades y las distancias $D_{ij}$ entre las ciudades $i$ y $j$ representando el costo de traslado. Uno debe planear la ruta *óptima* del viajero, quien pasará por todas las ciudades y regresará finalmente al punto de partida, minimizando la distancia total.

Como es sabido, no existe ningún método para encontrar la solución exacta para el problema anterior. Los esquemas de recocido simulado son una heurística propuesta para resolver el problema del viajero.

El recocido simulado está basado en la analogía entre la simulación del recocido de sólidos y la solución de problemas de optimización combinacional. Así fue como el algoritmo propuesto por Cerny y Kirkpatrick llegó a conocerse como *Recocido Simulado*.

En la física, recocido significa calentar un sólido a un valor máximo, al cual todas las partículas aleatoriamente se acomodan en la fase líquida, y después enfriarlo lentamente. De esta forma, todas las partículas se acomodan en el estado de baja temperatura de una red correspondiente. A cada temperatura $T$, se le permite al sólido alcanzar un *equilibrio térmico* que es caracterizado por la *distribución de Boltzmann*:

$$P(\omega) = \frac{1}{Z(T)} \exp\left(\frac{-U(\omega)}{kT}\right), \qquad (5.0\text{-}1)$$

donde $U(\omega)$ es la energía del estado, $Z(T)$ es la función de partición dependiendo de $T$ y $k$ es la constante de Boltzmann. Para una $T$ dada, la ecuación de arriba es igual a la distribución de Gibbs. Claramente, conforme la temperatura decrementa, la distribución de arriba se concentra en los estados con menor energía y cuando la temperatura se aproxima a cero, sólo los estados mínimos tienen una probabilidad diferente a cero. Disminuir la temperatura es crucial. Se sabe que si el enfriamiento es demasiado rápido y el sistema no logra alcanzar un equilibrio térmico para cada temperatura, un mínimo global no puede ser alcanzado.

Para una temperatura dada, la evolución para un equilibrio térmico es simulada por el *Algoritmo de metrópolis*. En éste, el algoritmo es usado para generar secuencias de configuraciones de un problema de optimización combinacional. El recocido simulado es entonces una secuencia de Algoritmos de metrópolis valuados en una secuencia de temperaturas decrecientes, tales que el equilibrio es alcanzado en cada temperatura.

Demos una definición formal del algoritmo de Recocido Simulado: Sean $\omega, \eta, \ldots$ las configuraciones de un problema de optimización combinacional (correspondientes a los estados de un sólido) y sea $U(\omega)$ el costo (también llamado energía) de la configuración $\omega$ (correspondiente a la energía del estado $\omega$ en un sistema termodinámico). Los elementos de las configuraciones están indexados por $S = (s_1, s_2, \ldots, s_N)$ y el espacio de estado común es denotado por $\Lambda = (0, 1, \ldots, L-1)$.

El conjunto de todas las posibles configuraciones se denota con $\Omega$. Debido a que $\forall s \in S : \omega_s \in \Lambda, \Omega = \Lambda^N$.

## 5.1 Algoritmo del Recocido Simulado

- Empezar con $k = 0$ y un valor aleatorio para $\omega$. Escoger un valor lo suficientemente elevado para la temperatura inicial $T = T_0$.

- Construir una perturbación de prueba $\eta$ de la actual configuración de $\omega$ tal que $\eta$ difiera únicamente en un elemento de $\omega$.

- (Criterio de Metrópolis) Hacer el cálculo de $\Delta U = U(\eta) - U(\omega)$ y aceptar $\eta$ si $\Delta U < 0$, de lo contrario aceptarlo con probabilidad $\exp(-\Delta U / T)$ (analogía con termodinámica):

$$\omega = \begin{cases} \eta & si\ \Delta U \le 0, \\ \eta & si\ \Delta U > 0\ y\ \xi < \exp(-\Delta U / T), \\ \omega & cualquier\ otro\ caso \end{cases} \qquad (5.1\text{-}1)$$

  donde $\xi$ es un número aleatorio uniforme entre 0 y 1.

- Repetir el segundo paso hasta alcanzar el equilibrio. Este equilibrio se alcanzará una vez que $\Delta U$ sea siempre menor a 0.

- Decrementar la temperatura: $T = T_{k+1}$ e ir al segundo paso con $k = k + 1$ hasta que el sistema se congele.

El algoritmo de arriba es conocido también como *recocido homogéneo* debido a que está descrito por una secuencia de cadenas homogéneas de Markov. Si la temperatura decrementa después de cada transición, el algoritmo es descrito por una cadena no homogénea de Markov y por ello nos referimos a éste como *recocido no homogéneo*. Ésta es la forma más común de recocido. Obtenemos tal algoritmo si quitamos el cuarto paso del algoritmo del recocido simulado.

## 5.2 Esquemas de Decremento de Temperatura

Se puede demostrar que el recocido simulado converge con una probabilidad de uno a una configuración global óptima si determinadas condiciones se cumplen para el esquema de temperatura. Es decir que $T_k$ se acerca a 0 no más rápido que $C / \ln(k)$ para una constante $C$ independiente de $k$ (recocido no homogéneo). Para el recocido homogéneo existen otras condiciones, pero nos concentraremos en el recocido no homogéneo debido a que es el esquema que más se utiliza.

Es claro que el esquema teórico anterior no puede ser implementado ya que es demasiado lento. Por lo tanto, debemos hacer una aproximación. Debido a esta aproximación, el algoritmo ya no garantiza encontrar una optimización global.

Temperatura Inicial

La temperatura inicial $T_0$ debe ser lo suficientemente elevada para que virtualmente todas las transiciones sean aceptadas. Es extremadamente difícil determinar dicho valor ya que está relacionado a los valores máximo y mínimo de la función de energía a ser minimizada. Existen algunas heurísticas para obtener una estimación razonable de la temperatura inicial, pero usualmente se le da a $T_0$ un valor relativamente bajo, obteniendo así una ejecución más rápida del algoritmo. En los experimentos realizados en esta tesis se ha utilizado el valor de $T_0 = 2.5$.

Temperatura Final

Obviamente, $\lim_{k \to \infty} T_k = 0$ puede únicamente ser aproximado en un número finito de valores para $T_k$. Por lo tanto, necesitamos un criterio de parada que determine el valor final de la temperatura. Podemos simplemente determinar el número de valores $T_k$ o detener la ejecución del algoritmo en las últimas configuraciones obtenidas por el recocido simulado que tengan ya prácticamente la misma energía.

Esquema de decremento de temperatura

El punto más importante es tener una regla de decremento de temperatura. Las reglas logarítmicas son normalmente muy lentas para darles un uso práctico (Figura 5.2-1). Por el contrario, las reglas exponenciales (Figura 5.2-2) son las que se usan con mayor frecuencia:

$$T_{k+1} = c \cdot T_k , \qquad k = 0 , 1 , 2 , \ldots \tag{5.2-1}$$

donde $c < 1$ es una constante cercana a uno. Esta regla fue propuesta por primera vez con $c = 0.95$ y se utiliza ampliamente actualmente. En los experimentos de esta tesis se utiliza este valor también.

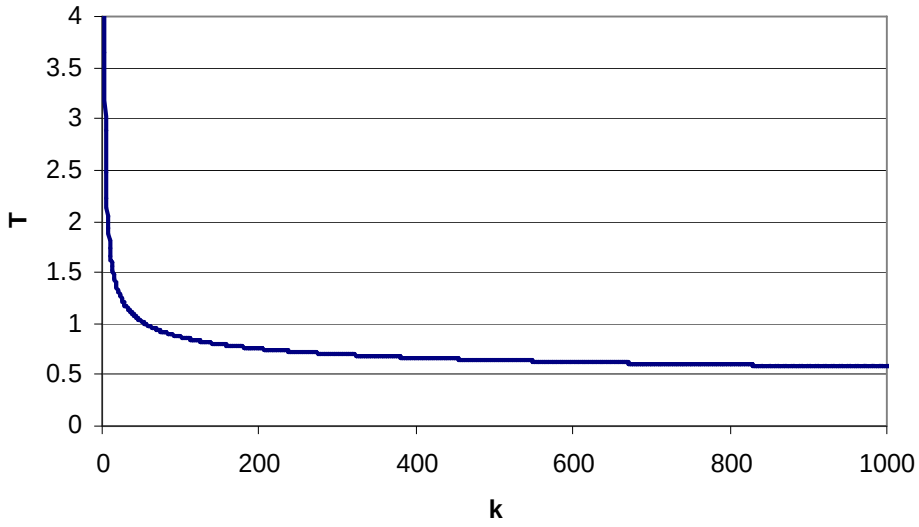



Figura 5.2-1
*Regla logarítmica*

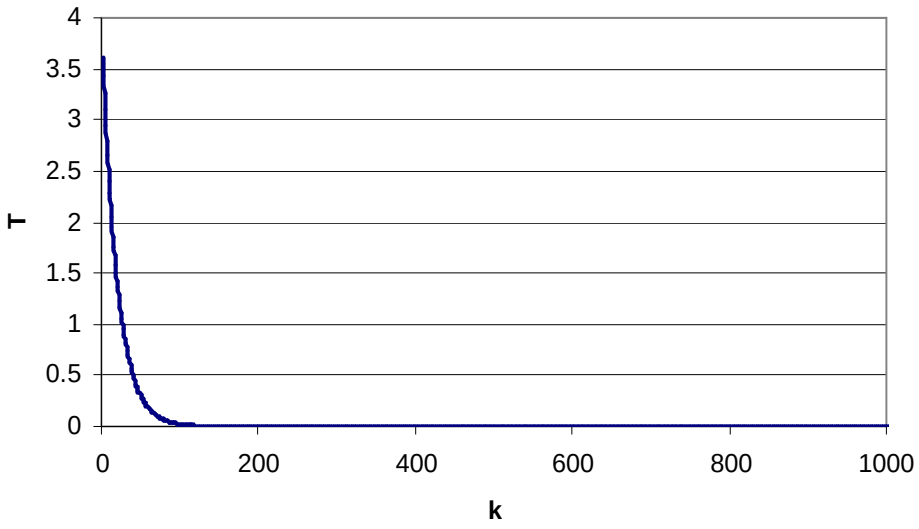



Figura 5.2-2
*Regla exponencial*

# 6. Esquema de Análisis

A lo largo de este capítulo se pretende dar una descripción del camino que el algoritmo desarrollado toma para obtener los resultados deseados en el tiempo deseado. No está contemplado mostrar lo que hace cada línea del código, pero sí envolverlo con una explicación formal que nos guíe por las etapas que el algoritmo pasa.

## 6.1 Planteamiento y Esquema General

El algoritmo presentado en esta tesis está basado en un esquema de recocido simulado que trabaja en tres niveles de resolución de la imagen original. El algoritmo comienza por definir las clases en las que se va a segmentar la imagen. En esta tesis todos los experimentos fueron sometidos a una segmentación de cinco clases, pero esto no significa que las herramientas empleadas no permitan la segmentación en un número menor o mayor de clases. Estas clases pueden ser escogidas definiendo ventanas que contengan niveles de gris representativos de la imagen. El proceso de clasificación implica un procesamiento de alto nivel que no se realiza en esta tesis. Lo que se realizan son segmentaciones pero, a fin de ilustrar mejor nuestra propuesta, se identifican a las regiones segmentadas como clases. Al segmentar una imagen se desea resaltar determinadas áreas eliminando cualquier ruido y distorsión en ellas. Por ello, las clases que se escojan se definirán calculando un promedio de los grises que se hayan obtenido en la ventana de interés. Este promedio define la clase. Al tomar una muestra de vegetación en una imagen de percepción remota no tenemos un mismo nivel de gris en toda el área, pero sí están todos éstos dentro de un rango reducido que permite ser representados por un promedio de ellos y distinguir las áreas de vegetación sin problemas en las imágenes segmentadas.

Una vez elegidas las clases, cada pixel es analizado con una serie de métodos probabilísticos para elegir la clase óptima en la que debe caer. Apoyado en la regla de Bayes, el Modelo de Potts y un esquema de recocido simulado, el algoritmo logra, después de una serie de iteraciones, obtener resultados óptimos de segmentación.

Regla de Bayes:

$$P(W_i|X) = \frac{P(X|W_i)\,P(W_i)}{P(X)},$$

siendo $X$ la imagen original y $W_i$ las clases con $i = 1,2,\ldots,5$ donde:

- $P(W_i|X)$ representa la probabilidad a posteriori.
- $P(W_i)$ es la probabilidad a priori de la clase $W_i$
- $P(X|W_i)$ es la probabilidad condicional de $X$ dado $W_i$
- $P(X)$ es la probabilidad total de $X$

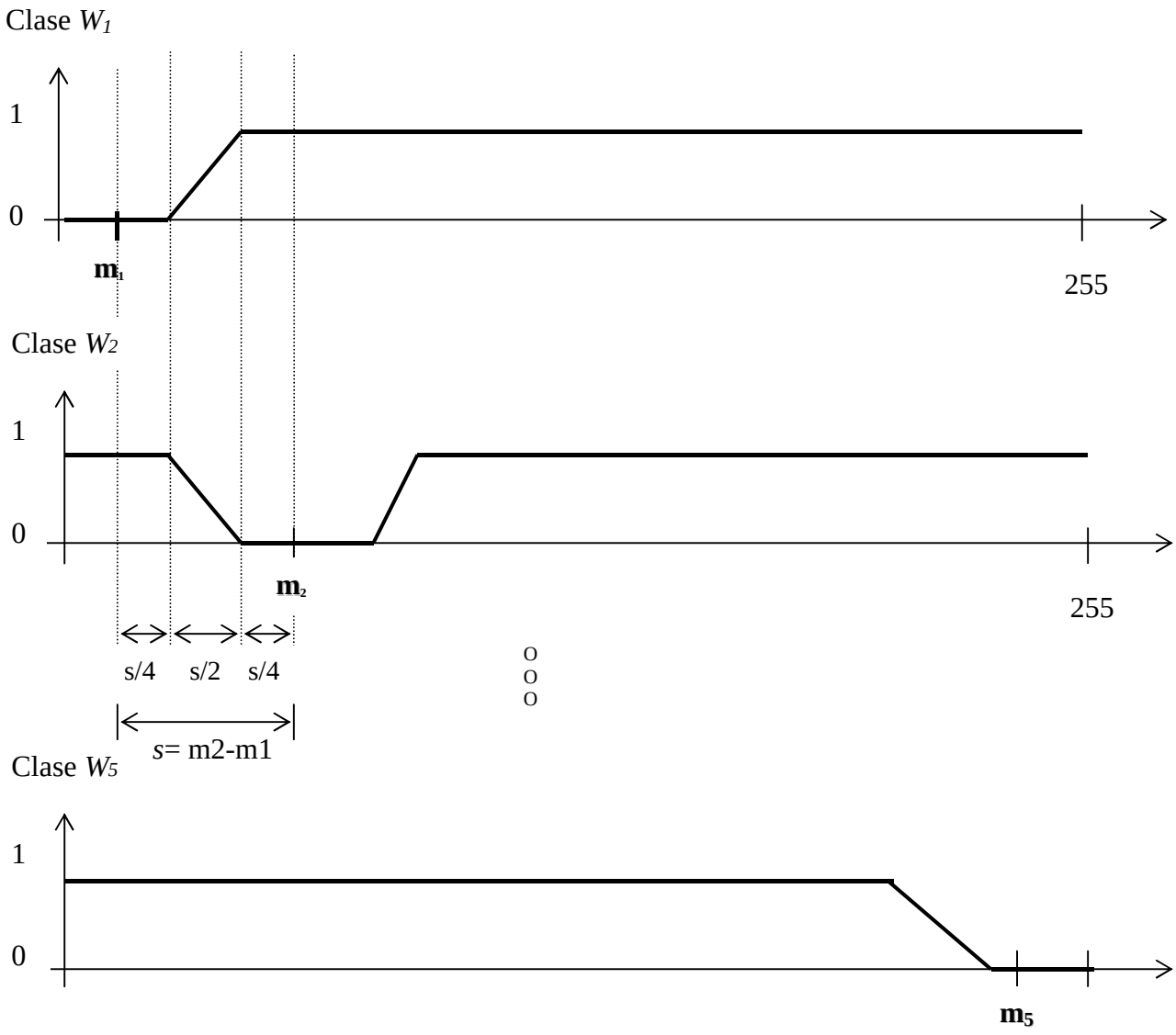
# 6.2 Funciones de Energía

Las funciones de energía son aquellas que surgen a partir de la definición de las clases que se van a utilizar para segmentar la imagen. Cada función de energía representa la probabilidad condicional de que un pixel tome el valor de la imagen original dado el valor de la clase que ésta represente. De esta forma se aproximan las probabilidades conjuntas del numerador de la regla de Bayes.

Una vez obtenidas las probabilidades condicionales se emplea la aproximación de la regla de Bayes $P(W_i|X) \approx P(X|W_i)\,P(W_i)$ para calcular la energía a posteriori:

$$U(W_i|X) \approx U(X|W_i) + U(W_i)$$

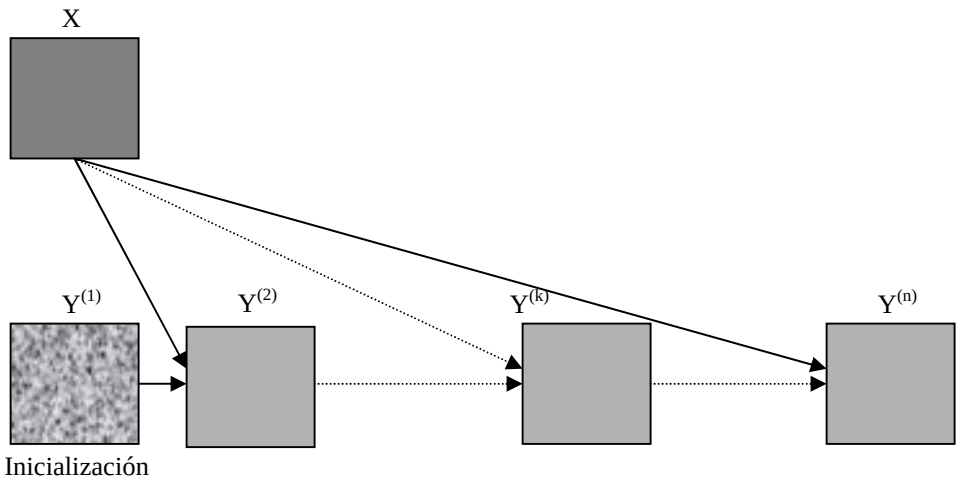
*TERMINOS DE ENERGIA* $U(X|W_i)$



En las gráficas anteriores ‘m’ se refieren a las medias obtenidas de las ventanas analizadas o simplemente el nivel de gris empleado para cada clase.

## 6.3 Implementación del Algoritmo de Recocido Simulado

Siendo X la imagen original (en niveles de gris) e Y la imagen segmentada (en este caso cinco clases), la simulación toma como datos iniciales a X y a la imagen de clases originada en la inicialización, $Y^{(1)}$. La figura siguiente muestra la esquematización del algoritmo de simulación, para el caso de *n* iteraciones:




La figura anterior muestra un esquema iterativo. Para la energía U(Y) se considera al modelo de Potts, el cual se aplica en vecindades en 8-conexidad. La energía $U(X|w_i)$ se evalúa sobre la imagen original X.

Para la inicialización del algoritmo, se realiza una determinación aleatoria de la solución $Y^{(1)}$ (la clase de cada pixel se determina mediante un generador de números aleatorios, con distribución uniforme, en el rango del número de clases).

Se escoge una temperatura inicial $T^{(1)}$ suficientemente elevada ($T^{(1)}$ = 2.5) y se barre secuencialmente la imagen.

En cada etapa (k), se escoge un pixel g de coordenadas [i][j]. Su valor es $Y_g^{(k)}$ y su configuración de vecindad es $V_g^{(k)}$. Se efectúa el sorteo de una variable aleatoria (clase "elección”) con distribución uniforme: $\xi \in \Omega$ .

Para el Criterio de metrópolis y el Modelo de Potts:

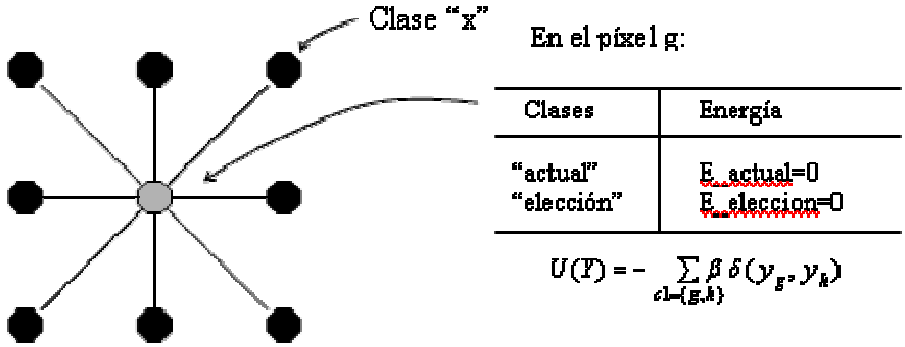



Fig. 2. Vecindad $V_g^{(k)}$ asociada al píxel g (al centro de la ventana en 8-conexidad).

## 6.4 Segmentación en Baja Resolución

Cuando se pretende trabajar con una imagen de baja resolución, el procesamiento al que va a ser sometida dicha imagen no se ve afectado por el tiempo en que tarde en entregar resultados. Siendo una imagen pequeña, la cantidad de pixeles a analizar es baja y la tardanza del algoritmo no es factor importante en la entrega de resultados. Por lo tanto, en este caso no es utilizado el esquema multiescala y la imagen es procesada únicamente en su tamaño original, es decir todas las iteraciones por las que pasa, la analizan sin haber sufrido alteraciones en los pixeles originales.

La determinación de una imagen de baja resolución dependerá del equipo empleado para hacer el procesamiento, puesto que, como ya se mencionó, lo que nos interesa es una entrega rápida de resultados. Así, mientras para determinado equipo el análisis de una imagen bajo este algoritmo puede rebasar los 5 minutos, para otro, con una capacidad de procesamiento mucho más elevada, puede hacerlo en menos de un minuto. El hardware con el que trabajemos será factor crucial que refleje la eficiencia del algoritmo. Como se muestra en los resultados de esta tesis, el esquema multiescala entrega resultados con menor tardanza, pero para imágenes con baja resolución esta diferencia es insignificante.

## 6.5 Proyección de Regiones Segmentadas

El fundamento del esquema multiescala es trabajar con distintas resoluciones de la imagen original. Cuando se trabaja con una imagen de alta resolución, cualquier procesamiento al que se pueda someter puede tardar mucho. La idea es reducir la resolución de la imagen para trabajar con éstas y disminuir el procesamiento. La imagen de baja resolución puede someterse a determinada cantidad de iteraciones y el resultado de estas es proyectado como una imagen de resolución media, duplicando el área de cada pixel (de 1x1 a 2x2). Esta imagen de resolución media se somete nuevamente a igual o distinta cantidad de iteraciones, y se procede de igual forma duplicando la resolución de la imagen para proyectarla en una de tamaño original.

## 6.6 Segmentación en Alta Resolución

El esquema multiescala fue diseñado con el objetivo de tener un rápido procesamiento en imágenes de gran tamaño. A partir de una imagen en su tamaño original, se procesa para obtener de ella niveles con menor resolución. En este caso se trabaja con tres niveles: el original (n x n), resolución media (n/2 x n/2) y de baja resolución (n/4 x n/4). De esta forma el total de iteraciones a las que se ve sometida la imagen son distribuidas entre los diversos niveles. Como ya se explicó antes, a menor resolución mayor es la rapidez en la que se entregan los resultados.

Por ejemplo, envés de trabajar con 40 iteraciones en la resolución original de la imagen, ésta se somete a 10 iteraciones en baja resolución, 10 en resolución media y 20 más en la resolución original. Más adelante se muestra en los resultados como el tiempo de procesamiento es mucho menor bajo el esquema multiescala.

# 7. Resultados

## 7.1 Imágenes de Prueba

Una vez programado, el algoritmo se puso a prueba para distintos escenarios. Para los siguientes experimentos se utilizaron imágenes que se caracterizaran en diferentes sentidos. En algunas, las expresiones faciales de las personas en las imágenes era el objetivo a seguir. El interés radicaba en poder mantener las mismas expresiones que los rostros pudieran externar una vez aplicado el algoritmo. En éstas, los niveles de gris en una cara son muy similares, pero si no se hace una diferenciación adecuada entre ellos, se mantiene un mismo nivel en todo el rostro y se pierde por completo el mensaje que se pueda comunicar a través de él. Por ello, no en todos los experimentos se utilizaron ventanas para definir las clases. En el segundo experimento, por ejemplo, los niveles de gris que determinan las cinco clases fueron obtenidos empíricamente, escogiendo los niveles óptimos que permitieran una mejor definición de la imagen. Para el resto de los experimentos esto no fue necesario, ya que las áreas de interés en las imágenes mantienen distancias relativamente grandes entre ellas a lo largo de la escala de grises.

### 7.1.1 Imágenes Sintéticas

En los tres experimentos siguientes se trabajó con imágenes sintéticas que pretenden evaluar el algoritmo en dos características relacionadas con la fidelidad original de la imagen: homogeneidad y bordes. Para los tres experimentos se utilizaron imágenes originales sometidas a ruido gaussiano con media 0 y varianza distinta para cada uno de ellos (0.01, 0.05 y 0.2).

Cuando nos referimos a la homogeneidad, nos interesa analizar si las áreas que pertenecen a determinada clase, son fieles a esta en toda su extensión. Cuando se presenta cualquier tipo de ruido en la imagen, éste evita que se aprecien áreas totalmente homogéneas. En lo referente a los bordes, nos interesa analizar que las fronteras existentes en la imagen, en cuanto a clases se refiere, se respeten fielmente. Es deseado obtener, después de haber sido sometida la imagen al algoritmo, bordes que se apeguen lo más posible a los originales.

Lo ideal sería gozar de ambas características al final del procesamiento, sin embargo, como los experimentos lo muestran, no siempre se logra eliminar el ruido por completo. En algunos procesos, después de una serie considerable de iteraciones, el resultado muestra un comportamiento ruidoso. Sin embargo, estas mismas respetan los bordes de la imagen original, es decir, las fronteras entre una clase y otra permanecen fieles. Por otro lado, en los procesos en los que logramos eliminar de manera considerable el ruido y las áreas se muestran homogéneas, los bordes no siguen las originales (en este caso líneas rectas).

Como ya se explicó anteriormente, el procesamiento consiste en someter la imagen a tres etapas de manipulación. Estas tres etapas permiten un menor tiempo de procesamiento, sometiendo la imagen a varias iteraciones en dimensiones menores antes de hacer el análisis en el tamaño original. Sin embargo, éste no es el único propósito de este modelo multiescala. Jugar con el número de iteraciones en cada escala del procesamiento otorga la posibilidad de priorizar entre las dos características de fidelidad antes mencionadas.

Ante el mismo número de iteraciones totales obtenemos resultados diferentes. Si las iteraciones en la primera etapa, cuando la imagen tiene dimensiones 2 veces menor a la original, son las de mayor cantidad, gozamos de homogeneidad fiel. Si por el contrario, el mayor número de iteraciones recae en la tercera etapa, la que trabaja con las dimensiones originales, los bordes (líneas rectas para la imagen de los experimentos con imagen sintética) se respetan.

Los tres experimentos muestran este comportamiento, sin embargo el segundo lo trata más específicamente. En éste, a partir de una imagen con ruido gaussiano de varianza igual a 0.05 agregado, se hacen 90 iteraciones en 4 de los procesos. Estas 90 iteraciones son distribuidas de distinta forma para cada uno de ellos. Los sistemas fueron: 10+10+70, 70+10+10, 10+70+10, y 30+30+30. En el primer caso gozamos de bordes bien definidos, ya que la mayor parte del análisis recae en la última etapa. Sin embargo las áreas que deberían mantener fidelidad a determinada clase, a pesar de que sí existe una clara distinción entre éstas, se notan ruidosas, heterogéneas. En el segundo las áreas están casi limpias de ruido, es decir son casi totalmente homogéneas, pero sus bordes se perciben altamente alterados. En los últimos dos casos, existe cierto equilibrio entre las dos características de fidelidad. No gozan de una homogeneidad como la del segundo caso, pero no existe un nivel de ruido tan elevado como en el primer caso. Sus bordes no están tan alterados como en el sistema 70+10+10, pero las líneas rectas siguen sin ser respetadas en su totalidad.

El primer experimento nos muestra como el ruido es factor determinante para obtener una imagen resultante mucho más fiel a la original. La imagen bajo la configuración 20+20+20 logra una similitud importante a la original.

Las últimas dos muestras del experimento 3 son interesantes. Nótese que en una de ellas se aplica un sistema 80+10+10. Sin embargo el factor de homogeneización toma un efecto contraproducente. De las cinco clases originales, los resultados muestran únicamente 3 finales. Es decir hay 2 pares de niveles de gris que se fusionan.

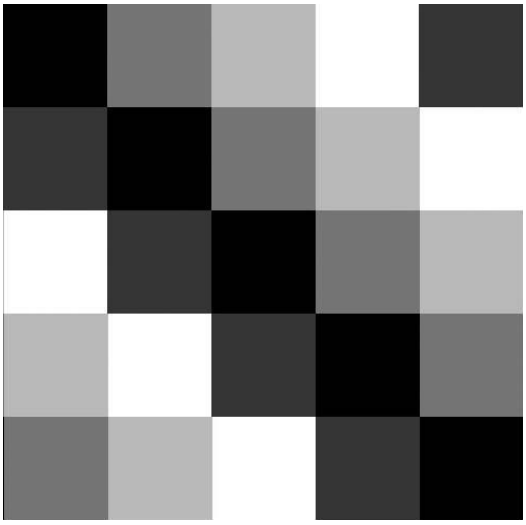
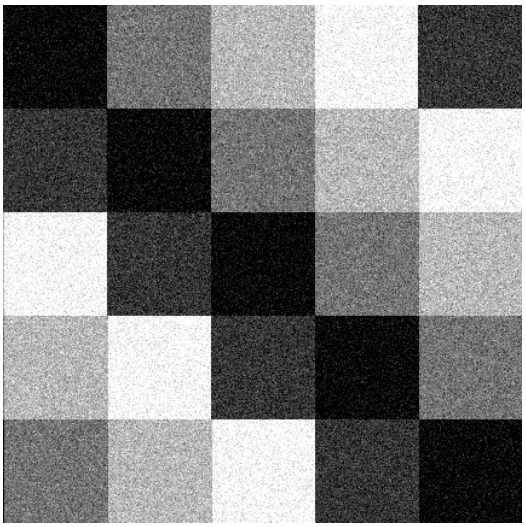
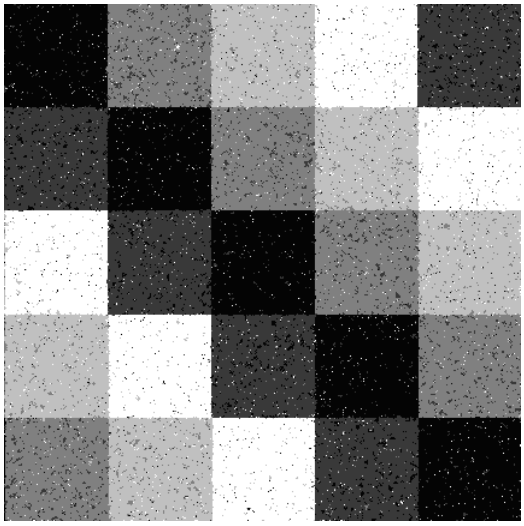


Imagen original

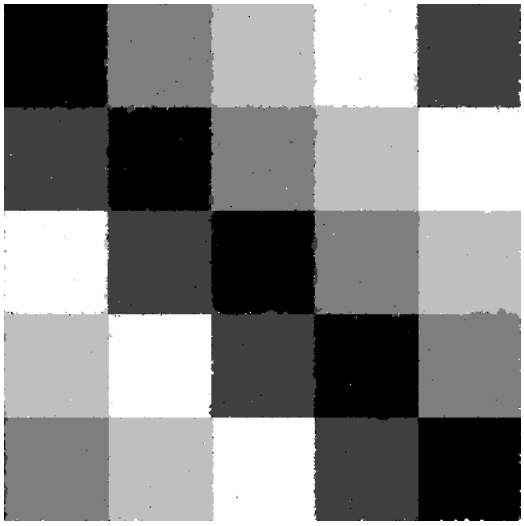
Experimento 1



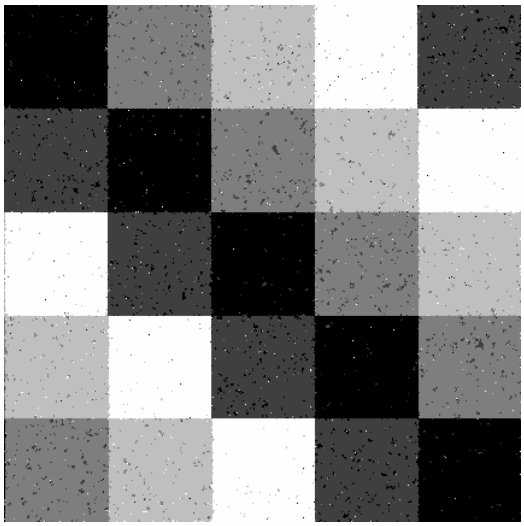
Original con ruido gaussiano. var = 0.01



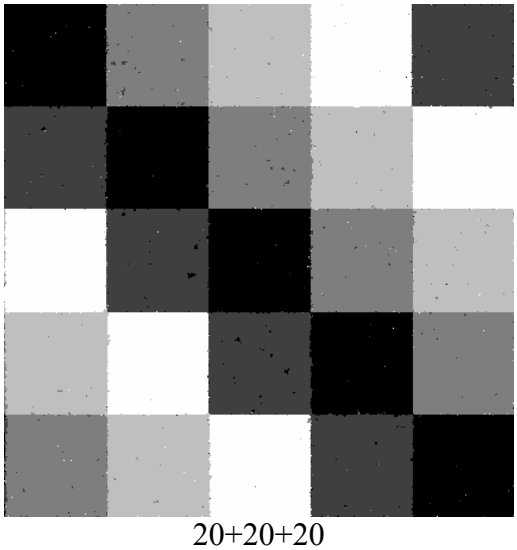
10+10+20



40+10+10

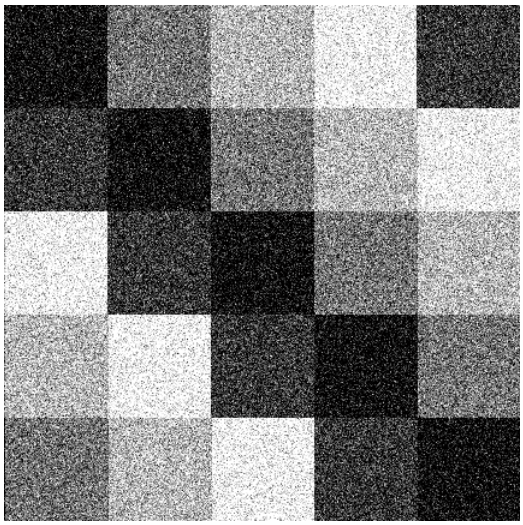


10+10+40

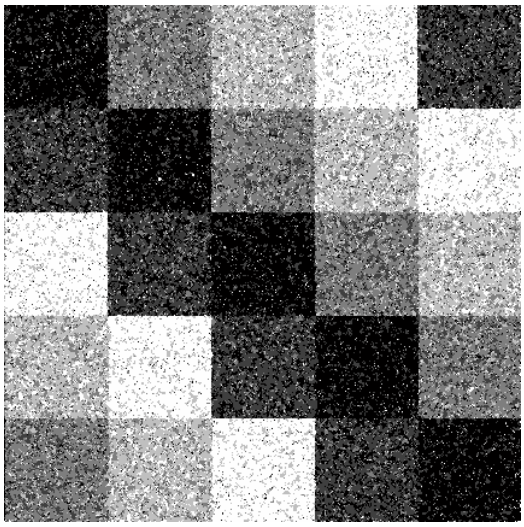


20+20+20

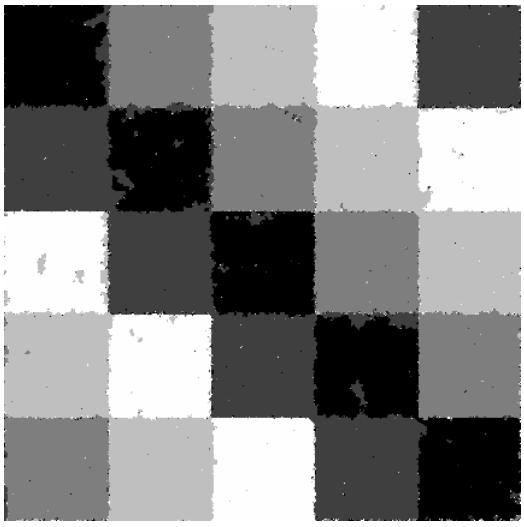
Experimento 2



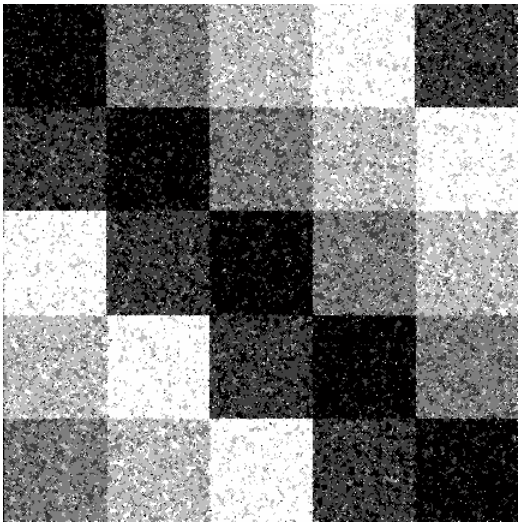
Original con ruido gaussiano. var = 0.05



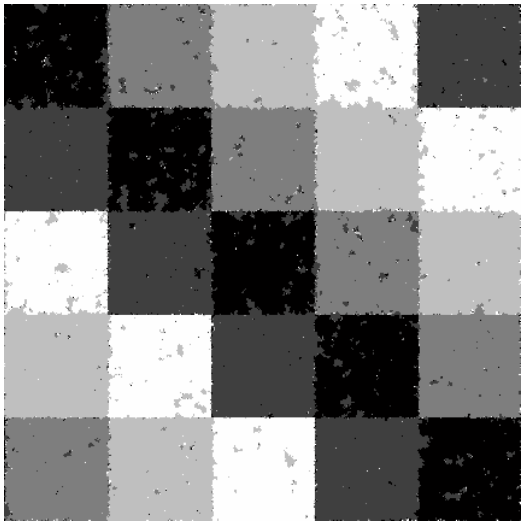
10+10+20



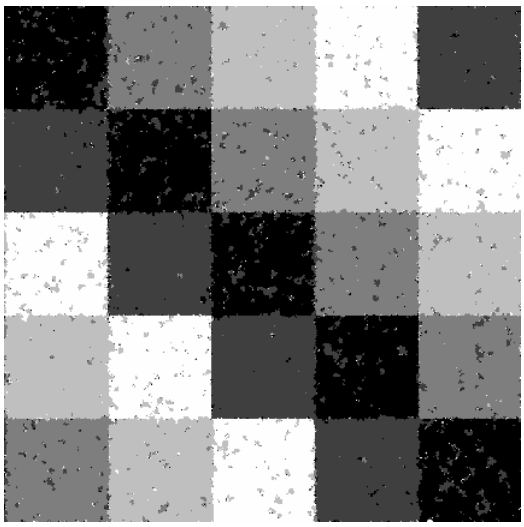
70+10+10



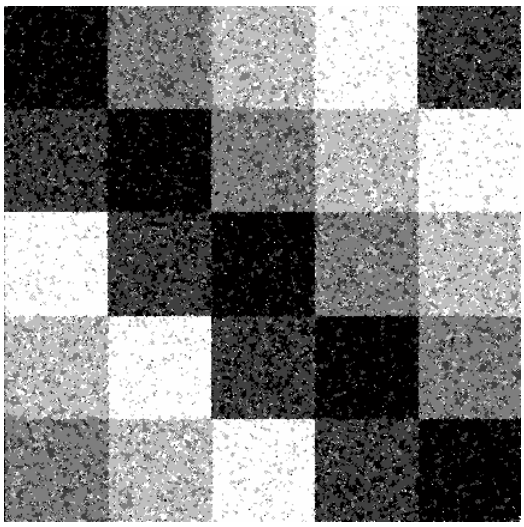
10+10+40



30+30+30

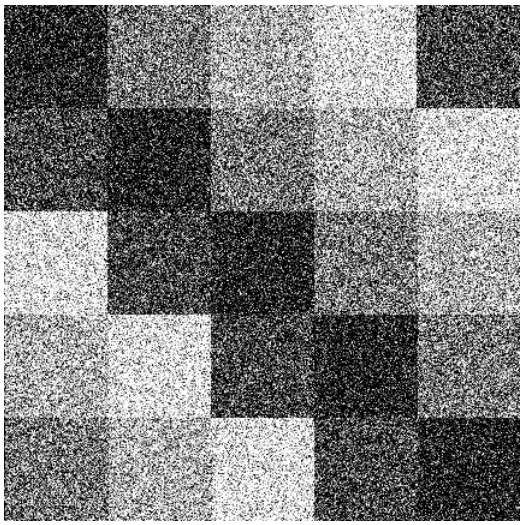


10+70+10

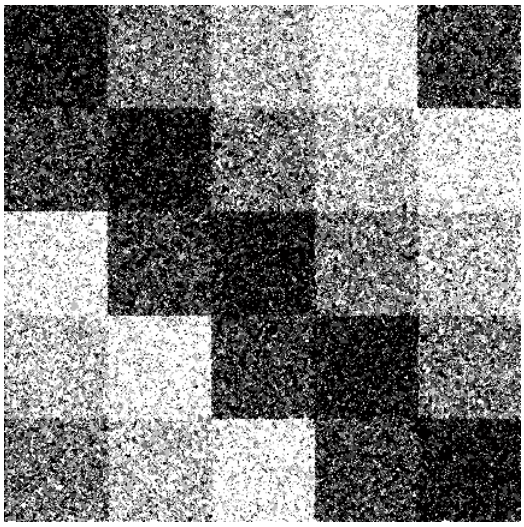


10+10+70

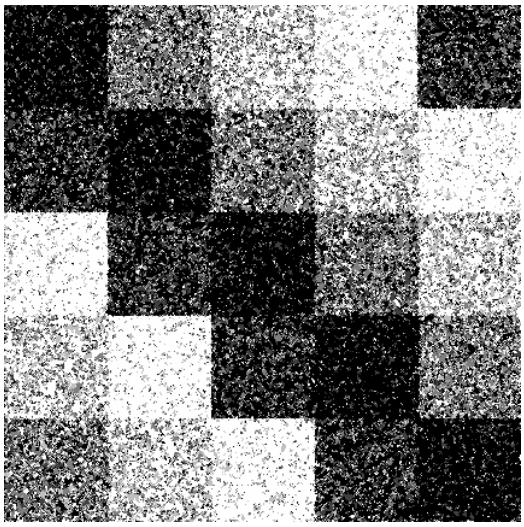
Experimento 3



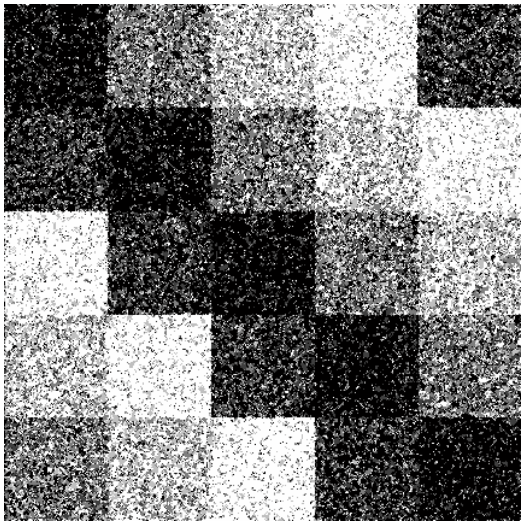
Original con ruido gaussiano. var = 0.2



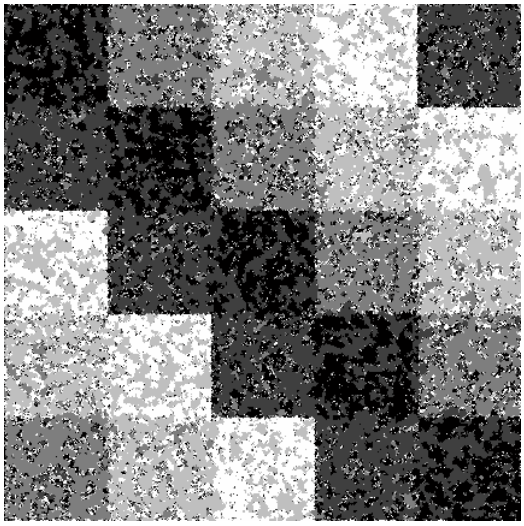
10+10+20



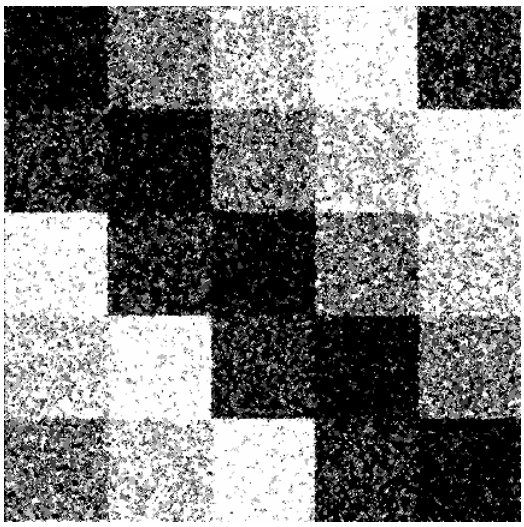
10+10+40



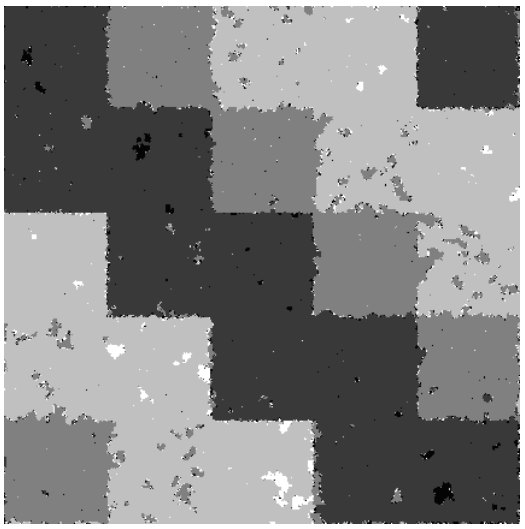
10+10+30



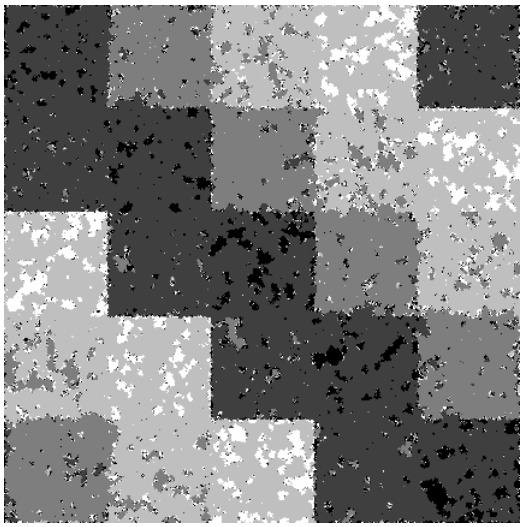
20+20+60



10+10+80



80+10+10



30+30+30

### 7.1.2 Escenas Naturales

Experimento 1

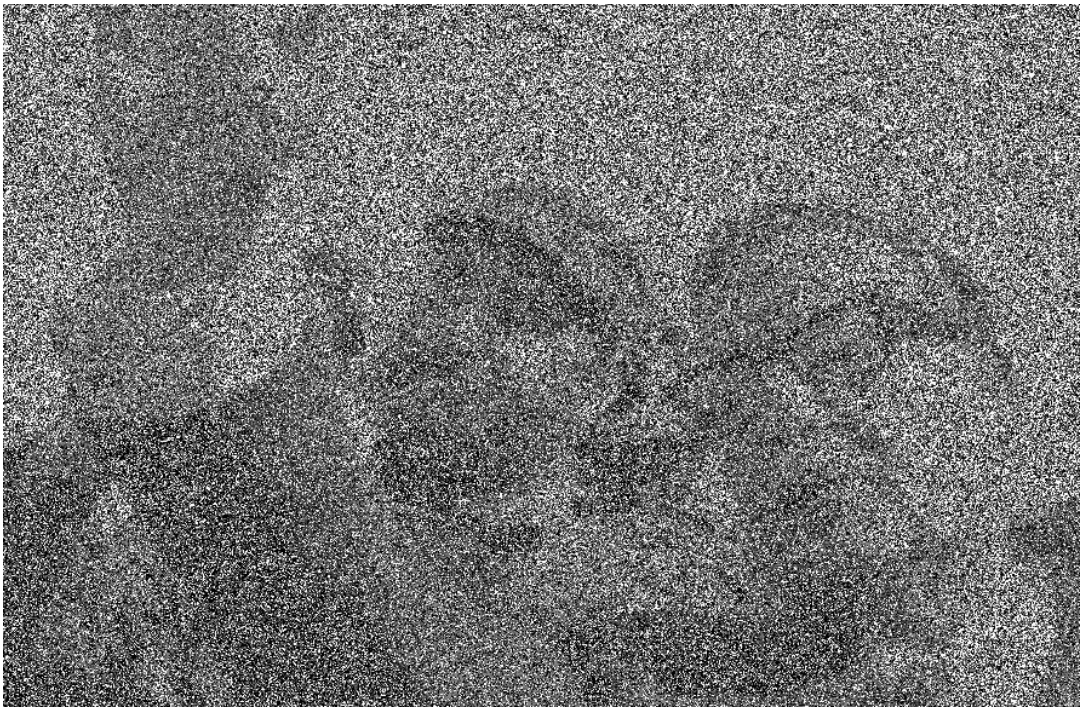
La imagen de prueba fue sometida a una única etapa del algoritmo en la que no existe proceso de multiescala. Las iteraciones se realizaron desde la primera, con el tamaño original de la imagen. Este primer experimento pretende mostrar la relación que existe entre el número de iteraciones y la fidelidad con la que se presentan los resultados. En este caso, a partir 19 iteraciones, se empieza a distinguir la similitud con la imagen original. Las imágenes siguientes muestran claramente como el ruido total de la fase de inicialización desaparece paulatinamente, a través de las iteraciones hasta alcanzar un nivel máximo de fidelidad. Existirá un nivel en el que la imagen ya no cambie aunque se someta a más iteraciones, ya que ha alcanzado la mayor fidelidad posible de acuerdo a lo permitido por el algoritmo. Es importante resaltar nuevamente que la imagen original está basada en un campo de 256 niveles de gris y los resultados trabajan con únicamente 5 niveles. A pesar de ello, el resultado final, en este caso después de 60 iteraciones, nos otorga un entendimiento total del contenido de la imagen, lográndose apreciar incluso detalles relativamente pequeños.

Las gráficas mostradas después de las imágenes resultantes, nos dan una idea de los campos en los que cada tono de gris va a caer dependiendo de su cercanía a la clase correspondiente. Estas gráficas son empleadas en el algoritmo para determinar la clase óptima que cada nivel de gris en la imagen debe tomar.

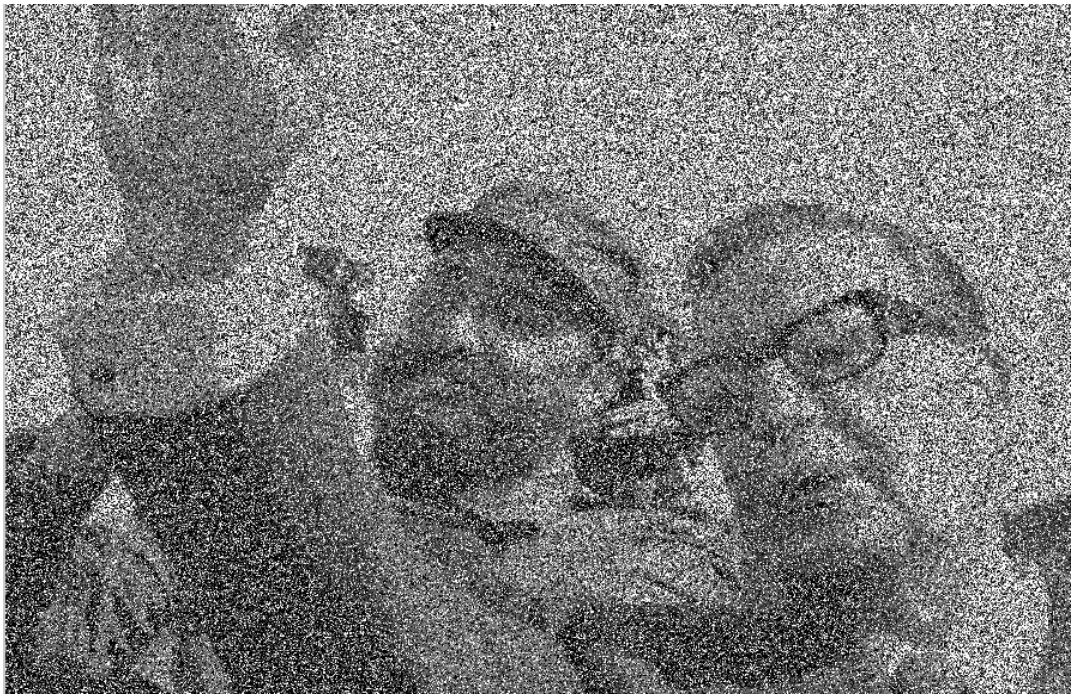
Imagen original:

Tamaño: 1422 x 2048 pixeles

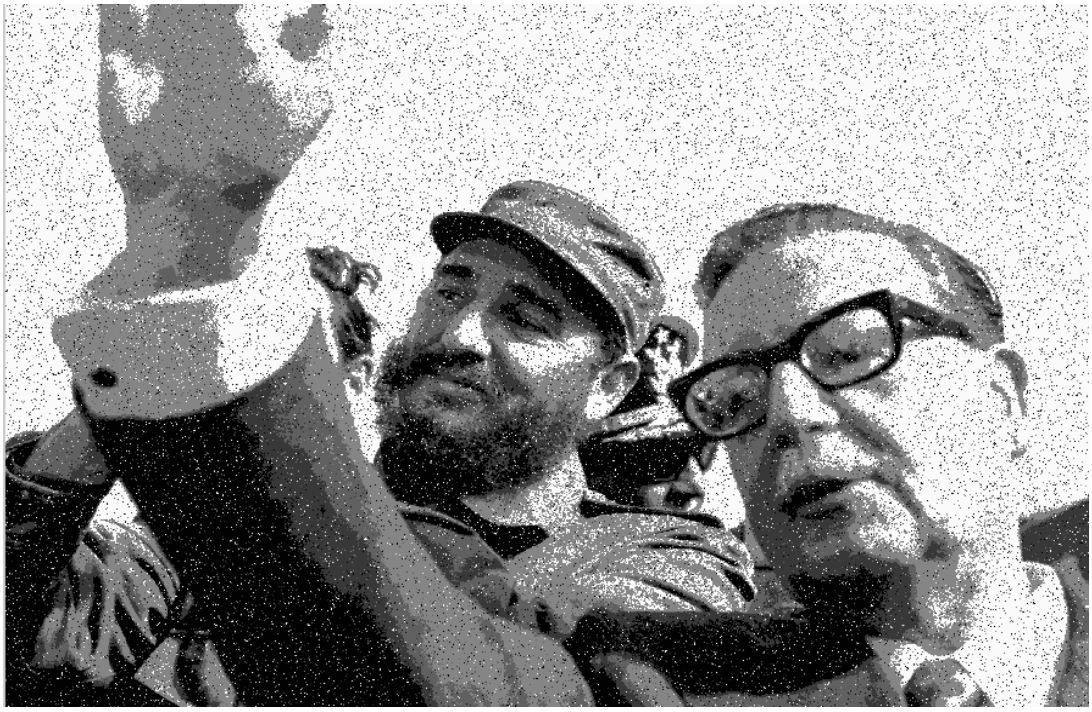
Después de 19 iteraciones:



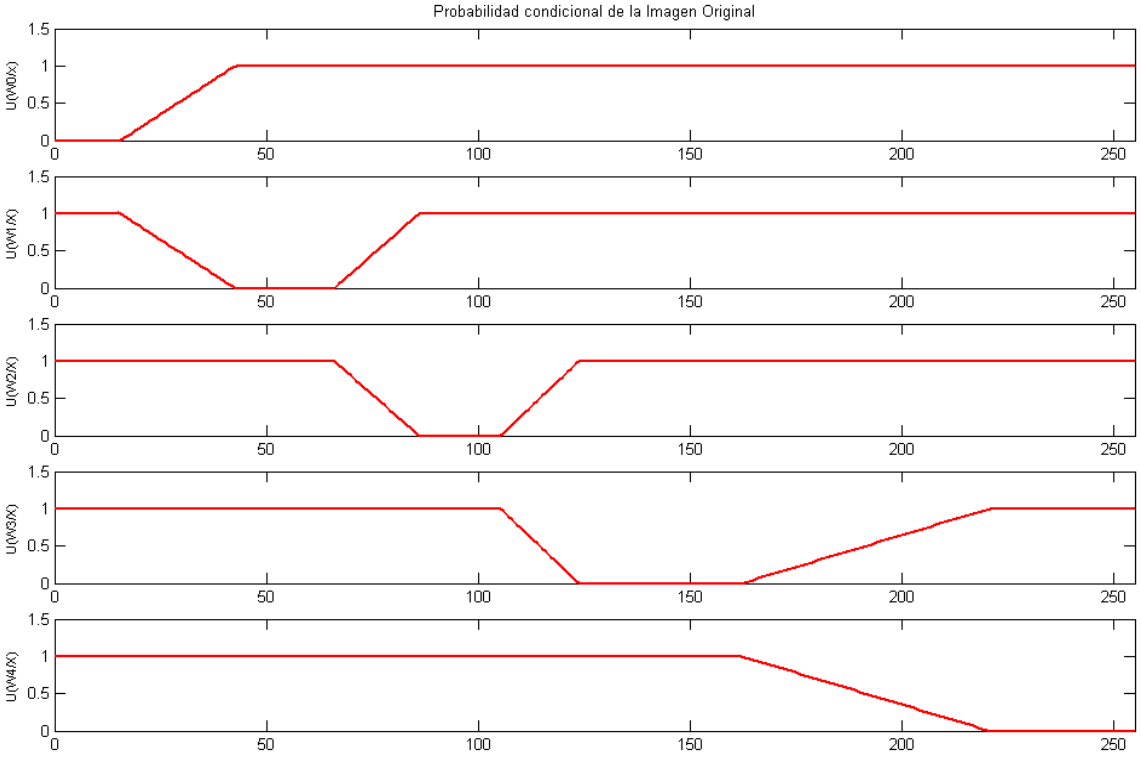
Después de 20 iteraciones:



Después de 30 iteraciones:



Después de 60 iteraciones:

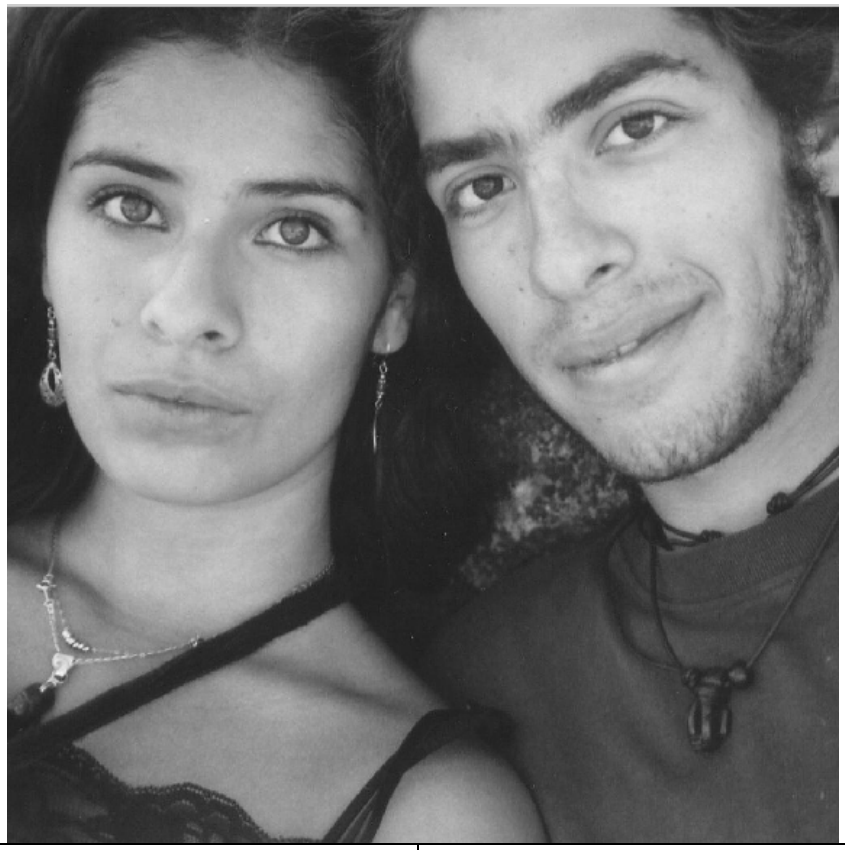



Experimento 2

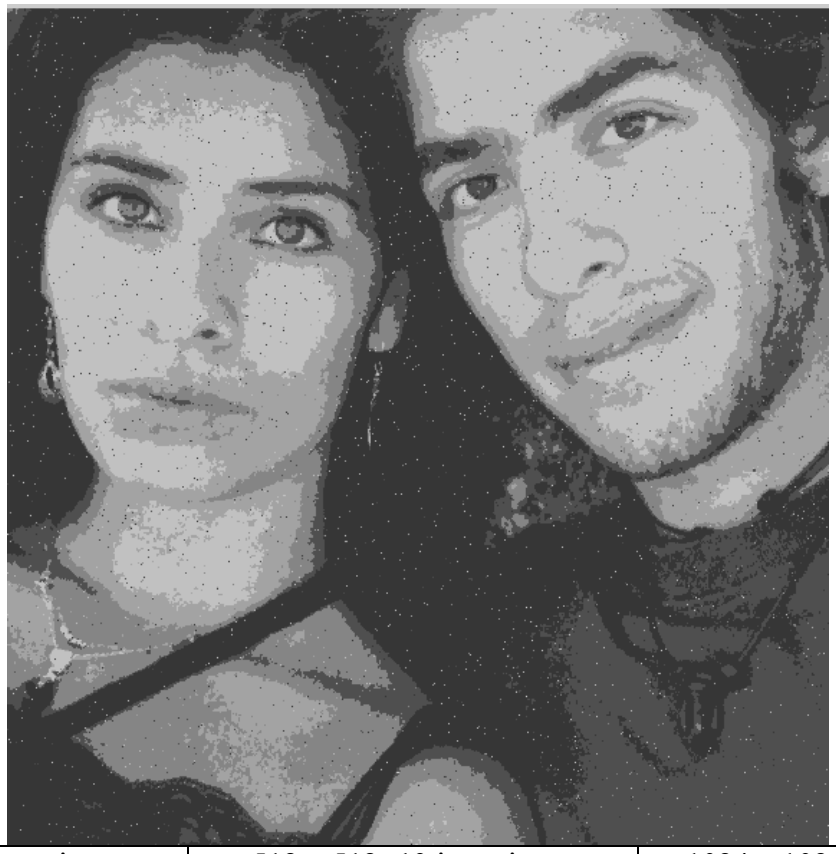
Para este segundo experimento ya se utiliza el esquema multiescala. Se trabajó con tres tamaños de la imagen: la original de 1024 x 1024, una segunda etapa de 512 x 512 y la tercera de 256 x 256 pixeles. A partir de la primera muestra (10+10+20) se tiene un entendimiento total de la imagen. Sin embargo, con las siguientes imágenes nos podemos percatar de la eliminación del ruido que en esta primera claramente existe.

A partir de un total de 50 iteraciones, se puede decir que la imagen alcanza su nivel máximo de fidelidad. Existen otras dos muestras más; una bajo 60 iteraciones y la otra para 70. Lo único que cambia a partir de que se alcanza este equilibrio, es una variación leve en los contornos de las figuras que otorga cada clase para conformar la imagen total. Si observamos detalladamente, existen pequeñas diferencias entre las distintas muestras finales.

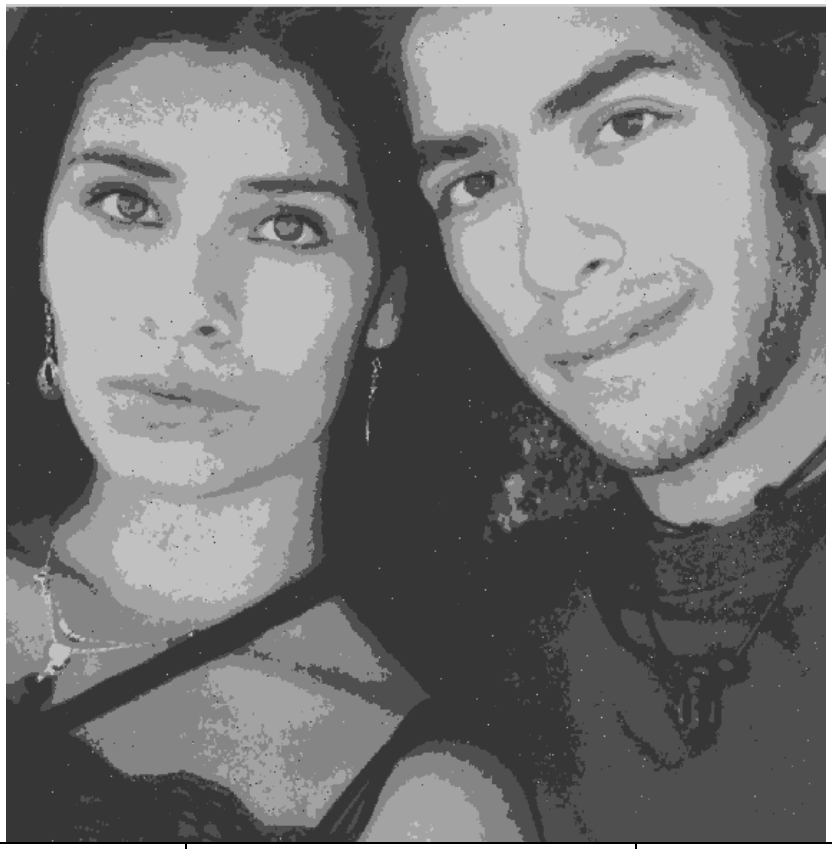
Las últimas dos muestras (de un total de 70 iteraciones cada una) muestran la diferencia que representa distribuir de manera distinta el número de iteraciones entre las etapas de procesamiento. Como ya se mencionó anteriormente, esta diferenciación otorga resultados interesantes. En el caso del sistema 50+10+10, es fácilmente perceptible que gozamos de contornos mucho más suaves que en el sistema 10+10+50. Sin embargo esto obliga a perder detalles en la imagen por la incapacidad de manejar áreas muy pequeñas. En el primero carecemos de áreas punteadas que otorgan sensaciones de tercera dimensión, sombras y volúmenes, que dan una impresión más realista. Esto no significa que una sea mejor que la otra, simplemente que tenemos distintos resultados que pueden ser explotados de acuerdo a las necesidades.



| | |
|---|---|
| Imagen original | Tamaño: 1024 x 1024 pixeles |



| 256 x 256: **10** iteraciones | 512 x 512: **10** iteraciones | 1024 x 1024: **20** iteraciones |
|---|---|---|



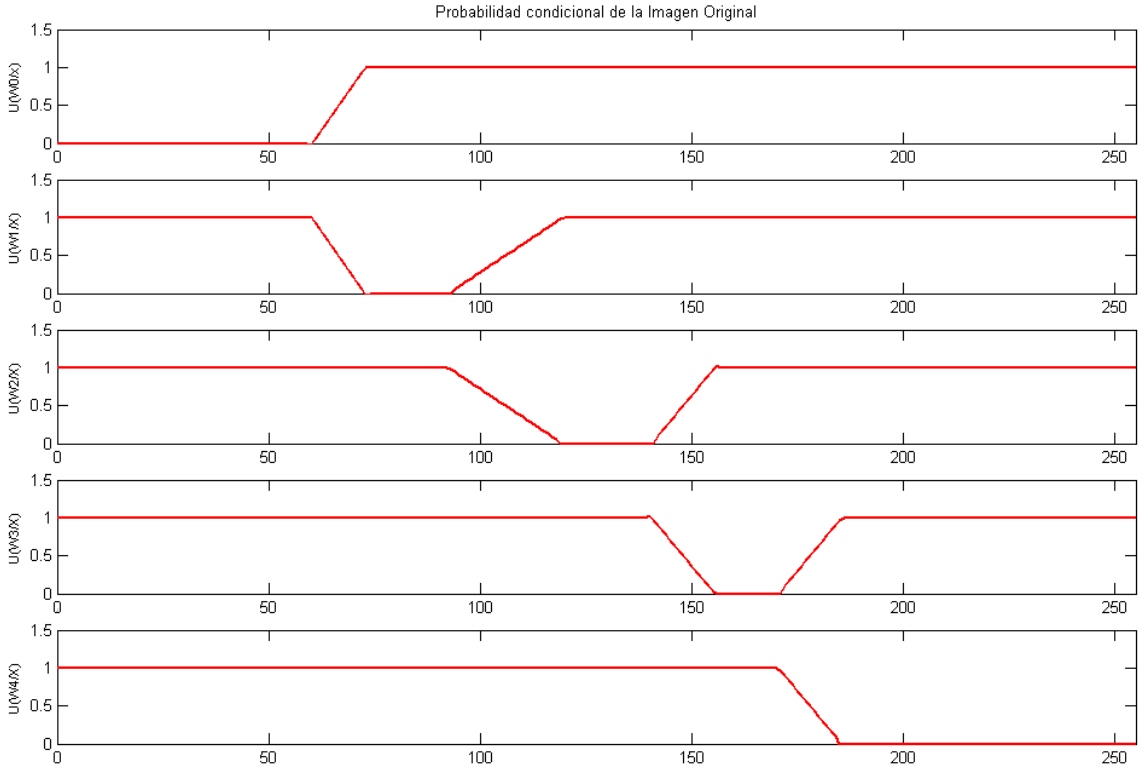
| 256 x 256: **10** iteraciones | 512 x 512: **10** iteraciones | 1024 x 1024: **30** iteraciones |
|---|---|---|

| 256 x 256: **10** iteraciones | 512 x 512: **10** iteraciones | 1024 x 1024: **40** iteraciones |
|---|---|---|

| 256 x 256: **10** iteraciones | 512 x 512: **10** iteraciones | 1024 x 1024: **50** iteraciones |
|---|---|---|

| 256 x 256: **50** iteraciones | 512 x 512: **10** iteraciones | 1024 x 1024: **10** iteraciones |
|---|---|---|

| 256 x 256: **30** iteraciones | 512 x 512: **10** iteraciones | 1024 x 1024: **10** iteraciones |
|---|---|---|




Experimento 3

Los resultados presentados en este tercer experimento están enfocados a imágenes de percepción remota. En sí, el algoritmo elaborado para esta tesis tiene como principal área de aplicación este tipo de imágenes ya que son imágenes con un nivel muy alto de resolución.

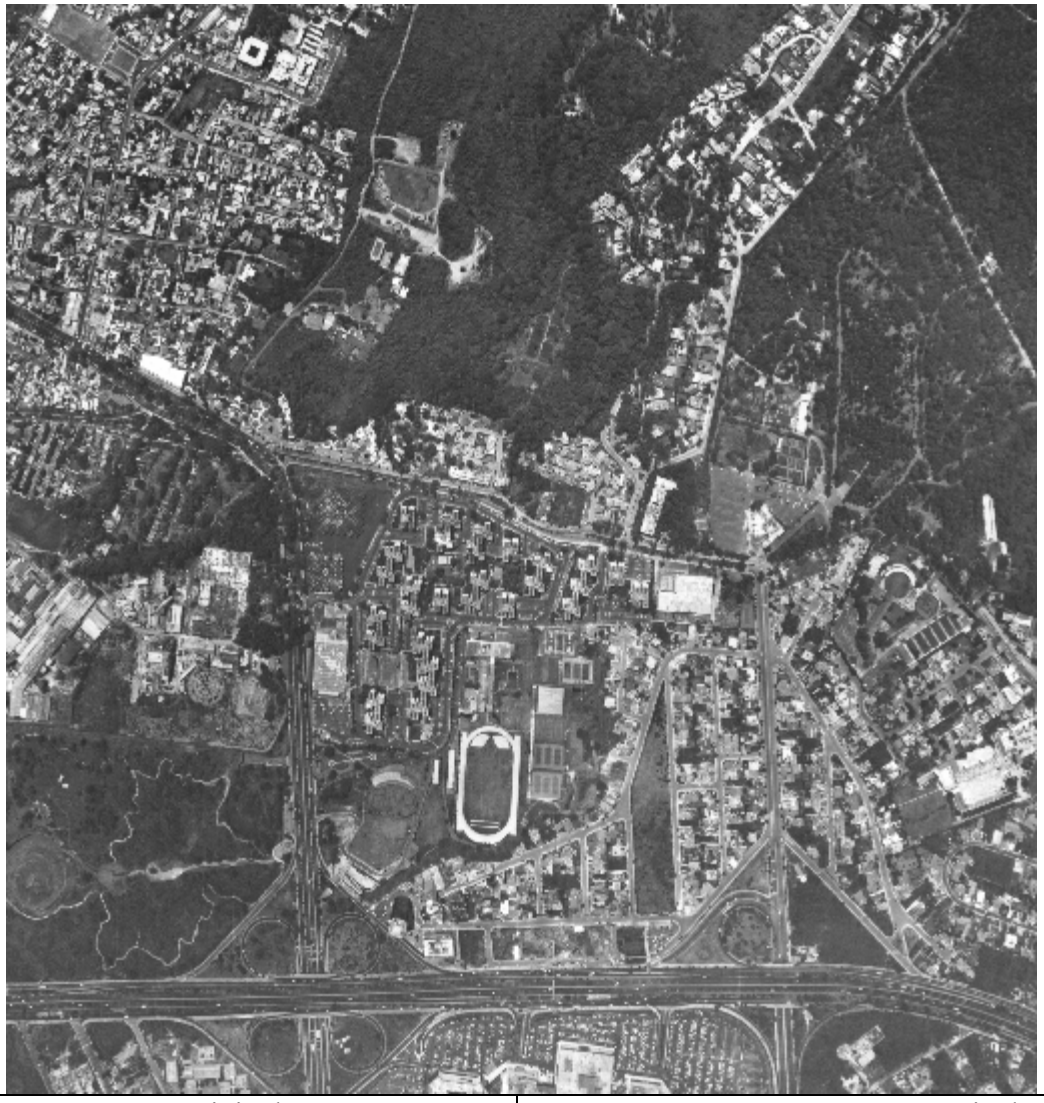
La característica de interés en imágenes de percepción remota es lograr identificar distintos tipos de suelo, y en particular en una de carácter urbano nos interesa identificar calles, áreas verdes, y distinguir entre distintos tipos de edificaciones. Si se logra esto, detalles de la imagen que sean más específicos caen fuera de nuestra área de interés y pueden ser considerados como despreciables al momento de someterla a fases de segmentación.

De las imágenes mostradas a continuación se obtienen algunas conclusiones interesantes. Los resultados son contundentes, demostrando que con sólo cinco niveles de gris es suficiente para tener una comprensión totalmente aceptable de la imagen. Nuevamente se comprueba que la distribución de iteraciones entre las tres escalas se puede manipular dependiendo de los factores que nos interese atacar. Para las imágenes en las que se tiene una mayor carga de iteraciones en la primera etapa, logramos erradicar casi por completo el ruido, pero las fronteras entre los distintos usos de suelo no se respetan fielmente. Por otro lado, en los experimentos en los que se cargan más iteraciones en la segunda o tercera etapa no se elimina el ruido totalmente, es decir no se goza de áreas cien por ciento homogéneas, pero las fronteras se mantienen más similares a las originales.

En lo referente al tiempo de procesamiento, este experimento nos permite llegar a conclusiones de otro carácter. Los resultados obtenidos para la última imagen (fachada de una casa) las apoyan. El tiempo de procesamiento para las imágenes de percepción remota a partir de 1500 x 1500 pixeles de

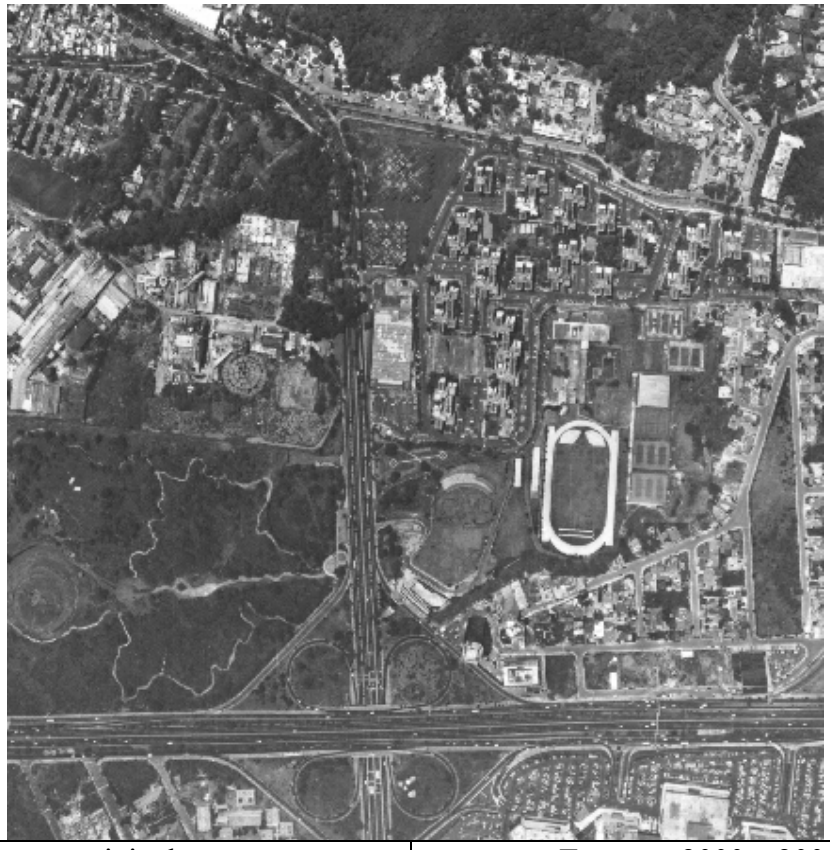
resolución, rebasa los 30 minutos para aquellas que otorgan 10 iteraciones a la última etapa. Sin embargo, el análisis de la imagen de la casa que tiene un tamaño de 1544 x 1544 pixeles, dura alrededor de cinco minutos. Esto nos demuestra que, bajo este algoritmo, la resolución no es la única característica determinante para el tiempo de procesamiento de una imagen.

Otra característica esencial de las imágenes de percepción remota de zonas urbanas es la heterogeneidad que existe entre los diversos niveles de gris desplegados a lo largo de éstas. A menos que se presenten extensas áreas verdes, entre las calles y las edificaciones existe un contraste significativo de tonalidad. Esta característica obliga al algoritmo a realizar un procesamiento más prolongado, ya que en el caso de zonas homogéneas las decisiones para elegir una determinada clase se toman mucho más rápido, ya que se realiza un menor número de ciclos de procesamiento.

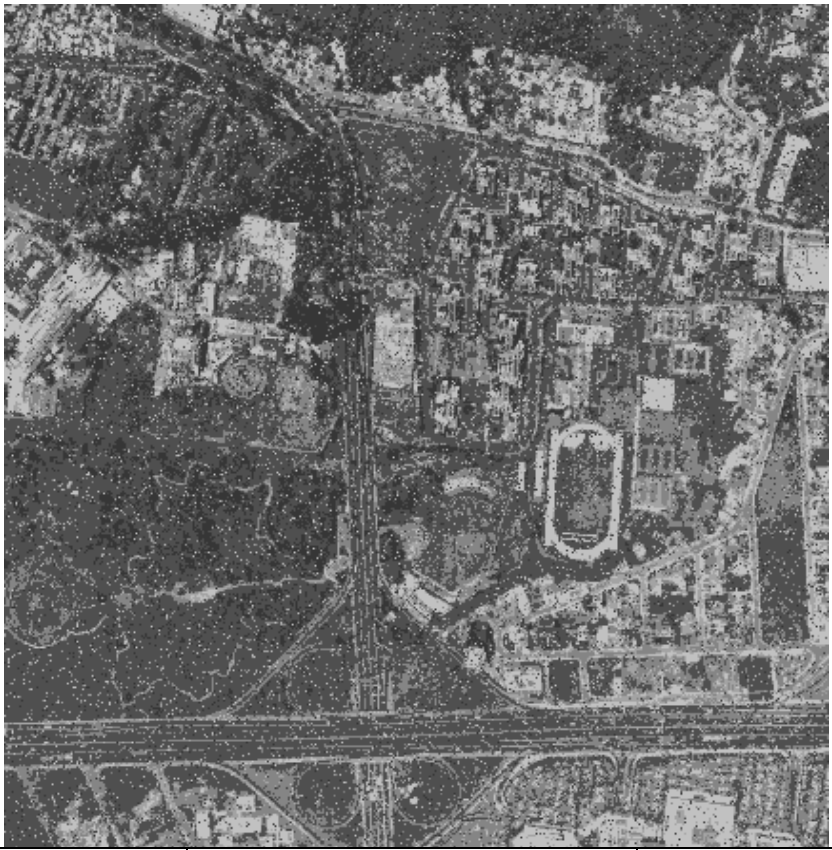


| | |
|---|---|
| Imagen original | Tamaño: 3000 x 3000 pixeles |

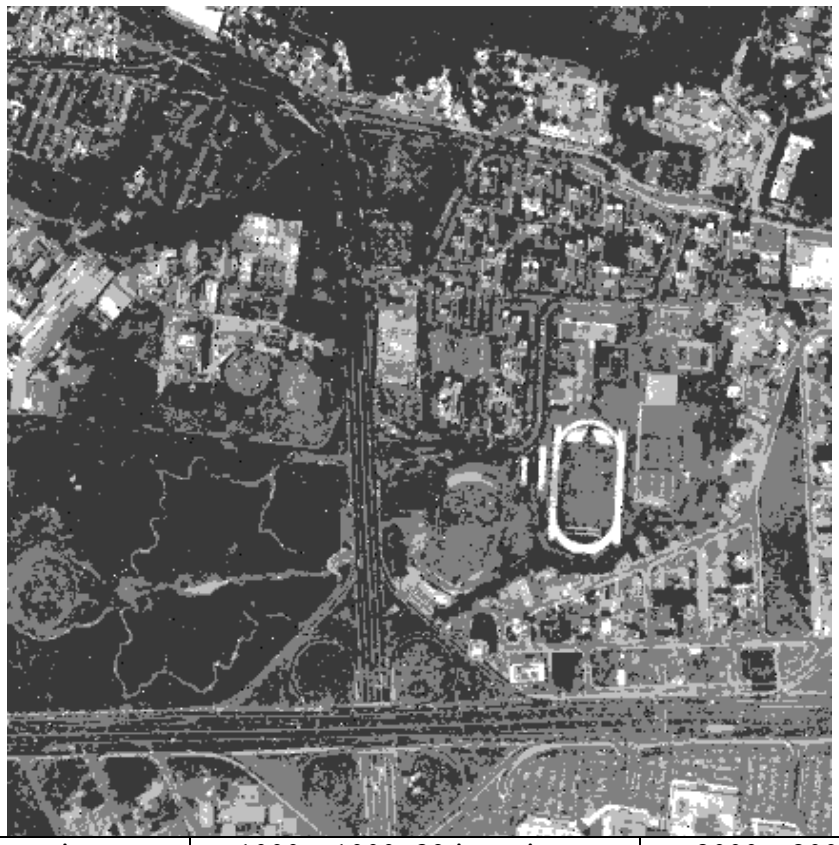
| 750 x 750: **10** iteraciones | 1500 x 1500: **10** iteraciones | 3000 x 3000: **10** iteraciones |
|---|---|---|



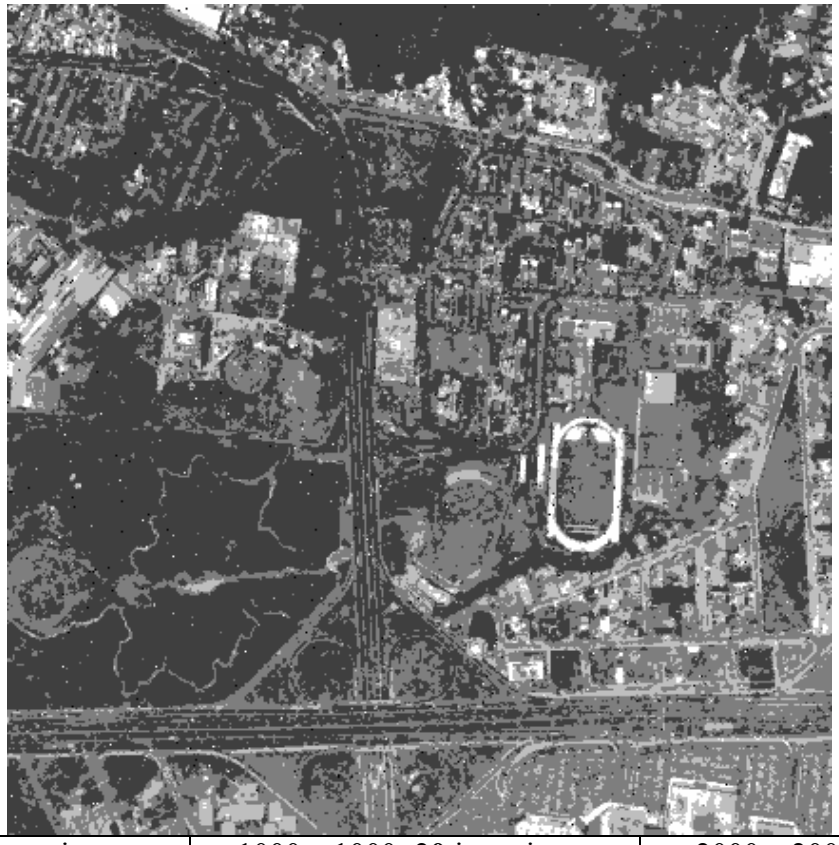
| Imagen original | Tamaño: 2000 x 2000 pixeles |
|---|---|



| 500 x 500: **10** iteraciones | 1000 x 1000: **10** iteraciones | 2000 x 2000: **10** iteraciones |
|---|---|---|

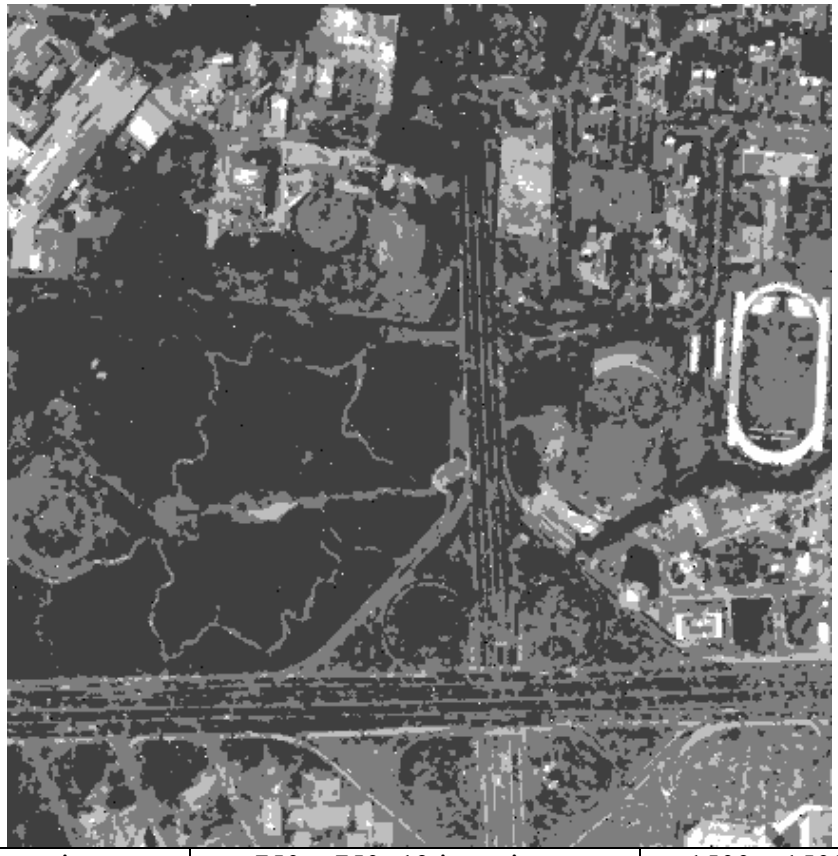


| | | |
|---|---|---|
| 500 x 500: **10** iteraciones | 1000 x 1000: **30** iteraciones | 2000 x 2000: **5** iteraciones |



| | | |
|---|---|---|
| 500 x 500: **20** iteraciones | 1000 x 1000: **20** iteraciones | 2000 x 2000: **5** iteraciones |

| Imagen original | Tamaño: 1500 x 1500 pixeles |
|---|---|



| 375 x 375: **30** iteraciones | 750 x 750: **10** iteraciones | 1500 x 1500: **10** iteraciones |
|---|---|---|

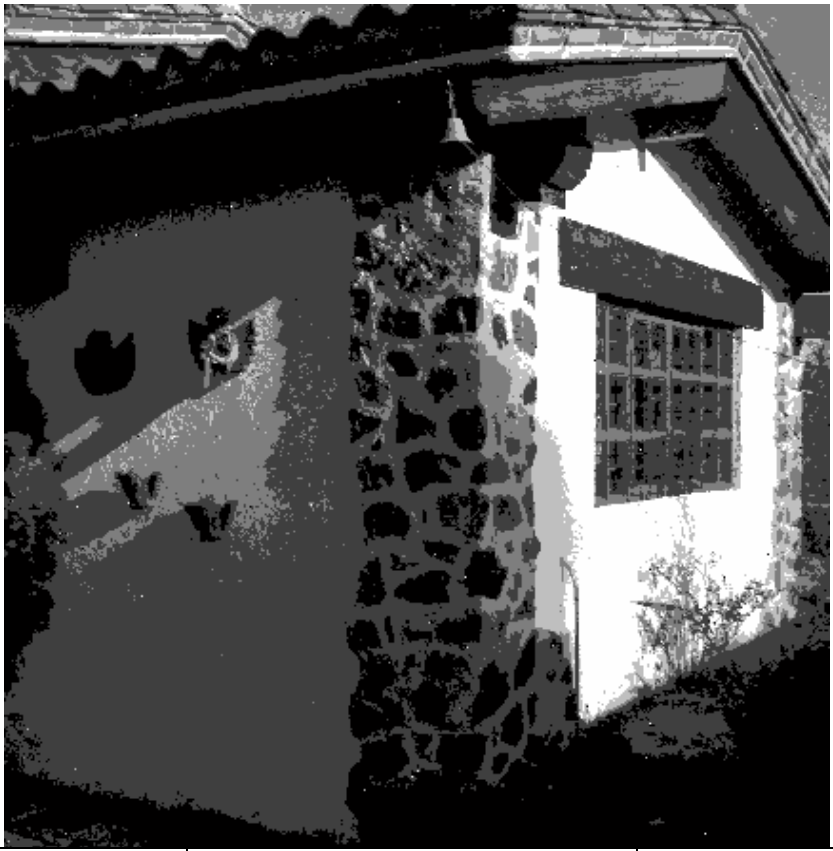
| 375 x 375: **10** iteraciones | 750 x 750: **30** iteraciones | 1500 x 1500: **5** iteraciones |
|---|---|---|

| 375 x 375: **20** iteraciones | 750 x 750: **20** iteraciones | 1500 x 1500: **5** iteraciones |
|---|---|---|

| Imagen original | Tamaño: 1544 x 1544 pixeles |
|---|---|

| 386 x 386: **10** iteraciones | 772 x 772: **10** iteraciones | 1544 x 1544: **10** iteraciones |
|---|---|---|

| 386 x 386: **10** iteraciones | 772 x 772: **10** iteraciones | 1544 x 1544: **20** iteraciones |
|---|---|---|



| 386 x 386: **10** iteraciones | 772 x 772: **10** iteraciones | 1544 x 1544: **40** iteraciones |
|---|---|---|

## 7.2 Comparación de Funcionalidad

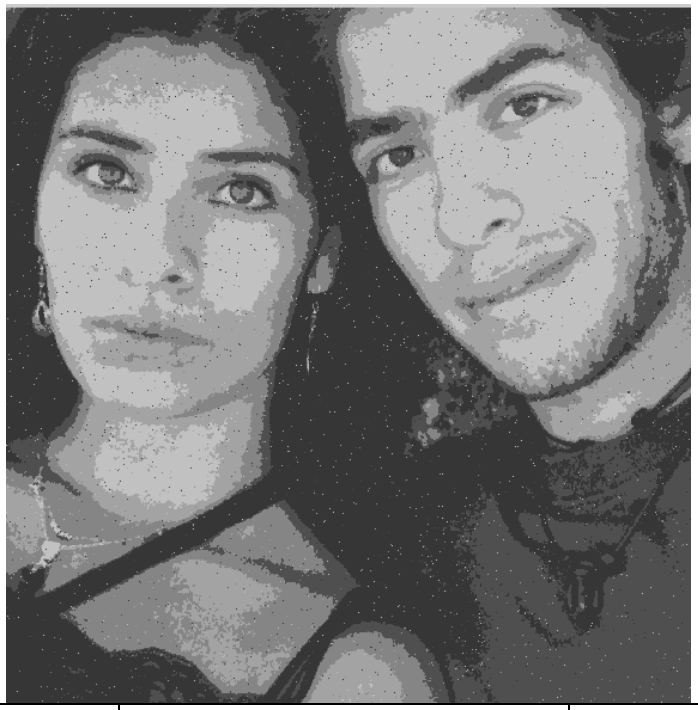
El objetivo de estos dos últimos experimentos es el de comparar el tiempo que emplea cada esquema para realizar el procesamiento. En el primero, la imagen original es sometida a 40 iteraciones. En un caso bajo el esquema multiescala y el segundo sin él, es decir aplicando el algoritmo 40 veces sobre la imagen de dimensiones originales. Los resultados hablan por sí solos. Ambas imágenes presentan el mismo nivel de fidelidad y de entendimiento. No es posible distinguir cuál de las dos fue sometida a qué esquema sin saberlo desde un principio. En pocas palabras, la efectividad de ambos es igual. Sin embargo, su eficiencia no es la misma. En el esquema multiescala, el proceso se llevó a cabo en 1 minuto y 50 segundos. Para el otro caso, bajo las mismas condiciones, el proceso tardó 3 minutos y 4 segundos en otorgar el resultado; un 50% más tardado.

En el segundo experimento se presentan 3 muestras, dos de ellas bajo el sistema multiescala. Entre estas dos se utiliza distinta distribución de iteraciones para analizar también la diferencia en el tiempo de procesamiento bajo distintos sistemas de multiescala. Como era de esperarse, aquel que presenta mayor número de iteraciones para la resolución original tarda más. Con respecto a la muestra que se obtiene bajo 60 iteraciones notamos cierto nivel de ruido en las zonas homogéneas y de esta forma, una vez más se demuestra el efecto paso bajas que elimina parte del ruido de la imagen original. El tiempo de procesamiento es de casi 50% menos para el sistema 10+10+40 y de más del 55% menos para el sistema 15+15+30.

Los siguientes resultados avalan el funcionamiento efectivo y eficiente del esquema multiescala.

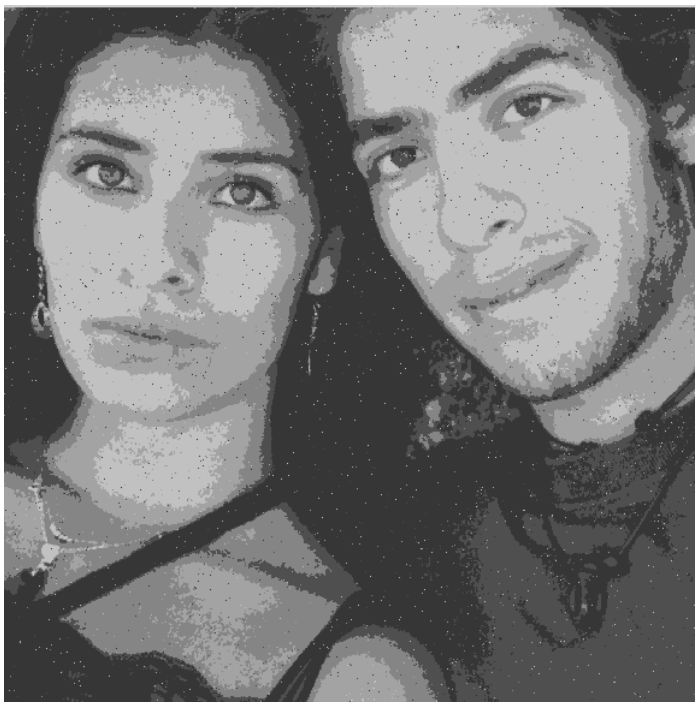
<u>Experimento 1</u>

Tiempo de procesamiento: 1 minuto, 50 segundos



| | | |
|---|---|---|
| 256 x 256: **10** iteraciones | 512 x 512: **10** iteraciones | 1024 x 1024: **20** iteraciones |
| | Total: **40** iteraciones | |

Esquema sin Multiescala
Después de **40** iteraciones
Tiempo de procesamiento: 3 minutos, 4 segundos



Experimento 2

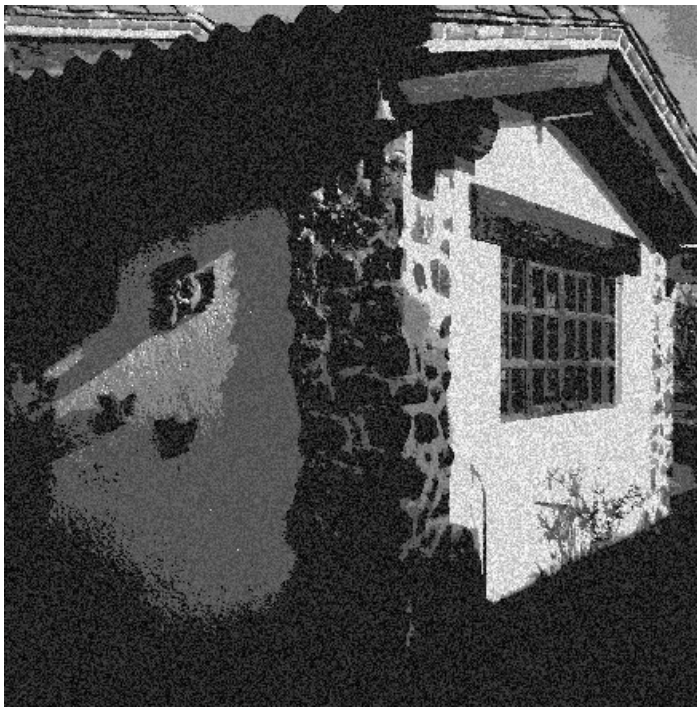
Tiempo de procesamiento: 5 minutos, 22 segundos

| | | |
|---|---|---|
| 386 x 386: **15** iteraciones | 772 x 772: **15** iteraciones | 1544 x 1544: **30** iteraciones |
| | Total: **60** iteraciones | |

Tiempo de procesamiento: 6 minutos, 36 segundos

| | | |
|---|---|---|
| 386 x 386: **10** iteraciones | 772 x 772: **10** iteraciones | 1544 x 1544: **40** iteraciones |
| | Total: **60** iteraciones | |

Esquema sin Multiescala
Después de **60** iteraciones
Tiempo de procesamiento: 9 minutos, 25 segundos



# 8. Conclusiones

Son dos las principales áreas a concluir para la tesis presentada. La primera referida a las ventajas que presenta el algoritmo bajo un esquema multiescala sobre aquellos que trabajan sobre la imagen original a lo largo de todas las iteraciones, y una segunda en la que podemos analizar los distintos comportamientos que se presentan jugando con la distribución de iteraciones entre las escalas de procesamiento.

En los experimentos hechos para comparación de funcionalidad entre los dos esquemas, se puede concluir satisfactoriamente que el esquema multiescala presenta la enorme ventaja de ser un proceso alrededor de 50% más rápido que el esquema tradicional. Las imágenes que el algoritmo otorga no presentan diferencias significativas e incluso nulas entre los dos esquemas. La efectividad es la misma, por lo que no hay efectos contraproducentes en ningún sentido por usar distribución de iteraciones entre tres grados de resolución. Para imágenes pequeñas la diferencia puede ser de unos cuantos minutos, pero cuando se trata de una imagen de alta resolución de percepción remota, la diferencia pude llegar a ser de hasta 20 minutos.

También en lo referente al tiempo de procesamiento, las imágenes de percepción remota nos demuestran que la resolución de la imagen no es el único factor determinante. Para imágenes del mismo tamaño pueden existir tiempos distintos. Ese otro factor es la homogeneidad de niveles de gris que existen en la imagen. Si ésta contiene áreas importantes de niveles de gris muy similares, el procesamiento es más veloz. Por ello, las imágenes de percepción remota, y sobre todo las de tipo urbano, tardan muchas veces más. La distribución de niveles de gris a lo largo de estas es sumamente heterogénea y no hay áreas extensas con niveles de gris similares.

Otra ventaja importante del esquema multiescala es el comportamiento paso bajas que se presenta al aumentar el número de iteraciones en la imagen en baja resolución, eliminando así gran parte del ruido de esta. A partir de esto se distinguen dos importantes características de fidelidad que deben cuidarse al someterse una imagen bajo este algoritmo: homogeneidad y bordes. La homogeneidad como ya lo mencioné se refiere a la capacidad e eliminar el ruido en la imagen obteniendo áreas fieles a determinada clase en toda su extensión. Sin embargo, cuando se abusa de las iteraciones en baja resolución, se descuida el otro factor de fidelidad. Los bordes pierden sus figuras originales si no existe un equilibrio adecuado entre las iteraciones y no se da peso a las de alta resolución. Al aumentar las iteraciones en alta resolución, las fronteras entre las clases se respetan mejor. Incluso se pueden perder clases en los resultados finales fusionándose un par o más de ellos si se somete a demasiado procesamiento en baja resolución.

Una desventaja que tiene este algoritmo es que las clases no se elijen de manera óptima. Hay en sí dos opciones: obtenerlas a partir de la media de niveles de gris dentro de ventanas representativas, o simplemente obtenerlas a base de experimentos y tanteos que permitan trabajar con los niveles más significativos. Una mejora importante a futuro para este algoritmo sería, a mi parecer, incluir un otro que trabaje en conjunto con éste que logre identificar de manera óptima las clases con las que se deba trabajar e incluso el número óptimo de ellas suficiente para tener un total entendimiento de la imagen resultante.

# 9. Referencias

HINES, William
Probability and Statistics in Engineering
Fourth Edition
New Jersey
John Wiley & Sons, 2003

LEON-GARCÍA, Alberto
Probability and Random Processes for electrical engineering
2nd edition
Massachusetts
Addison-Wesley Publishing Company, 1994

PAPOULIS, A.
Probability, Random Variables and Stochastic Processes
4th edition
Boston
McGraw-Hill, 1991

PEEBLES, P.Z.
Probability, Random Variables and Random Signal Principles
3rd edition
New York
McGraw-Hill, 1993

PROAKIS, J.G.
Digital Communications
4th edition
Boston
McGraw-Hill, 2001

RONG, Li X.
Probability, Random Signals, and Statistics
Florida
CRC Press, 1999

ZOLTAN KATO
Modelisations markoviennes multiresolutions en vision par ordinateur. Application a la segmentation d'images SPOT
PhD thesis
Nice
University of Nice Sophia Antipolis, 1994

# I. Apéndice

El siguiente es el código del algoritmo programado para la presente tesis. Bajo éste fueron sometidos todos los experimentos realizados en ella. Cabe señalar que cambios menores fueron necesarios para cada experimento, pero básicamente se empleó el mismo para todos ellos.

```
clc;
clear all;
I=imread('DFtruncado5.jpg','jpg');
figure
colormap(gray(256));
image(I);
axis('off');

I=double(I);
I=I+1;

% %figure(1)
% %ini=round(ginput(1));
%
% %primera clase
% clase1=I(342:410,776:840);
%
% %segunda clase
% clase2=I(198:289,95:188);
%
% %tercera clase
% clase3=I(347:424,927:1000);
%
% %cuarta clase
% clase4=I(562:629,623:691);
%
% %quinta clase
% clase5=I(400:483,500:583);
%
%
% figure(2)
% colormap(gray(256));
% subplot(2,3,1)
% image(clase1)
% title('Clase Wo');
% axis('off');
% axis('square');
% colormap(gray(256));
% subplot(2,3,2)
% image(clase2)
% title('Clase W1');
% axis('off');
% axis('square');
% colormap(gray(256));
% subplot(2,3,3)
% image(clase3)
% title('Clase W2');
% axis('off');
% axis('square');
% colormap(gray(256));
% subplot(2,3,4)
% image(clase4)
% title('Clase W3');
% axis('off');
% axis('square');
% colormap(gray(256));
% subplot(2,3,6)
% image(clase5)
% title('Clase W4');
% axis('off');
% axis('square');

%medias

% Med1(1)=round(mean(mean(clase1)));
% Med1(2)=round(mean(mean(clase2)));
% Med1(3)=round(mean(mean(clase3)));
% Med1(4)=round(mean(mean(clase4)));
% Med1(5)=round(mean(mean(clase5)));
% Med1=sort(Med1);


Med1(1)=1;
Med1(2)=64;
Med1(3)=128;
Med1(4)=191;
Med1(5)=255;
Med1=sort(Med1);


figure;
for i=1:4
          S(i)=Med1(i+1)-Med1(i);
    end

X=0:255;

%Términos de probabilidad condicional
Clase W0 esta la obtenemos de aplicar el
entrenamiento
subplot(5,1,1);

    for i=1:256
        if i<(Med1(1)+round(S(1)/4))
      W0(i)=0;
            elseif
i>=((Med1(1)+round(S(1)/4)))&i<((Med1(1)+r
ound(3*S(1)/4)))
      W0(i)=2*i/S(1)-(2*Med1(1)/S(1)+6/4-
1);
        else
       W0(i)=1;
        end
        end% se clasifica la probabilidad
de acuerdo al criterio de Bayes

            plot(X,W0,'Color','red',...
    'LineWidth',2);

        title('Probabilidad condicional de
la Imagen Original');
    ylabel('U(W0/x)','FontSize',8);
```

```
    axis([0 255 0 1.5]);

    %Términos de la probabilidad
condicional Clase W1
subplot(5,1,2);
            for i=1:256
             if i<(Med1(2)-
round(3*S(1)/4))|i>=(Med1(2)+round(3*S(2)/
4))
         W1(i)=1;
             elseif i>=((Med1(2)-
round(3*S(1)/4)))&i<=((Med1(2)-
round(S(1)/4)))
             W1(i)=-
2*i/S(1)+(2*Med1(1)/S(1)+6/4);
            elseif
i>(Med1(2)+round(S(2)/4))&i<(Med1(2)+round
(3*S(2)/4))
            W1(i)=2*i/S(2)-
(2*Med1(2)/S(2)+6/4-1);
            else
            W1(i)=0;
             end
end
plot(X,W1,'Color','red',...
    'LineWidth',2);
ylabel('U(W1/X)','FontSize',8);
axis([0 255 0 1.5]);

%Términos de la probabilidad condicional
Clase W2  Se repite para todas la ventanas
y su probabilidades apriori
        subplot(5,1,3);
        for i=1:256
   if i<=(Med1(3)-
round(3*S(2)/4))|i>(Med1(3)+round(3*S(3)/4
))
      W2(i)=1;
   elseif i>((Med1(3)-
round(3*S(2)/4)))&i<((Med1(3)-
round(S(2)/4)))
      W2(i)=-2*i/S(2)+2*Med1(2)/S(2)+6/4;
   elseif
i>((Med1(3)+round(S(3)/4)))&i<=((Med1(3)+r
ound(3*S(3)/4)))
      W2(i)=2*i/S(3)-(2*Med1(3)/S(3)+6/4-
1);
   else
      W2(i)=0;
   end
end

plot(X,W2,'Color','red',...
    'LineWidth',2);
ylabel('U(W2/X)','FontSize',8);
axis([0 255 0 1.5]);

%Términos de la probabilidad condicional
Clase W3
subplot(5,1,4);
for i=1:256
   if i<(Med1(4)-
round(3*S(3)/4))|i>=(Med1(4)+round(3*S(4)/
4))
      W3(i)=1;
   elseif i>=((Med1(4)-
round(3*S(3)/4)))&i<=((Med1(4)-
round(S(3)/4)))
      W3(i)=-2*i/S(3)+2*Med1(3)/S(3)+6/4;
   elseif
i>((Med1(4)+round(S(4)/4)))&i<((Med1(4)+ro
und(3*S(4)/4)))
      W3(i)=2*i/S(4)-(2*Med1(4)/S(4)+6/4-
1);
   else
      W3(i)=0;
   end
end

plot(X,W3,'Color','red',...
    'LineWidth',2);
ylabel('U(W3/X)','FontSize',8);
axis([0 255 0 1.5]);

            %Términos de la probabilidad
condicional Clase W4
        subplot(5,1,5);
        for i=1:256
         if i<=(Med1(5)-round(3*S(4)/4))
       W4(i)=1;
          elseif i>((Med1(5)-
round(3*S(4)/4)))&i<((Med1(5)-
round(S(4)/4)))
      W4(i)=-2*i/S(4)+2*Med1(4)/S(4)+6/4;
     else
          W4(i)=0;
    end
        end

    plot(X,W4,'Color','red',...
    'LineWidth',2);
    ylabel('U(W4/X)','FontSize',8);
    axis([0 255 0 1.5]);



A = size(I);

z=A(1,1);
x=A(1,2);



% MULTIESCALA%


D= rand(z/2,x/2);


for n = 1:z/2
    for m = 1:x/2
        i=2*n-1;
        j=2*m-1;

D(n,m)=(I(i,j)+I(i,j+1)+I(i+1,j)+I(i+1,j+1
))/4;
    end
end


E= rand(z/4,x/4);
```

```
for n = 1:z/4
    for m = 1:x/4
        i=2*n-1;
        j=2*m-1;

E(n,m)=(D(i,j)+D(i,j+1)+D(i+1,j)+D(i+1,j+1
))/4;
    end
end

D=double(D);
%D=D+1;
D=round(D);

E=double(E);
%E=E+1;
E=round(E);


%Determinacion aleatoria de la solucion Y1

P=size(E);
f=P(1,1);
g=P(1,2);

Y1=rand(size(E));

for i=1:f
    for j=1:g
        if Y1(i,j)<=0.2
        Y1(i,j)=1;
        elseif Y1(i,j)>0.2 & Y1(i,j)<=0.4
        Y1(i,j)=2;
        elseif Y1(i,j)>0.4 & Y1(i,j)<=0.6
        Y1(i,j)=3;
        elseif Y1(i,j)>0.6 & Y1(i,j)<=0.8
        Y1(i,j)=4;
        elseif Y1(i,j)>0.8
        Y1(i,j)=5;
        end
    end
end




%  ITERACIONES  256 x 256


T=2.5;
for  n= 1:10


% % contador de pixel actual

P=rand(size(E));

for  i=1:f
    for  j=1:g
pix=0;
pix1=0;
pix2=0;
pix3=0;
pix4=0;
pix5=0;
pix6=0;
pix7=0;
pix8=0;
%           %esquina superior izquierda
          if i==1 & j==1
            if Y1(i+1,j)==Y1(i,j)
               pix=pix+1;
            end
            if Y1(i+1,j+1)==Y1(i,j)
                pix=pix+1;
            end
            if Y1(i,j+1)==Y1(i,j)
                pix=pix+1;
            end
            P(i,j)=pix;

%             %parte superior
        elseif j==1 & i~=1 & i~=f
            if Y1(i+1,j)==Y1(i,j)
                pix1=pix1+1;
            end
            if Y1(i+1,j+1)==Y1(i,j)
                pix1=pix1+1;
            end
            if Y1(i,j+1)==Y1(i,j)
               pix1=pix1+1;
            end
             if Y1(i-1,j+1)==Y1(i,j)
                pix1=pix1+1;
            end
             if Y1(i-1,j)==Y1(i,j)
                pix1=pix1+1;
            end
            P(i,j)=pix1;

%             %centro
        elseif j~=1 & j~=g & i~=1 & i~=f
            if Y1(i+1,j)==Y1(i,j)
                pix2=pix2+1;
            end
            if Y1(i+1,j+1)==Y1(i,j)
                pix2=pix2+1;
            end
            if Y1(i,j+1)==Y1(i,j)
                pix2=pix2+1;
            end
             if Y1(i-1,j+1)==Y1(i,j)
                pix2=pix2+1;
            end
             if Y1(i-1,j)==Y1(i,j)
                pix2=pix2+1;
            end
            if Y1(i-1,j-1)==Y1(i,j)
                pix2=pix2+1;
            end
            if Y1(i,j-1)==Y1(i,j)
                pix2=pix2+1;
            end
            if Y1(i+1,j-1)==Y1(i,j)
                pix2=pix2+1;
            end
            P(i,j)=pix2;

%             %esquina superior derecha
        elseif i==f & j==1
            if Y1(i,j+1)==Y1(i,j)
```

```
                pix3=pix3+1;
            end
            if Y1(i-1,j+1)==Y1(i,j)
                pix3=pix3+1;
            end
            if Y1(i-1,j)==Y1(i,j)
                pix3=pix3+1;
            end
            P(i,j)=pix3;

%             %costado lateral derecho
        elseif i==f & j~=1 & j~=g
            if Y1(i-1,j-1)==Y1(i,j)
                pix4=pix4+1;
            end
            if Y1(i-1,j+1)==Y1(i,j)
                pix4=pix4+1;
            end
            if Y1(i-1,j)==Y1(i,j)
                pix4=pix4+1;
            end
            P(i,j)=pix4;

            %esquina inferior derecha
        elseif i==f & j==g
            if Y1(i-1,j)==Y1(i,j)
               pix5=pix5+1;
            end
            if Y1(i-1,j-1)==Y1(i,j)
               pix5=pix5+1;
            end
            if Y1(i,j-1)==Y1(i,j)
               pix5=pix5+1;
            end
            P(i,j)=pix5;

%             %parte inferior
        elseif j==g & i~=1 & i~=f
            if Y1(i-1,j)==Y1(i,j)
               pix6=pix6+1;
            end
            if Y1(i-1,j-1)==Y1(i,j)
                pix6=pix6+1;
            end
            if Y1(i,j-1)==Y1(i,j)
                pix6=pix6+1;
            end
            if Y1(i+1,j-1)==Y1(i,j)
                pix6=pix6+1;
            end
            if Y1(i+1,j)==Y1(i,j)
                pix6=pix6+1;
            end
         P(i,j)=pix;

%             %esquina inferior izquierda
        elseif i==1 & j==g
            if Y1(i,j-1)==Y1(i,j)
                pix7=pix7+1;
            end
            if Y1(i+1,j-1)==Y1(i,j)
                pix7=pix7+1;
            end
            if Y1(i+1,j)==Y1(i,j)
                pix7=pix7+1;
            end
            P(i,j)=pix7;

%             %costado lateral izquierdo
        elseif i==1 & j~=1 & j~=g
            if Y1(i,j-1)==Y1(i,j)
                pix8=pix8+1;
            end
            if Y1(i+1,j-1)==Y1(i,j)
                pix8=pix8+1;
            end
            if Y1(i+1,j)==Y1(i,j)
                pix8=pix8+1;
            end
            if Y1(i+1,j+1)==Y1(i,j)
                pix8=pix8+1;
            end
            if Y1(i,j+1)==Y1(i,j)
                pix8=pix8+1;
            end
            P(i,j)=pix8;
        end
        end

    end



% contador de pixel eleccion

R=rand(size(E));


for i=1:f
    for j=1:g
        if R(i,j)<=0.2
        R(i,j)=1;
        elseif R(i,j)>0.2 & R(i,j)<=0.4
        R(i,j)=2;
        elseif R(i,j)>0.4 & R(i,j)<=0.6
        R(i,j)=3;
        elseif R(i,j)>0.6 & R(i,j)<=0.8
        R(i,j)=4;
        elseif R(i,j)>0.8
        R(i,j)=5;
        end
    end
end


P1=rand(size(E));

for  i=1:f
    for  j=1:g
pix=0;
pix1=0;
pix2=0;
pix3=0;
pix4=0;
pix5=0;
pix6=0;
pix7=0;
pix8=0;
%           %esquina superior izquierda
          if i==1 & j==1
            if R(i+1,j)==R(i,j)
```

```
               pix=pix+1;
           end
           if R(i+1,j+1)==R(i,j)
               pix=pix+1;
           end
           if R(i,j+1)==R(i,j)
               pix=pix+1;
           end
           P1(i,j)=pix;

%            %parte superior
       elseif j==1 & i~=1 & i~=f
           if R(i+1,j)==R(i,j)
               pix1=pix1+1;
           end
           if R(i+1,j+1)==R(i,j)
               pix1=pix1+1;
           end
           if R(i,j+1)==R(i,j)
              pix1=pix1+1;
           end
            if R(i-1,j+1)==R(i,j)
               pix1=pix1+1;
           end
            if R(i-1,j)==R(i,j)
               pix1=pix1+1;
           end
           P1(i,j)=pix1;

%            %centro
       elseif j~=1 & j~=g & i~=1 & i~=f
           if R(i+1,j)==R(i,j)
               pix2=pix2+1;
           end
           if R(i+1,j+1)==R(i,j)
               pix2=pix2+1;
           end
           if R(i,j+1)==R(i,j)
               pix2=pix2+1;
           end
            if R(i-1,j+1)==R(i,j)
               pix2=pix2+1;
           end
            if R(i-1,j)==R(i,j)
               pix2=pix2+1;
           end
           if R(i-1,j-1)==R(i,j)
               pix2=pix2+1;
           end
           if R(i,j-1)==R(i,j)
               pix2=pix2+1;
           end
           if R(i+1,j-1)==R(i,j)
               pix2=pix2+1;
           end
           P1(i,j)=pix2;

%            %esquina superior derecha
       elseif i==f & j==1
           if R(i,j+1)==R(i,j)
               pix3=pix3+1;
           end
           if R(i-1,j+1)==R(i,j)
               pix3=pix3+1;
           end
           if R(i-1,j)==R(i,j)
               pix3=pix3+1;
           end
           P1(i,j)=pix3;

%            %costado lateral derecho
       elseif i==f & j~=1 & j~=g
           if R(i-1,j-1)==R(i,j)
               pix4=pix4+1;
           end
           if R(i-1,j+1)==R(i,j)
               pix4=pix4+1;
           end
           if R(i-1,j)==R(i,j)
               pix4=pix4+1;
           end
           P1(i,j)=pix4;

           %esquina inferior derecha
       elseif i==f & j==g
           if R(i-1,j)==R(i,j)
              pix5=pix5+1;
           end
           if R(i-1,j-1)==R(i,j)
              pix5=pix5+1;
           end
           if R(i,j-1)==R(i,j)
              pix5=pix5+1;
           end
           P1(i,j)=pix5;

%            %parte inferior
       elseif j==g & i~=1 & i~=f
           if R(i-1,j)==R(i,j)
              pix6=pix6+1;
           end
           if R(i-1,j-1)==R(i,j)
               pix6=pix6+1;
           end
           if R(i,j-1)==R(i,j)
               pix6=pix6+1;
           end
           if R(i+1,j-1)==R(i,j)
               pix6=pix6+1;
           end
           if R(i+1,j)==R(i,j)
               pix6=pix6+1;
           end
        P1(i,j)=pix;

%            %esquina inferior izquierda
       elseif i==1 & j==g
           if R(i,j-1)==R(i,j)
               pix7=pix7+1;
           end
           if R(i+1,j-1)==R(i,j)
               pix7=pix7+1;
           end
           if R(i+1,j)==R(i,j)
               pix7=pix7+1;
           end
           P1(i,j)=pix7;

%            %costado lateral izquierdo
       elseif i==1 & j~=1 & j~=g
           if R(i,j-1)==R(i,j)
               pix8=pix8+1;
```

```
            end
            if R(i+1,j-1)==R(i,j)
                pix8=pix8+1;
            end
            if R(i+1,j)==R(i,j)
                pix8=pix8+1;
            end
            if R(i+1,j+1)==R(i,j)
                pix8=pix8+1;
            end
            if R(i,j+1)==R(i,j)
                pix8=pix8+1;
            end
            P1(i,j)=pix8;
        end
        end

   end



% Modelo de Potts
    %Definicion del valor de beta
        B=0.35;
     %Energias
        P=P*-B;
        P1=P1*-B;


% comparacion entre pixel actual y pixel
eleccion
    %se les suma su proyeccion sobre las
clases



 for  i=1:f
    for  j=1:g

    if Y1(i,j)==1
            P(i,j)=P(i,j)+W0(E(i,j));
        elseif Y1(i,j)==2
            P(i,j)=P(i,j)+W1(E(i,j));
        elseif Y1(i,j)==3
            P(i,j)=P(i,j)+W2(E(i,j));
        elseif Y1(i,j)==4
            P(i,j)=P(i,j)+W3(E(i,j));
        elseif Y1(i,j)==5
            P(i,j)=P(i,j)+W4(E(i,j));
        end

        if R(i,j)==1
            P1(i,j)=P1(i,j)+W0(E(i,j));
        elseif R(i,j)==2
            P1(i,j)=P1(i,j)+W1(E(i,j));
        elseif R(i,j)==3
            P1(i,j)=P1(i,j)+W2(E(i,j));
        elseif R(i,j)==4
            P1(i,j)=P1(i,j)+W3(E(i,j));
        elseif R(i,j)==5
            P1(i,j)=P1(i,j)+W4(E(i,j));
        end

        if P1(i,j)<=P(i,j)
            Y1(i,j)=R(i,j);
        else
            p=exp(-((P1(i,j)-
P(i,j))/T^n));
            ale=rand;
            if ale<p
               Y1(i,j)=R(i,j);
            end
        end
    end
end

T=T*.95

end   %fin de primera iteracion



%  PROYECCION c/pixel 4 veces

Y2=rand(size(Y1)*2);

P=size(D);
r=P(1,1);
s=P(1,2);

for n=1:g
    for m=1:f
        i=2*n-1;
        j=2*m-1;
        Y2(i,j)=Y1(n,m);
        Y2(i+1,j+1)=Y1(n,m);
        Y2(i+1,j)=Y1(n,m);
        Y2(i,j+1)=Y1(n,m);
    end
end


%  ITERACIONES  512 x 512


for  n= 1:10


% % contador de pixel actual

P=rand(size(D));

for  i=1:r
    for  j=1:s
pix=0;
pix1=0;
pix2=0;
pix3=0;
pix4=0;
pix5=0;
pix6=0;
pix7=0;
pix8=0;
%           %esquina superior izquierda
          if i==1 & j==1
            if Y2(i+1,j)==Y2(i,j)
               pix=pix+1;
            end
            if Y2(i+1,j+1)==Y2(i,j)
                pix=pix+1;
            end
            if Y2(i,j+1)==Y2(i,j)
```

```
                pix=pix+1;
            end
            P(i,j)=pix;

%             %parte superior
        elseif j==1 & i~=1 & i~=r
            if Y2(i+1,j)==Y2(i,j)
                pix1=pix1+1;
            end
            if Y2(i+1,j+1)==Y2(i,j)
                pix1=pix1+1;
            end
            if Y2(i,j+1)==Y2(i,j)
               pix1=pix1+1;
            end
             if Y2(i-1,j+1)==Y2(i,j)
                pix1=pix1+1;
            end
             if Y2(i-1,j)==Y2(i,j)
                pix1=pix1+1;
            end
            P(i,j)=pix1;

%             %centro
        elseif j~=1 & j~=s & i~=1 & i~=r
            if Y2(i+1,j)==Y2(i,j)
                pix2=pix2+1;
            end
            if Y2(i+1,j+1)==Y2(i,j)
                pix2=pix2+1;
            end
            if Y2(i,j+1)==Y2(i,j)
                pix2=pix2+1;
            end
             if Y2(i-1,j+1)==Y2(i,j)
                pix2=pix2+1;
            end
             if Y2(i-1,j)==Y2(i,j)
                pix2=pix2+1;
            end
            if Y2(i-1,j-1)==Y2(i,j)
                pix2=pix2+1;
            end
            if Y2(i,j-1)==Y2(i,j)
                pix2=pix2+1;
            end
            if Y2(i+1,j-1)==Y2(i,j)
                pix2=pix2+1;
            end
            P(i,j)=pix2;

%             %esquina superior derecha
        elseif i==r & j==1
            if Y2(i,j+1)==Y2(i,j)
                pix3=pix3+1;
            end
            if Y2(i-1,j+1)==Y2(i,j)
                pix3=pix3+1;
            end
            if Y2(i-1,j)==Y2(i,j)
                pix3=pix3+1;
            end
            P(i,j)=pix3;

%             %costado lateral derecho
        elseif i==r & j~=1 & j~=s
            if Y2(i-1,j-1)==Y2(i,j)
                pix4=pix4+1;
            end
            if Y2(i-1,j+1)==Y2(i,j)
                pix4=pix4+1;
            end
            if Y2(i-1,j)==Y2(i,j)
                pix4=pix4+1;
            end
            P(i,j)=pix4;

            %esquina inferior derecha
        elseif i==r & j==s
            if Y2(i-1,j)==Y2(i,j)
               pix5=pix5+1;
            end
            if Y2(i-1,j-1)==Y2(i,j)
               pix5=pix5+1;
            end
            if Y2(i,j-1)==Y2(i,j)
               pix5=pix5+1;
            end
            P(i,j)=pix5;

%             %parte inferior
        elseif j==s & i~=1 & i~=r
            if Y2(i-1,j)==Y2(i,j)
               pix6=pix6+1;
            end
            if Y2(i-1,j-1)==Y2(i,j)
                pix6=pix6+1;
            end
            if Y2(i,j-1)==Y2(i,j)
                pix6=pix6+1;
            end
            if Y2(i+1,j-1)==Y2(i,j)
                pix6=pix6+1;
            end
            if Y2(i+1,j)==Y2(i,j)
                pix6=pix6+1;
            end
         P(i,j)=pix;

%             %esquina inferior izquierda
        elseif i==1 & j==s
            if Y2(i,j-1)==Y2(i,j)
                pix7=pix7+1;
            end
            if Y2(i+1,j-1)==Y2(i,j)
                pix7=pix7+1;
            end
            if Y2(i+1,j)==Y2(i,j)
                pix7=pix7+1;
            end
            P(i,j)=pix7;

%             %costado lateral izquierdo
        elseif i==1 & j~=1 & j~=s
            if Y2(i,j-1)==Y2(i,j)
                pix8=pix8+1;
            end
            if Y2(i+1,j-1)==Y2(i,j)
                pix8=pix8+1;
            end
            if Y2(i+1,j)==Y2(i,j)
                pix8=pix8+1;
```

```
            end
            if Y2(i+1,j+1)==Y2(i,j)
                pix8=pix8+1;
            end
            if Y2(i,j+1)==Y2(i,j)
                pix8=pix8+1;
            end
            P(i,j)=pix8;
        end
        end

   end



% contador de pixel eleccion

R=rand(size(D));


for i=1:r
    for j=1:s
        if R(i,j)<=0.2
        R(i,j)=1;
        elseif R(i,j)>0.2 & R(i,j)<=0.4
        R(i,j)=2;
        elseif R(i,j)>0.4 & R(i,j)<=0.6
        R(i,j)=3;
        elseif R(i,j)>0.6 & R(i,j)<=0.8
        R(i,j)=4;
        elseif R(i,j)>0.8
        R(i,j)=5;
        end
    end
end


P1=rand(size(D));

for  i=1:r
    for  j=1:s
pix=0;
pix1=0;
pix2=0;
pix3=0;
pix4=0;
pix5=0;
pix6=0;
pix7=0;
pix8=0;
%           %esquina superior izquierda
          if i==1 & j==1
            if R(i+1,j)==R(i,j)
               pix=pix+1;
            end
            if R(i+1,j+1)==R(i,j)
                pix=pix+1;
            end
            if R(i,j+1)==R(i,j)
                pix=pix+1;
            end
            P1(i,j)=pix;

%             %parte superior
        elseif j==1 & i~=1 & i~=r
            if R(i+1,j)==R(i,j)
                pix1=pix1+1;
            end
            if R(i+1,j+1)==R(i,j)
                pix1=pix1+1;
            end
            if R(i,j+1)==R(i,j)
               pix1=pix1+1;
            end
             if R(i-1,j+1)==R(i,j)
                pix1=pix1+1;
            end
             if R(i-1,j)==R(i,j)
                pix1=pix1+1;
            end
            P1(i,j)=pix1;

%             %centro
        elseif j~=1 & j~=s & i~=1 & i~=r
            if R(i+1,j)==R(i,j)
                pix2=pix2+1;
            end
            if R(i+1,j+1)==R(i,j)
                pix2=pix2+1;
            end
            if R(i,j+1)==R(i,j)
                pix2=pix2+1;
            end
             if R(i-1,j+1)==R(i,j)
                pix2=pix2+1;
            end
             if R(i-1,j)==R(i,j)
                pix2=pix2+1;
            end
            if R(i-1,j-1)==R(i,j)
                pix2=pix2+1;
            end
            if R(i,j-1)==R(i,j)
                pix2=pix2+1;
            end
            if R(i+1,j-1)==R(i,j)
                pix2=pix2+1;
            end
            P1(i,j)=pix2;

%             %esquina superior derecha
        elseif i==r & j==1
            if R(i,j+1)==R(i,j)
                pix3=pix3+1;
            end
            if R(i-1,j+1)==R(i,j)
                pix3=pix3+1;
            end
            if R(i-1,j)==R(i,j)
                pix3=pix3+1;
            end
            P1(i,j)=pix3;

%             %costado lateral derecho
        elseif i==r & j~=1 & j~=s
            if R(i-1,j-1)==R(i,j)
                pix4=pix4+1;
            end
            if R(i-1,j+1)==R(i,j)
                pix4=pix4+1;
            end
```

```
            if R(i-1,j)==R(i,j)
                pix4=pix4+1;
            end
            P1(i,j)=pix4;

            %esquina inferior derecha
        elseif i==r & j==s
            if R(i-1,j)==R(i,j)
               pix5=pix5+1;
            end
            if R(i-1,j-1)==R(i,j)
               pix5=pix5+1;
            end
            if R(i,j-1)==R(i,j)
               pix5=pix5+1;
            end
            P1(i,j)=pix5;

%             %parte inferior
        elseif j==s & i~=1 & i~=r
            if R(i-1,j)==R(i,j)
               pix6=pix6+1;
            end
            if R(i-1,j-1)==R(i,j)
                pix6=pix6+1;
            end
            if R(i,j-1)==R(i,j)
                pix6=pix6+1;
            end
            if R(i+1,j-1)==R(i,j)
                pix6=pix6+1;
            end
            if R(i+1,j)==R(i,j)
                pix6=pix6+1;
            end
         P1(i,j)=pix;

%             %esquina inferior izquierda
        elseif i==1 & j==s
            if R(i,j-1)==R(i,j)
                pix7=pix7+1;
            end
            if R(i+1,j-1)==R(i,j)
                pix7=pix7+1;
            end
            if R(i+1,j)==R(i,j)
                pix7=pix7+1;
            end
            P1(i,j)=pix7;

%             %costado lateral izquierdo
        elseif i==1 & j~=1 & j~=s
            if R(i,j-1)==R(i,j)
                pix8=pix8+1;
            end
            if R(i+1,j-1)==R(i,j)
                pix8=pix8+1;
            end
            if R(i+1,j)==R(i,j)
                pix8=pix8+1;
            end
            if R(i+1,j+1)==R(i,j)
                pix8=pix8+1;
            end
            if R(i,j+1)==R(i,j)
                pix8=pix8+1;
            end
            P1(i,j)=pix8;
        end
        end

   end



% Modelo de Potts
    %Definicion del valor de beta
        B=0.35;
     %Energias
        P=P*-B;
        P1=P1*-B;


% comparacion entre pixel actual y pixel
eleccion
    %se les suma su proyeccion sobre las
clases



 for  i=1:r
    for  j=1:s

    if Y2(i,j)==1
            P(i,j)=P(i,j)+W0(D(i,j));
        elseif Y2(i,j)==2
            P(i,j)=P(i,j)+W1(D(i,j));
        elseif Y2(i,j)==3
            P(i,j)=P(i,j)+W2(D(i,j));
        elseif Y2(i,j)==4
            P(i,j)=P(i,j)+W3(D(i,j));
        elseif Y2(i,j)==5
            P(i,j)=P(i,j)+W4(D(i,j));
        end

        if R(i,j)==1
            P1(i,j)=P1(i,j)+W0(D(i,j));
        elseif R(i,j)==2
            P1(i,j)=P1(i,j)+W1(D(i,j));
        elseif R(i,j)==3
            P1(i,j)=P1(i,j)+W2(D(i,j));
        elseif R(i,j)==4
            P1(i,j)=P1(i,j)+W3(D(i,j));
        elseif R(i,j)==5
            P1(i,j)=P1(i,j)+W4(D(i,j));
        end

        if P1(i,j)<=P(i,j)
            Y2(i,j)=R(i,j);
        else
            p=exp(-((P1(i,j)-
P(i,j))/T^n));
            ale=rand;
            if ale<p
               Y2(i,j)=R(i,j);
            end
       end
    end
end

T=T*.95
```

```
end   %fin de segunda iteracion



%  PROYECCION c/pixel 4 veces

Y3=rand(size(Y2)*2);

P=size(I);
z=P(1,1);
x=P(1,2);

for n=1:r
    for m=1:s
        i=2*n-1;
        j=2*m-1;
        Y3(i,j)=Y2(n,m);
        Y3(i+1,j+1)=Y2(n,m);
        Y3(i+1,j)=Y2(n,m);
        Y3(i,j+1)=Y2(n,m);
    end
end



%  ITERACIONES  1024 x 1024


for  n= 1:30


% % contador de pixel actual

P=rand(size(I));

for  i=1:z
    for  j=1:x
pix=0;
pix1=0;
pix2=0;
pix3=0;
pix4=0;
pix5=0;
pix6=0;
pix7=0;
pix8=0;
%           %esquina superior izquierda
          if i==1 & j==1
            if Y3(i+1,j)==Y3(i,j)
               pix=pix+1;
            end
            if Y3(i+1,j+1)==Y3(i,j)
                pix=pix+1;
            end
            if Y3(i,j+1)==Y3(i,j)
                pix=pix+1;
            end
            P(i,j)=pix;

%             %parte superior
        elseif j==1 & i~=1 & i~=z
            if Y3(i+1,j)==Y3(i,j)
                pix1=pix1+1;
            end
            if Y3(i+1,j+1)==Y3(i,j)
                pix1=pix1+1;
            end
            if Y3(i,j+1)==Y3(i,j)
               pix1=pix1+1;
            end
             if Y3(i-1,j+1)==Y3(i,j)
                pix1=pix1+1;
            end
             if Y3(i-1,j)==Y3(i,j)
                pix1=pix1+1;
            end
            P(i,j)=pix1;

%             %centro
        elseif j~=1 & j~=x & i~=1 & i~=z
            if Y3(i+1,j)==Y3(i,j)
                pix2=pix2+1;
            end
            if Y3(i+1,j+1)==Y3(i,j)
                pix2=pix2+1;
            end
            if Y3(i,j+1)==Y3(i,j)
                pix2=pix2+1;
            end
             if Y3(i-1,j+1)==Y3(i,j)
                pix2=pix2+1;
            end
             if Y3(i-1,j)==Y3(i,j)
                pix2=pix2+1;
            end
            if Y3(i-1,j-1)==Y3(i,j)
                pix2=pix2+1;
            end
            if Y3(i,j-1)==Y3(i,j)
                pix2=pix2+1;
            end
            if Y3(i+1,j-1)==Y3(i,j)
                pix2=pix2+1;
            end
            P(i,j)=pix2;

%             %esquina superior derecha
        elseif i==z & j==1
            if Y3(i,j+1)==Y3(i,j)
                pix3=pix3+1;
            end
            if Y3(i-1,j+1)==Y3(i,j)
                pix3=pix3+1;
            end
            if Y3(i-1,j)==Y3(i,j)
                pix3=pix3+1;
            end
            P(i,j)=pix3;

%             %costado lateral derecho
        elseif i==z & j~=1 & j~=x
            if Y3(i-1,j-1)==Y3(i,j)
                pix4=pix4+1;
            end
            if Y3(i-1,j+1)==Y3(i,j)
                pix4=pix4+1;
            end
            if Y3(i-1,j)==Y3(i,j)
                pix4=pix4+1;
            end
            P(i,j)=pix4;
```

```
            %esquina inferior derecha
        elseif i==z & j==x
            if Y3(i-1,j)==Y3(i,j)
               pix5=pix5+1;
            end
            if Y3(i-1,j-1)==Y3(i,j)
               pix5=pix5+1;
            end
            if Y3(i,j-1)==Y3(i,j)
               pix5=pix5+1;
            end
            P(i,j)=pix5;

%             %parte inferior
        elseif j==x & i~=1 & i~=z
            if Y3(i-1,j)==Y3(i,j)
               pix6=pix6+1;
            end
            if Y3(i-1,j-1)==Y3(i,j)
                pix6=pix6+1;
            end
            if Y3(i,j-1)==Y3(i,j)
                pix6=pix6+1;
            end
            if Y3(i+1,j-1)==Y3(i,j)
                pix6=pix6+1;
            end
            if Y3(i+1,j)==Y3(i,j)
                pix6=pix6+1;
            end
         P(i,j)=pix;

%             %esquina inferior izquierda
        elseif i==1 & j==x
            if Y3(i,j-1)==Y3(i,j)
                pix7=pix7+1;
            end
            if Y3(i+1,j-1)==Y3(i,j)
                pix7=pix7+1;
            end
            if Y3(i+1,j)==Y3(i,j)
                pix7=pix7+1;
            end
            P(i,j)=pix7;

%             %costado lateral izquierdo
        elseif i==1 & j~=1 & j~=x
            if Y3(i,j-1)==Y3(i,j)
                pix8=pix8+1;
            end
            if Y3(i+1,j-1)==Y3(i,j)
                pix8=pix8+1;
            end
            if Y3(i+1,j)==Y3(i,j)
                pix8=pix8+1;
            end
            if Y3(i+1,j+1)==Y3(i,j)
                pix8=pix8+1;
            end
            if Y3(i,j+1)==Y3(i,j)
                pix8=pix8+1;
            end
            P(i,j)=pix8;
        end
        end
```

```
   end



% contador de pixel eleccion

R=rand(size(I));


for i=1:z
    for j=1:x
        if R(i,j)<=0.2
        R(i,j)=1;
        elseif R(i,j)>0.2 & R(i,j)<=0.4
        R(i,j)=2;
        elseif R(i,j)>0.4 & R(i,j)<=0.6
        R(i,j)=3;
        elseif R(i,j)>0.6 & R(i,j)<=0.8
        R(i,j)=4;
        elseif R(i,j)>0.8
        R(i,j)=5;
        end
    end
end


P1=rand(size(I));

for  i=1:z
    for  j=1:x
pix=0;
pix1=0;
pix2=0;
pix3=0;
pix4=0;
pix5=0;
pix6=0;
pix7=0;
pix8=0;
%           %esquina superior izquierda
          if i==1 & j==1
            if R(i+1,j)==R(i,j)
               pix=pix+1;
            end
            if R(i+1,j+1)==R(i,j)
                pix=pix+1;
            end
            if R(i,j+1)==R(i,j)
                pix=pix+1;
            end
            P1(i,j)=pix;

%             %parte superior
        elseif j==1 & i~=1 & i~=z
            if R(i+1,j)==R(i,j)
                pix1=pix1+1;
            end
            if R(i+1,j+1)==R(i,j)
                pix1=pix1+1;
            end
            if R(i,j+1)==R(i,j)
               pix1=pix1+1;
            end
             if R(i-1,j+1)==R(i,j)
```

```
                pix1=pix1+1;
            end
             if R(i-1,j)==R(i,j)
                pix1=pix1+1;
            end
            P1(i,j)=pix1;

%             %centro
        elseif j~=1 & j~=x & i~=1 & i~=z
            if R(i+1,j)==R(i,j)
                pix2=pix2+1;
            end
            if R(i+1,j+1)==R(i,j)
                pix2=pix2+1;
            end
            if R(i,j+1)==R(i,j)
                pix2=pix2+1;
            end
             if R(i-1,j+1)==R(i,j)
                pix2=pix2+1;
            end
             if R(i-1,j)==R(i,j)
                pix2=pix2+1;
            end
            if R(i-1,j-1)==R(i,j)
                pix2=pix2+1;
            end
            if R(i,j-1)==R(i,j)
                pix2=pix2+1;
            end
            if R(i+1,j-1)==R(i,j)
                pix2=pix2+1;
            end
            P1(i,j)=pix2;

%             %esquina superior derecha
        elseif i==z & j==1
            if R(i,j+1)==R(i,j)
                pix3=pix3+1;
            end
            if R(i-1,j+1)==R(i,j)
                pix3=pix3+1;
            end
            if R(i-1,j)==R(i,j)
                pix3=pix3+1;
            end
            P1(i,j)=pix3;

%             %costado lateral derecho
        elseif i==z & j~=1 & j~=x
            if R(i-1,j-1)==R(i,j)
                pix4=pix4+1;
            end
            if R(i-1,j+1)==R(i,j)
                pix4=pix4+1;
            end
            if R(i-1,j)==R(i,j)
                pix4=pix4+1;
            end
            P1(i,j)=pix4;

            %esquina inferior derecha
        elseif i==z & j==x
            if R(i-1,j)==R(i,j)
               pix5=pix5+1;
            end
            if R(i-1,j-1)==R(i,j)
               pix5=pix5+1;
            end
            if R(i,j-1)==R(i,j)
               pix5=pix5+1;
            end
            P1(i,j)=pix5;

%             %parte inferior
        elseif j==x & i~=1 & i~=z
            if R(i-1,j)==R(i,j)
               pix6=pix6+1;
            end
            if R(i-1,j-1)==R(i,j)
                pix6=pix6+1;
            end
            if R(i,j-1)==R(i,j)
                pix6=pix6+1;
            end
            if R(i+1,j-1)==R(i,j)
                pix6=pix6+1;
            end
            if R(i+1,j)==R(i,j)
                pix6=pix6+1;
            end
         P1(i,j)=pix;

%             %esquina inferior izquierda
        elseif i==1 & j==x
            if R(i,j-1)==R(i,j)
                pix7=pix7+1;
            end
            if R(i+1,j-1)==R(i,j)
                pix7=pix7+1;
            end
            if R(i+1,j)==R(i,j)
                pix7=pix7+1;
            end
            P1(i,j)=pix7;

%             %costado lateral izquierdo
        elseif i==1 & j~=1 & j~=x
            if R(i,j-1)==R(i,j)
                pix8=pix8+1;
            end
            if R(i+1,j-1)==R(i,j)
                pix8=pix8+1;
            end
            if R(i+1,j)==R(i,j)
                pix8=pix8+1;
            end
            if R(i+1,j+1)==R(i,j)
                pix8=pix8+1;
            end
            if R(i,j+1)==R(i,j)
                pix8=pix8+1;
            end
            P1(i,j)=pix8;
        end
        end

    end



% Modelo de Potts
```

```
    %Definicion del valor de beta
        B=0.35;
     %Energias
        P=P*-B;
        P1=P1*-B;


% comparacion entre pixel actual y pixel
eleccion
    %se les suma su proyeccion sobre las
clases



 for  i=1:z
    for  j=1:x

    if Y3(i,j)==1
            P(i,j)=P(i,j)+W0(I(i,j));
        elseif Y3(i,j)==2
            P(i,j)=P(i,j)+W1(I(i,j));
        elseif Y3(i,j)==3
            P(i,j)=P(i,j)+W2(I(i,j));
        elseif Y3(i,j)==4
            P(i,j)=P(i,j)+W3(I(i,j));
        elseif Y3(i,j)==5
            P(i,j)=P(i,j)+W4(I(i,j));
        end

        if R(i,j)==1
            P1(i,j)=P1(i,j)+W0(I(i,j));
        elseif R(i,j)==2
            P1(i,j)=P1(i,j)+W1(I(i,j));
        elseif R(i,j)==3
            P1(i,j)=P1(i,j)+W2(I(i,j));
        elseif R(i,j)==4
            P1(i,j)=P1(i,j)+W3(I(i,j));
        elseif R(i,j)==5
            P1(i,j)=P1(i,j)+W4(I(i,j));
        end
```

```
        if P1(i,j)<=P(i,j)
            Y3(i,j)=R(i,j);
        else
            p=exp(-((P1(i,j)-
P(i,j))/T^n));
            ale=rand;
            if ale<p
               Y3(i,j)=R(i,j);
           end
       end
    end
end

T=T*.95


end   %fin de la tercera iteracion

for  i=1:z
    for  j=1:x
        if Y3(i,j)==1
            Y3(i,j)=Med1(1);
        elseif Y3(i,j)==2
            Y3(i,j)=Med1(2);
        elseif Y3(i,j)==3
            Y3(i,j)=Med1(3);
        elseif Y3(i,j)==4
            Y3(i,j)=Med1(4);
        elseif Y3(i,j)==5
            Y3(i,j)=Med1(5);
        end
    end
end

figure
colormap(gray(256));
image(Y3);
axis('off');
```