**UNIVERSIDAD NACIONAL DE ASUNCIÓN**

**Facultad Politécnica**

# "Eliminación de ruido blanco con LPG-PCA adaptativo en dos etapas"

Trabajo Final de Grado presentado por

César Vasconsellos

como requisito
para obtener el título de Licenciado
en Ciencias Informáticas

Orientador:
Christian E. Schaerer

San Lorenzo - Paraguay. Julio de 2017

*Dedicado a mis abuelos maternos*

# Agradecimientos

A mis abuelos, Abdón Benítez y María Alice Alegre por su apoyo incondicional en cada etapa de mi vida.

A Christian Schaerer, por su extensa colaboración durante el desarrollo de este trabajo, por todos sus consejos y enseñanzas.

A Santiago Gómez, por su colaboración y enseñanzas.

# Eliminación de ruido blanco con LPG-PCA adaptativo en dos etapas

**RESUMEN:** El ruido en imágenes digitales es inevitablemente introducido tanto en su captura como en su transmisión, por lo que es necesario un procedimiento que logre filtrarlo para mejorar la calidad de la imagen. En este trabajo implementaremos la Agrupación Local de Pixeles (LPG) adaptativo conjuntamente con el Análisis de Componentes Principales (PCA) para llevar a cabo la eliminación del ruido blanco, cuyas características son media cero y desviación estándar $\sigma$.

El PCA se encarga de decorrelacionar la señal del ruido por medio de un cambio de base para luego reducir el error medio cuadrático al minimizar la varianza. El conjunto de entrenamiento del PCA es obtenido por el LPG al centrar una ventana $K$ en el pixel a ser limpiado, el cual contendrá la estructura local de la imagen, luego se centra otra ventana $L > K$ que servirá como ventana de entrenamiento. Implementando el *block matching method* se toman las muestras similares dentro de la ventana $L$.

Como la calidad de la eliminación del ruido depende de la cantidad de muestras, y en ciertos casos podría tenerse muy pocas muestras debido a ciertos patrones irregulares en la estructura local, puede tomarse una ventana $K$ de menor tamaño para eliminar este patrón irregular y obtener una mayor cantidad de muestras, aumentando la precisión del LPG y realizar mejores estimaciones en el PCA.

El mismo procedimiento se lleva a cabo pixel por pixel hasta limpiar toda la imagen. Luego, el nivel de ruido es actualizado y se vuelve a ejecutar el método propuesto para obtener mejores resultados visuales debido al fuerte ruido presente en la imagen original.

*Palabras clave: Elminación de ruido, LPG, PCA*

# Índice General

**Índice de Figuras** VIII

**Índice de Tablas** IX

**Lista de Símbolos** X

**1. Introducción** 1

1.1. Estado del arte . . . 2

1.2. Originalidad del trabajo . . . 4

1.3. Objetivos . . . 5

1.3.1. Objetivo general . . . 5

1.3.2. Objetivos específicos . . . 5

**2. Fundamentos teoricos** 6

2.1. Algebra Lineal . . . 6

2.1.1. Autovalores y autovectores . . . 6

2.1.2. Diagonalización de una matriz . . . 7

2.1.3. Matrices Hermitianas . . . 8

2.2. Probabilidades . . . 10

2.2.1. Media aritmética . . . 10

2.2.2. Varianza . . . 10

2.2.3. Covarianza . . . 11

2.2.4. Matriz de covarianzas . . . 12

2.2.5. Estimación insesgada de varianza mínima . . . . . 12

**3. Análisis de Componentes Principales 15**

3.1. Introducción . . . . . 15

3.2. Detalles técnicos del PCA . . . . . 16

**4. Agrupación Local de Pixeles 21**

4.1. Modelando la variable vectorial . . . . . 21

4.2. Block matching method . . . . . 23

**5. LPG-PCA adaptativo 26**

5.1. Limpieza basada en LPG-PCA . . . . . 26

5.2. Mejora de la eliminación de ruido en la segunda etapa . . . . . 30

5.3. Imágenes a color . . . . . 33

**6. Resultados experimentales 35**

**7. Conclusión 46**

**A. Abstract presentado en la Conference of Computational Interdisciplinary Science CCIS 2016 47**

**B. Codigo del software Eliminación de ruido blanco con LPG-PCA adaptativo en dos etapas 49**

B.1. Algoritmo de eliminación de ruido blanco en imagenes de escala de grises (Gray-Image-Denoising) . . . . . 49

B.2. Algoritmo de eliminación de ruido blanco (LPG-PCA adaptativo) . . 54

B.3. Analisis de componentes principales (pca) . . . . . 56

B.4. Agrupacion Local de Pixeles (lpg) . . . . . 57

B.5. Imagenes a color (colorDenoising) . . . . . 59

B.6. Agregar ruido (addNoise) . . . . . 64

B.7. Calidad de la imagen (calidad) . . . . . 65
B.8. Nivel de ruido (psnr) . . . . . 67
B.9. Ordenamiento (qs) . . . . . 69

# Índice de Figuras

1.1. Diagrama del algoritmo. . . . . 4

4.1. Representacion del modelado del pixel. . . . . 22

4.2. Representacion de la reduccion del tamaño de la ventana $K$. . . . . 25

5.1. (a) Imagen original, (b) Imagen ruidosa (PSNR=22.1dB), (c) Imagen resultante de la primera etapa de la eliminación de ruido (PSNR=30.34dB), (d) Imagen resultante de la segunda etapa de la eliminación de ruido (PSNR=30.97dB). . . . . 31

6.1. Imágenes utilizadas en las pruebas: Cameraman, Lenna, Monarch, House, Einstein, Peppers, Barbara (color) y Parrot (color) . . . . . 37

6.2. (a) Imagen original, (b) Imagen ruidosa (PSNR=22.1dB), (c) Imagen resultante de la primera etapa de la eliminación de ruido (PSNR=29.10dB), (d) Imagen resultante de la segunda etapa de la eliminación de ruido (PSNR=29.33dB). . . . . 39

6.3. (a) Imagen original, (b) Imagen ruidosa (PSNR=22.1dB), (c) Imagen resultante de la primera etapa de la eliminación de ruido (PSNR=31.86dB), (d) Imagen resultante de la segunda etapa de la eliminación de ruido (PSNR=32.82dB). . . . . 40

6.4. (a) Imagen original, (b) Imagen ruidosa (PSNR=22.1dB), (c) Imagen resultante de la primera etapa de la eliminación de ruido (PSNR=29.75dB), (d) Imagen resultante de la segunda etapa de la eliminación de ruido (PSNR=30.23dB). . . . . 41

6.5. (a) Imagen original, (b) Imagen ruidosa (PSNR=22.1dB), (c) Imagen resultante de la primera etapa de la eliminación de ruido (PSNR=30.66dB), (d) Imagen resultante de la segunda etapa de la eliminación de ruido (PSNR=31.21dB). . . . . 42

6.6. (a) Imagen original, (b) Imagen ruidosa (PSNR=22.1dB), (c) Imagen resultante de la primera etapa de la eliminación de ruido (PSNR=30.30dB), (d) Imagen resultante de la segunda etapa de la eliminación de ruido (PSNR=30.76dB). . . . . 43

6.7. (a) Imagen original, (b) Imagen ruidosa (PSNR=22.1dB), (c) Imagen resultante de la primera etapa de la eliminación de ruido (PSNR=31.96dB), (d) Imagen resultante de la segunda etapa de la eliminación de ruido (PSNR=32.15dB). . . . . 44

6.8. (a) Imagen original, (b) Imagen ruidosa (PSNR=22.1dB), (c) Imagen resultante de la primera etapa de la eliminación de ruido (PSNR=32.05dB), (d) Imagen resultante de la segunda etapa de la eliminación de ruido (PSNR=32.31dB). . . . . 45

# Índice de Tablas

6.1. PSNR y SSIM de las imágenes luego de la eliminación de ruido con el método “Eliminación de ruido blanco con LPG-PCA adaptativo en dos etapas" . . . . . 37
6.2. Comparación de los métodos LPG-PCA adaptativo y LPG-PCA con tamaño fijo de ventana K . . . . . 38

# Lista de Símbolos


| | | |
|---|---|---|
| $\lambda$ | Autovalor | 6 |
| $I$ | Matriz identidad | 6 |
| $\Lambda$ | Matriz diagonal de autovalores | 7 |
| $S$ | Matriz vector caracteristico | 7 |
| $Q$ | Matriz ortonormal | 9 |
| $X$ | Conjunto de muestras | 10 |
| $\bar{X}$ | Conjunto de muestras centralizado | 10 |
| $N$ | Cantidad de muestras | 10 |
| $\sigma^2$ | Varianza | 11 |
| $C$ | Matriz de covarianzas | 12 |
| $\hat{\theta}$ | Estimador estadistico | 13 |
| $\mu$ | Promedio | 12 |
| $\hat{w}$ | Factor de encojimiento de la varianza | 14 |
| $s^2$ | Estimador de varianza | 14 |
| $\Phi^T$ | Matriz de transformacion ortonormal | 16 |
| $\in$ | Pertenece a | 17 |
| $\mathbb{R}$ | Conjunto de los numeros reales | 17 |
| $\Omega$ | Matriz de covarianzas | 17 |
| $e_i$ | Error medio cuadratico | 23 |
| $T$ | Threshold | 23 |

# Capítulo 1

# Introducción

Uno de los métodos predominantes de transmitir información es por medio de imágenes digitales. En la adquisición y transmisión de imágenes, inevitablemente estas adquieren ruido aditivo debido al mecanismo con el que fueron capturadas y transmitidas. Este ruido afecta a su calidad, el cual debe ser filtrado.

La característica principal del ruido blanco es que afecta a cada pixel con valores aleatorios que no guardan correlación entre si, tiene media cero y desviación estándar $\sigma$.

La eliminación de ruido es una técnica de manipulación de imágenes digitales cuyo objetivo es remover el ruido aditivo, manteniendo las características de la imagen para obtener la mayor calidad posible. En la actualidad existen varios métodos de eliminación de ruido, técnicas recientemente desarrolladas como curvlets [JD02], ridgelets [CK07] y wavelets [JS03]. Sparse representation [EA06], método K-SVD [MB06], shape-adaptive transform [AE07], bilateral filtering [TM98], non-local mean based methods [KB06] y non-local-collaborative filtering [Egi07].

En este trabajo presentamos un método eficiente de eliminación de ruido aditivo basado en análisis de componentes principales (PCA) utilizando agrupación local de pixeles (LPG) adaptativo.

## 1.1. Estado del arte

El PCA es una técnica estadística utilizada para el análisis exploratorio de datos. También es utilizada para reducir la dimensión de la información. Es decir, en un conjunto de datos, reducir la cantidad de variables manteniendo la mayor cantidad de información posible.

Permite identificar patrones en un conjunto multivariante, es una transformación ortonormal que busca las direcciones de mayor varianza en un conjunto de datos y expresarlos de tal manera que destaquen sus similitudes y diferencias. Dado que los patrones en los datos pueden ser difíciles de encontrar en conjuntos de alta dimensión, donde el lujo de la representación gráfica no está disponible, PCA es una poderosa herramienta para analizar los datos.

Una vez encontrado el patrón en el conjunto de datos, este puede comprimirse reduciendo la cantidad de dimensiones sin perder mucha información, como en la compresión de imágenes. Como veremos más adelante, PCA también puede usarse para la eliminación de ruido aditivo.

El LPG-PCA [LS10] es un método eficaz de remover el ruido. El LPG, se encarga de obtener las muestras para entrenar al PCA, el cual por medio de un cambio de base logra decorrelacionar la señal del ruido.

El objetivo es transformar el conjunto de datos original en el dominio del PCA, preservando solo los componentes principales. Con este fin, se utiliza la técnica de agrupación local de pixeles (LPG), modelando cada pixel con sus vecinos más cercanos como una variable vectorial, una ventana $K$ móvil se utilizara para la selección de muestras centrando otra ventana de tamaño $L$ mayor a $K$, el cual será la ventana de entrenamiento.

El *Block Matching Method* permite seleccionar solo las muestras similares correspondientes al pixel a ser limpiando utilizando como métrica de similitud la distancia cuadrática entre las muestras y el bloque central a ser limpiado, eliminando el ruido luego de aplicar la técnica *linear mínimum mean square-error estimation* (LMMSE), preservando los bordes, manteniendo lo mejor posible las características locales de la imagen.

La mayor parte del ruido es eliminado en la primera etapa, pero debido al fuerte ruido que afecta a la imagen original, el proceso debe volver a llevarse a cabo una vez que el nivel del ruido es actualizado para obtener mejores resultados visuales. Para actualizar el nivel de ruido se compara pixel por pixel la imagen ruidosa con el resultado de la primera etapa.

El procedimiento en la segunda etapa es el mismo que en la primera, con la diferencia en el nivel de ruido, como se muestra en la figura.

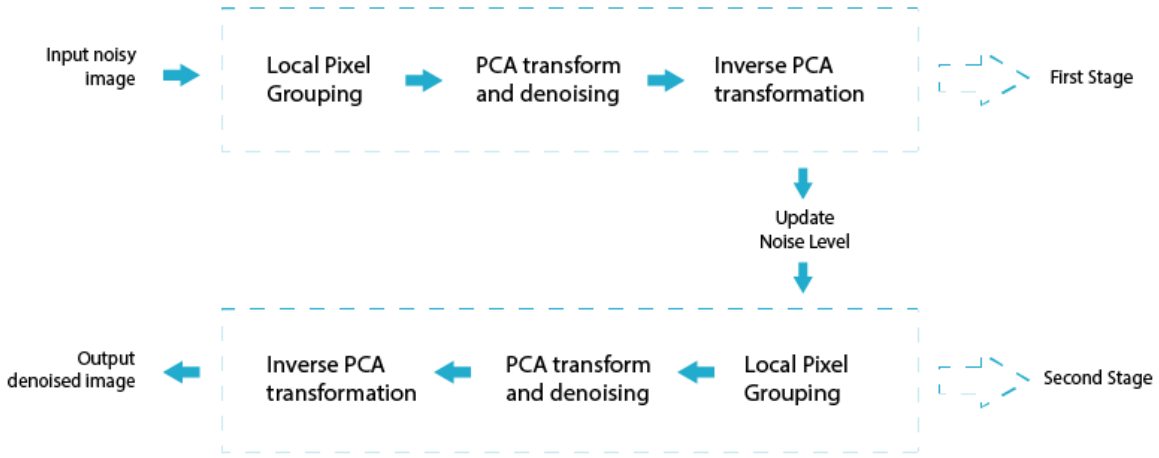


Figura 1.1: Diagrama del algoritmo.

## 1.2. Originalidad del trabajo

El algoritmo LPG que se usa en métodos basados en PCA usa una ventana $K$ fija a lo largo de toda la imagen. Al no ser flexible la dimensión de la ventana $K$, se utiliza el mismo criterio de búsqueda de muestras para cada pixel. Sin embargo, debido a las distintas características que se encuentran en las imágenes, podria obtenerse muy pocas muestras, por lo que se toma una cantidad minima independientemente de que todas las muestras obtenidas sean aceptadas en el *block matching method*, lo que genera en ciertos casos borrosidad en bordes o patrones que no son frecuentes en la imagen. Esta nueva técnica propone utilizar tamaños de ventana $K$ variable, dependiendo de la complejidad de la estructura local de la imagen, permitiendo al LPG reconocer con mayor eficiencia las características de la imagen al reducir el tamaño de la ventana $K$ cuando se encuentran patrones irregulares, ya sean bordes irregulares o estructuras en la imagen no frecuentes.

## 1.3. Objetivos

### 1.3.1. Objetivo general

Explorar la utilización de ventanas $K$ de tamaño variable en el LPG para mejorar su precisión y obtener mejores estimaciones de la matriz de transformación PCA.

### 1.3.2. Objetivos específicos

1. Experimentar distintos tamaños de ventana $K$ para la eliminación de ruido.
2. Determinar el criterio óptimo para adaptar el tamaño de la ventana $K$.
3. Explorar distintas medidas de *threshold* al ajustar el nivel de ruido para la segunda etapa del algoritmo.

# Capítulo 2

# Fundamentos teoricos

## 2.1. Algebra Lineal

El álgebra lineal es utilizado por el PCA para reconocer los patrones de las estructuras locales en las imágenes, por lo que es necesario una revisión de ciertos conceptos de esta materia.

### 2.1.1. Autovalores y autovectores

Sea la matriz $A \in \mathbb{R}_{nxn}$ y $\lambda \in \mathbb{R}$. $\lambda$ es un autovalor de $A$ si existe un vector no nulo $x \in \mathbb{R}$ tal que:

$$Ax = \lambda x. \tag{2.1}$$

Si existe $x$ que cumpla la condición, entonces $x$ es un autovector de $A$.

Ahora, en vez de escribir $\lambda x$, podría escribirse $\lambda I x$, siendo $I$ la matriz identidad, para pasar este término al lado izquierdo de la ecuación, de modo que:

$$(A - \lambda I)x = 0. \tag{2.2}$$

Entonces podemos ver que $x$ esta en el espacio nulo de $A - \lambda x$ y que $\lambda$ toma un valor de modo a que $A - \lambda I$ tenga un espacio nulo.

El objetivo es encontrar valores $\lambda$ para aquellos vectores distintos del cero, ya que el vector $x = 0$ siempre satisface esta condición. Para que la solución sea distinta de la trivial ($x = 0$), es necesario que la determinante de $A - \lambda I$ sea cero, así:

$$det(A - \lambda I) = 0. \tag{2.3}$$

El cual es llamado polinomio característico. El espacio nulo de $A - \lambda I$ es conocido como espacio característico, y todos los vectores dentro de este espacio son autovectores que cumplen con $Ax = \lambda x$.

Para encontrar los autovectores, primero debe calcularse el determinante de $A - \lambda I$ y encontrar las raíces de este polinomio. Entonces para cada autovalor se debe resolver la ecuación $(A - \lambda I)x = 0$.

### 2.1.2. Diagonalización de una matriz

Por definición $A$ y $B$, dos matrices cuadradas son equivalentes cuando existe una matriz cuadrada $P$ con $det\ P \neq 0$ tal que $A = PBP^{-1}$.

Una matriz diagonalizable es una matriz cuadrada que es equivalente a una matriz diagonal. Si $A_{nxn}$ posee $n$ vectores característicos linealmente independientes, y estos vectores forman las columnas de una matriz $S$, entonces $SAS^{-1} = \Lambda$, siendo $\Lambda$ una matriz diagonal que contiene a los valores característicos de $A$.

$SAS^{-1} = \Lambda = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda n \end{bmatrix}$. La matriz $S$ es una matriz invertible debido a que sus columnas son linealmente independientes y es llamado matriz vector característico, y la matriz $\Lambda$ es la matriz valor característico.

Para demostrar este hecho colocaremos los vectores característicos $x_i$ como columnas de $S$, $AS$ se calcula por columnas:

$$AS = A \begin{bmatrix} | & | & & | \\ x_1 & x_2 & \dots & x_n \\ | & | & & | \end{bmatrix} = \begin{bmatrix} | & | & & | \\ \lambda_1.x_1 & \lambda_2.x_2 & \dots & \lambda_n.x_n \\ | & | & & | \end{bmatrix}. \tag{2.4}$$

Y luego separemos la matriz $AS$ en un producto de la siguiente manera:

$$\begin{bmatrix} | & | & & | \\ \lambda_1.x_1 & \lambda_2.x_2 & \dots & \lambda_n.x_n \\ | & | & & | \end{bmatrix} = \begin{bmatrix} | & | & & | \\ x_1 & x_2 & \dots & x_n \\ | & | & & | \end{bmatrix} \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda n \end{bmatrix}. \tag{2.5}$$

Si se mantienen las matrices en el orden correcto, $AS = S\Lambda$, y despejando $A$ queda: $A = S\Lambda S^{-1}$.

### 2.1.3. Matrices Hermitianas

Una matriz hermitiana, es una matriz que es igual a su transpuesta conjugada, cuyos elementos en la diagonal son reales.
$A = \begin{bmatrix} a & b+bi \\ b-bi & c \end{bmatrix} = A^H$. Las propiedades de las matrices hermitianas son igualmente validas para las matrices simétricas reales, por lo tanto, las matrices simétricas también son hermitianas, es decir $A^T = A^H$.

La primera propiedad dice que si $A = A^H$, entonces para todos los vectores $x$ que son complejos, el número $x^H Ax$ es real. Para demostrar esta propiedad tomemos el conjugado de la matriz de 1 por 1 $x^H Ax$, el cual es $(x^H Ax)^H$, pero se obtiene de vuelta el mismo número ya que $(x^H Ax)^H = x^H A^H x^H H = x^H Ax$. Por lo podemos

afirmar que este número es real.

La segunda propiedad dice que si $A = A^H$, todo valor característico es real. Sea $Ax = \lambda x$, multiplicamos cada miembro por $x^H$, de modo que $x^H Ax = \Lambda x^H x$. Por la primera propiedad, el primer miembro es real, y el segundo es real y positivo ya que $x^H x = \|x\|^2$ y $x$ es distinto de cero. Así que $\lambda = x^H Ax/x^H x$ debe ser real.

La tercera propiedad dice que los vectores de una matriz simétrica real o hermitiana son ortogonales si sus valores característicos son distintos.
Si $Ax = \lambda_1 x$,$Ay = \lambda_2 y$, y$A = A^H$, entonces:

$$(\lambda_1 x)^H y = (Ax)^H y = x^H Ay = x^H(\lambda_2 y). \tag{2.6}$$

Debido a que $\lambda_1 \neq \lambda 2$, fuerza a que $x^H y = 0$, es decir, estos vectores son ortogonales. Si normalizamos estos vectores, ahora son ortonormales. Y si utilizamos estos vectores como las columnas de la matriz $S$, entonces tenemos $S^{-1}AS = \Lambda$. En caso de que $A$ sea real y simétrica, por la segunda propiedad sabemos que sus valores característicos son reales, y por la tercera propiedad que sus vectores característicos unitarios son ortogonales y reales.

Si estos vectores característicos unitarios forman la matriz ortogonal $Q$, por lo que $Q^T Q = I$ y también $Q^{-1} = Q^T$. Entonces tenemos que $S^{-1}AS = \Lambda$ pasa a ser $Q^{-1}AQ = \Lambda$ y $A = Q\Lambda Q^{-1} = Q\Lambda Q^T$.

Ahora podemos enunciar uno de los grandes teoremas del álgebra lineal que dice que una matriz simétrica real puede factorizarse en $A = Q\Lambda Q^T$. Siendo $Q$ la matriz que contiene a los vectores característicos y $\Lambda$ la matriz diagonal que contiene a los valores característicos. Este teorema es conocido como el teorema espectral.

## 2.2. Probabilidades

Antes de realizar cualquier estudio estadístico, lo primero a realizarse es la recolección y el ordenamiento de los datos. Generalmente la recogida de muestras se hace en función de una muestra de algún conjunto mayor que desea estudiarse. Para esto, describiremos los conceptos que serán utilizados tanto en el LPG como en el PCA.

### 2.2.1. Media aritmética

La media aritmética o media de un conjunto de valores $X = [x_1, x_2, ..., x_n]$, el cual es representado por $\bar{X}$. Es el resultado de dividir la suma de todos los valores de una variable entre el número total de observaciones. Y está definida por:

$$\bar{X} = \frac{\sum_{i=1}^{n} x_i}{N}. \tag{2.7}$$

Siendo $x_i$ los valores que toma la variable $X$, y $N$ la cantidad de experimentos o valores que esta toma.

La media aritmética se utilizara para centralizar el conjunto de datos.

### 2.2.2. Varianza

La media no otorga mucha información acerca del conjunto de datos ya que solo representa su punto central. Sin embargo, calculando la varianza, se puede medir la dispersión existente entre los valores que toma la variable con relación a la media aritmética. Cuando mayor sea la varianza, mayor será la dispersión, por lo que la media aritmética pierde representatividad.

Para su cálculo, la sumatoria de cada valor que toma la variable menos su media, al cuadrado, se divide sobre $N$.

$$\sigma^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{X})^2}{N}. \tag{2.8}$$

Como la diferencia entre cada valor $x_i$ y la media $\bar{X}$ se eleva al cuadrado, se sabe que la varianza siempre será mayor o igual a cero.

### 2.2.3. Covarianza

Los dos conceptos mencionados hasta ahora son aplicados solamente a problemas de una sola dimensión. Sin embargo, muchos conjuntos de datos tienen más de una dimensión, y el objetivo del análisis estadístico de este tipo de conjuntos suele ser ver si existe alguna relación entre las dimensiones.

La varianza trabaja en una sola dimensión, por lo que es posible calcular la varianza de cada dimensión del conjunto de datos respecto a la otra de forma independiente. Sin embargo, es útil poder medir cuanto varia cada dimensión de la media con respecto a cada una.

La covarianza se utiliza para medir como varían de forma conjunta dos variables $X$ e $Y$. Es decir, en dos dimensiones. Suponiendo que se calcula la covarianza entre una dimensión consigo misma, lo que en realidad se obtiene es su varianza. Pero si se tienen, por ejemplo, tres variables, se podría calcular la covarianza entre $X$ con $Y$, $X$ con $Z$ e $Y$ con $Z$ de la siguiente manera:

$$cov(X,Y) = \frac{\sum_{i=1}^{n}(x_i - \bar{X})^2.(y_i - \bar{Y})^2}{N}. \tag{2.9}$$

1. $cov(X,Y) > 0$ implica que la relación es positiva o directa, y que cuando una variable incrementa, la otra también lo hace

2. $cov(X, Y) = 0$ implica que no hay relación lineal entre las variables.

3. $cov(X, Y) < 0$ implica que la relación es negativa o inversa, y que cuando una variable incrementa, la otra decrece.

### 2.2.4. Matriz de covarianzas

Una manera útil de representar todas las posibles covarianzas entre las diferentes dimensiones es calculándolas e introduciéndolas en una matriz. Como ejemplo usaremos tres dimensiones. Entonces la matriz de covarianzas queda definida de la siguiente manera:

$$C = \frac{1}{n}\overline{XX}^T = \begin{bmatrix} cov(x,x) & cov(x,y) & cov(x,z) \\ cov(y,x) & cov(y,y) & cov(y,z) \\ cov(z,x) & cov(z,y) & cov(z,z) \end{bmatrix}, \tag{2.10}$$

donde $X$ es el conjunto de datos, cada columna es una muestra, y cada fila contiene las variables correspondientes a cada una. Se puede observar que los elementos de la diagonal son las varianzas de cada una de las dimensiones, y debido a que $\frac{\sum_{i=1}^{n}(x_i-\bar{X})^2.(y_i-\bar{Y})^2}{N} = \frac{\sum_{i=1}^{n}(y_i-\bar{Y})^2.(x_i-\bar{X})^2}{N}$ ya que la multiplicación es conmutativa, entonces $C_{i,j} = C_{j,i}$, lo que significa que la matriz de covarianzas es simétrica.

En los capítulos siguientes se verá como el PCA utiliza la matriz de covarianzas para poder de-correlacionar la señal de la imagen del ruido.

### 2.2.5. Estimación insesgada de varianza mínima

El error cuadrático medio (MSE por sus siglas en inglés)

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\bar{x} - \mu)^2, \tag{2.11}$$

de un estimador mide el promedio de los errores al cuadrado, lo que equivale a la diferencia entre el estimador y lo que se está estimando.

El MSE de un estimador $\hat{\theta}$ con respecto al parámetro desconocido $\theta$ se define como:

$$MSE(\hat{\theta}) = E[(\hat{\theta} - \theta)^2]. \tag{2.12}$$

Se considera un procedimiento para minimizar el MSE de un estimador insesgado lineal de varianza mínima (MVULE por sus siglas en inglés) de un conjunto de muestras con un comportamiento normal. Un estimador insesgado de varianza mínima es aquel con menor varianza que cualquier otro estimador insesgado. El objetivo es encoger el MVULE hacia su origen natural $\mu$, para el cual [Tho68] sugiere el siguiente estimador:

$$\hat{\mu} = \frac{(\bar{x} - \mu_0)^2}{(\bar{x} - \mu_0)^2 + S^2/n} + \mu_0. \tag{2.13}$$

En el caso de una distribución normal, tenemos $\mu_0 = 0$.

Para encoger el MVULE hacia su origen natural, se considera multiplicarlo por un factor de encogimiento $w$. Este origen natural puede surgir porque creemos que el estimador esta cerca del valor real.

Debido a que el MSE es igual a la suma de la varianza y el cuadrado del sesgo del estimador, tenemos:

$$MSE(\bar{\theta}) = Var(\bar{\theta}) + (sesgo(\bar{\theta}, \theta)^2). \tag{2.14}$$

El sesgo de un estimador es la diferencia entre su esperanza y el valor numero del parámetro que estima. En general, no son realizables los estimadores que minimizan

el MSE debido a que dependen del parámetro desconocido (sesgo).

Sin embargo, como el MSE de estimadores insesgados es la varianza, minimizar la varianza equivale a minimizar el MSE.

Se dispone de los estimadores de $\mu$ y $\sigma^2$. Siendo $\bar{x}$ un estimador de $\mu$ y $s^2 = \frac{1}{n}\sum_{j=1}^{n}(x_j - \bar{x})^2$ un estimador para $\sigma^2$, entonces tenemos que

$$\hat{w}\bar{x} = \frac{\bar{x}^2}{\bar{x}^2 + s/n}\bar{x}. \tag{2.15}$$

Siendo $0 < w \leq 1$.

Esta estimación se utiliza en la eliminación de ruido al encoger la varianza del conjunto de datos hacia su origen natural.

# Capítulo 3

# Análisis de Componentes Principales

## 3.1. Introducción

En espacios de muchas dimensiones resulta difícil ver algún tipo de información en los datos, por lo que un método analítico resulta necesario para encontrar patrones en un gran conjunto de datos.

PCA es una técnica matemática que se utiliza para transformar un conjunto de variables correlacionadas en un pequeño conjunto de variables llamadas componentes principales. Esto lo lleva a cabo identificando los ejes principales ortonormales que retienen la mayor varianza a través de los datos. Una vez obtenido el primer componente principal (el que retiene la mayor varianza), se prosigue a obtener el segundo componente principal (el que retiene la segunda mayor varianza), por medio de una proyección ortogonal para las nuevas coordenadas del nuevo eje, y así sucesivamente hasta obtener todos los ejes principales ortogonales entre sí que retienen el 100 % de la varianza de los datos.

El PCA es entonces un método que expresa de forma óptima los datos por medio de una nueva base que es combinación lineal de la base del conjunto de datos original.

## 3.2. Detalles técnicos del PCA

Sea $X_{mxn}$ que representa al correspondiente conjunto de datos,
$X = \begin{bmatrix} x_1 & x_2 & \dots & x_m \end{bmatrix}^T$ es la variable vectorial correspondiente a cada muestra de:

$$X = \begin{bmatrix} x_1^1 & x_1^2 & \dots & x_1^n \\ x_2^1 & x_2^2 & \dots & x_2^n \\ \vdots & \vdots & \ddots & \vdots \\ x_m^1 & x_m^2 & \dots & x_m^n \end{bmatrix}, \tag{3.1}$$

donde cada $X_i^j, i = 1, 2, ...n$, son las muestras de las variables $X_j = 1, 2, ..., m$. La i-ésima fila de la matriz de muestras $X$:

$$X_i = \begin{bmatrix} x_j^1 & x_j^2 & \dots & x_j^n \end{bmatrix}, \tag{3.2}$$

es llamado el vector muestra de $x_i$. El objetivo es encontrar una matriz $Y$, también de dimensiones $m$ y $n$, que sea transformación lineal de $X$, así $Y = PX$, el cual representa el cambio de base. Entonces primero necesitamos calcular alguna matriz $P$ de $mxm$.

Siendo $P_i, i = 1, 2, ...m,$, las filas de $P$, y $X_i, i = 1, 2, ...n,$, las columnas de $X$

$$PX = \begin{pmatrix} Px_1 & Px_2 & \dots & Px_n \end{pmatrix} = \begin{bmatrix} P_1.x_1 & P_1x_2 & \dots & P_1.x_n \\ P_2.x_1 & P_2.x_2 & \dots & P_2.x_n \\ \vdots & \vdots & \ddots & \vdots \\ P_m.x_1 & P_m.x_2 & \dots & P_m.x_n \end{bmatrix} = Y. \tag{3.3}$$

Cada elemento de $Y$ es un producto interno $p_i, x_j$, lo que nos dice que $X$ está siendo proyectado en las columnas de $P$. Esto implica que las filas de $P$ son la nueva base representando las columnas de $X$, en otras palabras, son las direcciones de los componentes principales.

Al buscar los ejes que maximizan la varianza, el Análisis de componentes principales logra de-correlacionar los datos originales, y luego utiliza estos ejes como la nueva base.

Siendo $m$ el número de variables y $n$ el número de muestras. Podemos ver a $X$ en términos de $m$ vectores fila, cada uno con $n$ elementos.

$$X = \begin{bmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,n} \\ x_{2,1} & x_{2,2} & \dots & x_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m,1} & x_{m,2} & \dots & x_{m,n} \end{bmatrix} = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_m \end{bmatrix} \in \mathbb{R}^{mxn}, X_i^T \in \mathbb{R}^n. \tag{3.4}$$

Debido a que cada variable es un vector fila, todas las muestras están contenidas para una variable en específico. Entonces podemos computar la matriz de covarianzas.

Para obtener la matriz de covarianzas, primero centralizamos la matriz. Luego, el promedio de cada fila es calculado como sigue:

$$\mu_i = \frac{1}{n} \sum_{j=1}^{n} X_i(j), \tag{3.5}$$

y luego se resta este promedio a cada fila para centralizar X:

$$\overline{X}_i = X_i - \mu_i = \begin{bmatrix} \overline{x}_i^1 & \overline{x}_i^2 & \dots & \overline{x}_i^n \end{bmatrix}, \tag{3.6}$$

donde $\overline{x}_i^j = x_i^j - \mu_i$. Así, la matriz centralizada de X:

$$\overline{X} = \begin{bmatrix} \overline{X}_1^T & \overline{X}_2^T & \dots & \overline{X}_m^T \end{bmatrix}^T. \tag{3.7}$$

Por lo que la matriz de covarianzas de la matriz centralizada puede calcularse multiplicando la matriz de muestras $\overline{X}$ por su transpuesta $\overline{X}^T$, y multiplicando cada componente por $\frac{1}{n}$:

$$\Omega = \frac{1}{n}\overline{XX}^T = \begin{bmatrix} x_1.x_1^T & x_1.x_2^T & \dots & x_1.x_m^T \\ x_2.x_1^T & x_2.x_2^T & \dots & x_2.x_m^T \\ \vdots & \vdots & \ddots & \vdots \\ x_m.x_1^T & x_m.x_2^T & \dots & x_m.x_m^T \end{bmatrix}. \tag{3.8}$$

Una vez calculada la matriz de covarianzas, podemos observar que cada elemento fuera de la diagonal es la covarianza de cada par posible de las $m$ variables, y en la diagonal principal tenemos la varianza de cada una de ellas. Entonces la matriz de covarianzas nos muestra que tan correlacionadas está cada par de variables.

Debido a que el objetivo del PCA es de-correlacionar los datos originales mientras se trata de maximizar la varianza de cada una de las variables (los elementos dentro de la diagonal de $\Omega$), por medio de un cambio de base, busca que la correlación entre cada par de variables posibles sea lo más cercano a cero en la matriz transformada. Es lo mismo que decir que las covarianzas de los elementos fuera de la diagonal en $\Omega$, deben ser lo más cercano posible del cero. Esto se hace para eliminar el ruido luego de aplicar el LMMSE.

Ahora podemos decir que lo que se busca es una matriz $P$ que diagionalice a $X$, con las máximas variaciones dentro de la diagonal. Una vez obtenida la matriz $Y = PX$, podemos decir que cumplimos con nuestro objetivo.

Teniendo en cuenta que la matriz de transformación $P$ es una matriz ortonormal. La expresión de $\Omega_y$ puede ser escrita como:
$\Omega_y = \frac{1}{n}YY^T = \frac{1}{n}(PX)(PX)^T = \frac{1}{n}(PX)(P^TX^T) = \frac{1}{n}P(XX^T)P^T = P\frac{XX^T}{n}P^T = P\Omega P^T$.
Debido a que $\Omega = \frac{1}{n}XX^T$, sabemos que es una matriz simétrica.

Por la propiedad 2 de las matrices hermitianas, sabemos que los valores característicos de $\Omega$ son reales, y por la propiedad 3, que sus autovectores unitarios son reales y ortogonales. Entonces podemos hacer referencia al teorema del álgebra lineal que dice que toda matriz simétrica real puede diagonalizarse. Por lo que $\Omega$ puede ser escrito como:

$$\Omega = \Phi\Lambda\Phi^T. \tag{3.9}$$

Donde $\Phi_{m,m}$ es una matriz ortonormal cuyas columnas son los autovectores de $\Omega$, y $\Lambda = diag\left[\lambda_1, \lambda_1, ..., \lambda_m\right]$ es una matriz diagonal cuyos elementos son los autovalores de $\Omega$.

Una vez diagonalizada $\Omega$, podemos elegir la matriz de transformación PCA, $P$. Si elegimos las filas de $\Phi$, como autovectores de $\Omega$, entonces $P = \Phi^T$. Y sustituyendo $P$ en la expresión de covarianzas de $\Omega_Y$, tenemos que $\Omega_y = \frac{1}{n}P\Omega_X P^T = \frac{1}{n}P^T(P\Lambda P^T)P$. Pero debido a que $P$ es una matriz ortonormal, $PP^T = I$, siendo $I$ la matriz identidad. Entonces, eligiendo adecuadamente $P$, tenemos: $\Omega_Y = \frac{1}{n}\Lambda$.

En conclusión. PCA es un método que nos permite tener información mediante la varianza de los datos. Siendo cada componente principal, aquellos ejes que retienen la mayor varianza de manera descendente, así, el primer componente principal retiene la mayor varianza, el segundo componente principal retiene la segunda mayor varianza, y así sucesivamente. Estos componentes principales se obtienen al ordenar de manera descendente los autovalores de $\Omega_x$ y colocarlos en la diagonal principal de $\Lambda$, y colocando sus autovectores (las filas de $P$) en el mismo orden dentro de la matriz de transformación ortonormal $P$. Siendo aquellos autovectores correspondientes a los autovalores de mayor valor, los autovectores de mayor importancia.

$P$ es la matriz de transformación ortonormal que decorrelaciona a $\bar{X}$, así $\bar{Y} = P\bar{X}$, por lo que la matriz de covarianzas de $Y$ es diagonal y decorrelaciona por completo a $\bar{X}$. Por lo que la energía de la señal se concentra en un subconjunto de los componentes principales en el dominio del PCA.

# Capítulo 4

# Agrupación Local de Pixeles

## 4.1. Modelando la variable vectorial

El ruido $V$ que afecta a la imagen tiene media cero y desviación estándar $\sigma$. Siendo $I$ la imagen original, al agregarle este ruido se obtiene una nueva imagen, $I_v = I + V$. Se sabe también que no existe correlación entre la imagen $X$ y el ruido $V$.

El objetivo de esta investigación es obtener una imagen $\tilde{I}$ lo más parecida posible a $I$. Los pixeles se identifican en función de las coordenadas espaciales y su valor de escala de grises (valor de intensidad), mientras la estructura local está definida por los pixeles más cercanos con distintos valores de intensidad.

Se desea que la eliminación del ruido preserve los bordes, por lo que se modela el pixel con sus vecinos más cercanos como una variable vectorial, la cual preservara la estructura local de la imagen para llevar a cabo la eliminación de ruido, por lo que se limpia toda la variable vectorial y no solo el pixel central.

Para llevar a cabo esta tarea se centra una ventana de $k$x$k$ en el pixel a ser limpiado, entonces la variable vectorial queda definida como $x = [x_1, ..., x_m]^T$, siendo $m = K^2$ el total de componentes. Debido a que esta variable vectorial esta afectada por el

ruido, se representa como $X_v = X + v$. Siendo $X_v = [x_v^1 ... X_m^v]$ el vector ruidoso, $X = [x_1 ... X_m]^T$ la versión sin ruido de la variable vectorial y $V = [v_1 ... v_m]^T$ la variable vectorial del ruido. Una vez modelada la variable vectorial, puede utilizarse el PCA para llevar a cabo la eliminación de ruido. Para remover el ruido de la imagen, se calcula la matriz de covarianzas de X, y la matriz de transformación PCA. Para ello, se necesita un conjunto de muestras Xv, que serán obtenidas al centrar una ventana LxL con L>K en el pixel a ser limpiado.

La ventana $K$, es una ventana móvil que recorrerá la ventana $L$ para la obtención de muestras, por lo que se puede obtener como máximo $(L-K+1)^2$ muestras. Pero como pueden haber muestras muy distintas dentro de la ventana $L$, debido a las distintas estructuras encontradas en la imagen, el PCA puede incurrir en una mala estimación de la matriz de covarianzas, y por ende en la matriz de transformación, lo que llevaría a resultados aún más ruidosos. Es por eso que el conjunto de muestras de entrenamiento debe ser lo más parecido posible al bloque que se desea limpiar. Este problema se abordará en la siguiente sección.

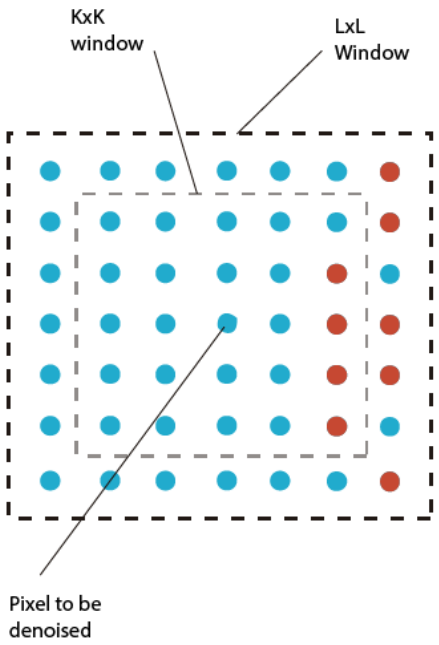


Figura 4.1: Representacion del modelado del pixel.

## 4.2. Block matching method

Debido al problema en la selección de muestras de entrenamiento, se necesita un método que lleve a cabo esta selección. Distintos métodos pueden ser utilizados, como: *block matching, correlation-based matching, K-means,* etc. Pero por cuestiones de simplicidad, nosotros utilizamos el *block matching method*, el cual está compuesto de tres puntos: definir el bloque, el método de búsqueda, y el criterio de concordancia.

Como ya mencionamos, el bloque está definido por la ventana $K$ centrada en el pixel a ser limpiado, el método de búsqueda es centrando otra ventana $L > K$ que contiene todos los bloques $k$x$k$ posibles, y el criterio de concordancia es la distancia cuadrática entre cada bloque y el bloque central que está definido por:

$$e_i = \frac{1}{m}\sum_{k=1}^{m}(x_0^v(k) - x_i^v(k))^2 \approx \frac{1}{m}\sum_{k=1}^{m}(x_0(k) - x_i(k))^2 + 2\sigma^2. \tag{4.1}$$

Donde $x_0^v$ es el vector columna correspondiente al bloque ruidoso que se desea limpiar, $x_i^v$ corresponde a todos los bloques ruidosos con los que es comparado el bloque central, $x_0$ corresponde a la versión sin ruido del bloque central, y $x_i$ corresponde a la versión sin ruido de los bloques con los que el bloque central es comparado. La ecuación 4.1 se vale del hecho de que la señal y el ruido no están correlacionados. Entonces solo serán aceptadas aquellas muestras que tengan un valor por debajo del *threshold* fijado:

$$e_i < T + 2\sigma^2, \tag{4.2}$$

Donde $T$ es el *threshold* que puede ser fijado por el programador.

La calidad de la limpieza depende de la cantidad de muestras que fueron obtenidas en el LPG, como máximo pueden obtenerse $(L - K + 1)^2$ muestras, pero debido a las diferentes características en las estructuras de la imagen que pueden estar con-

tenidas dentro de la ventana $L$, es probable que se obtenga una menor cantidad al aplicar el *Block Matching Method*.

Para llevar a cabo el PCA de manera optima, deben haber suficiente cantidad de muestras de entrenamiento. En [LS10] utilizan al menos $c.m$ muestras, donde $c = 8 \sim 10$ es una constante. De esta manera se logra garantizar una cantidad mínima de muestras, aunque no todas sean similares al bloque que se desea limpiar.

Con el nuevo enfoque presentado, por defecto, el tamaño del bloque central es $K = 5$, ya que encontramos que es el valor que mejor captura las características de la imagen, pero en zonas de la imagen en donde se encuentran pocas muestras, debido a un patrón irregular en su estructura, o debido a un borde irregular, se puede reducir el tamaño de la ventana $K$ posibilitando la omisión de este patrón irregular, permitiendo una mejor precisión en el LPG con la obtención de una mayor cantidad de muestras, mejorando la calidad de la limpieza. El criterio utilizado para reducir el tamaño de la ventana $K$ a 3 es que siendo $K = 5$, se deben obtener menos de 5 muestras.

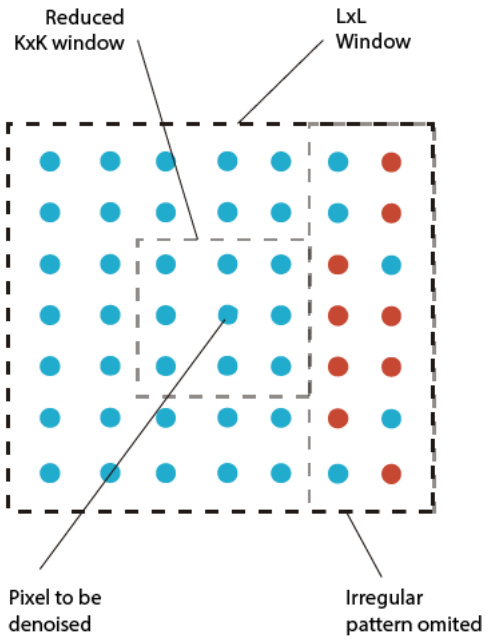


Figura 4.2: Representacion de la reduccion del tamaño de la ventana $K$.

La siguiente etapa consiste en estimar con el PCA la versión limpia $X$ del conjunto ruidoso $X_v$ y obtener el bloque central sin ruido. Una vez obtenido el bloque central sin ruido, puede aplicarse el mismo procedimiento a cada pixel hasta limpiar la imagen por completo.

# Capítulo 5

# LPG-PCA adaptativo

## 5.1. Limpieza basada en LPG-PCA

En el conjunto de muestras obtenidas por el LPG, $X_v$ es una matriz de $mxn$ cuyas n columnas son las muestras y las $m$ filas son las variables, es decir, cada componente $x_k^v$, $k = 1, 2, ..., m$, de la variable vectorial $x_v$ tiene $n$ muestras. Sea $X_k^v$ el vector fila que contiene las $n$ muestras de $x_k^v$, entonces el conjunto de datos $X_v$ puede representarse como $X_v = [(X_1^v)^T ... (X_m^v)^T]^T$, cuya contraparte sin ruido seria $X = [X_1^T ... X_m^T]^T$, donde $X_k$ es el vector fila que contiene las $n$ muestras de $x_k$.

Entonces $X_v = X + V$, donde $V = [V_1^T ... V_m^T]^T$ es el conjunto de datos correspondiente al ruido $v$ y $V_k$ es el vector muestra de $v_k$. El primer paso es centralizar el conjunto de datos $X_v$, para esto, se calcula la media $\mu_k = \frac{1}{n} \sum_{i=1}^{n} X_k^v(i)$ de $X_k^v$ y luego se centraliza $X_k^v$ restando la media a cada componente $\bar{X}_k^v = X_k^v - \mu_k^v$.
Debido a que el ruido $v_k$ tiene media cero, la matriz $V$ ya está centralizada, y $X_k$ también puede centralizarse por $\bar{X}_k = X_k - \mu_k$. Así, el conjunto de datos centralizado de $X_v$ y de $X$ son obtenidos como $\bar{X}_v = [(\bar{X}_1^T) ... (\bar{X}_m^T)]^T$ y $\bar{X} = [\bar{X}_1^T ... \bar{X}_m^T]$, entonces tenemos:

$$\bar{X}_v = \bar{X} + V. \tag{5.1}$$

Una vez centralizada la matriz de muestras, es posible calcular la matriz de covarianzas de $X$, denotado por $\Omega_{\bar{x}}$, para asi obtener la matriz $P_{\bar{x}}$ de transformación PCA. Pero debido a que el conjunto de datos $X_v$ está afectado por el ruido, $\Omega_{\bar{x}}$ no puede ser directamente computado.

$$\Omega_{\bar{x}_v} = \frac{1}{n}\bar{X}_v{\bar{X}_v}^T = \frac{1}{n}(\overline{XX}^T + \bar{X}v^T + V\bar{X}^T + VV^T). \tag{5.2}$$

Puesto que $\bar{X}_v$ y $V$ no están correlacionados, los productos de $\bar{X}V^T$ y $V\bar{X}^T$ son muy cercanos a cero, por lo que (5.2) puede simplificarse a:

$$\Omega_{\bar{x}_v} \approx \frac{1}{n}\bar{X}_v{\bar{X}_v}^T = \frac{1}{n}(\overline{XX}^T + VV^T) = \Omega_{\bar{x}} + \Omega_v. \tag{5.3}$$

Los componentes $\Omega(i,j)$ son la correlación existente entre $v_i$ y $v_j$. Debido a que los componentes $v_i$ y $v_j$ no están correlacionados para $i \neq j$, se puede saber que $\Omega_v$ es una matriz mxm diagonal con todos sus componentes en la diagonal siendo $\sigma^2$. Es decir, $\Omega_v$ puede ser escrito como $\sigma^2 I$, siendo $I$ la matriz identidad.

Ahora es posible demostrar fácilmente que la matriz $P_{\bar{x}}$ de transformación PCA asociada a $\Omega_{\bar{x}}$ es la misma matriz $P$ de transformación PCA que la asociada a $\Omega_{\bar{x}_v}$.

Debido a que $\Omega_{\bar{x}}$ es una matriz simétrica, puede ser descompuesta como:

$$\Omega_{\bar{x}} = \Phi_{\bar{x}}\Lambda_{\bar{x}}\Phi_{\bar{x}}^T. \tag{5.4}$$

Donde $\Phi_{\bar{x}}$ es la matriz ortonormal de $m$x$m$ autovectores y $\Lambda_{\bar{x}}$ es la matriz diagonal de $m$x$m$ autovalores. Debido a que $\Phi_{\bar{x}}$ es una matriz ortonormal, $\Omega_v$ puede ser escrita como:

$$\Omega_v = (\sigma^2 I)\Phi_{\bar{x}}\Phi_{\bar{x}}^T = \Phi_{\bar{x}}\Omega_v\Phi_{\bar{x}}^T. \tag{5.5}$$

Entonces:

$$\Omega_{\bar{x_v}} = \Omega_{\bar{x}} + \Omega_v = \Phi_{\bar{x}}\Lambda_{\bar{x}}\Phi_{\bar{x}}^T + \Phi_{\bar{x}}(\sigma^2)\Phi_{\bar{x}}^T = \Phi_{\bar{x}}(\Lambda_{\bar{x}} + \sigma^2 I)\Phi_{\bar{x}}^T = \Phi_{\bar{x}}\Lambda_{\bar{x_v}}\Phi_{\bar{x}}^T. \quad (5.6)$$

El hecho de que $\Lambda_{\bar{x}_v} = \Lambda_x + \sigma^2 I$ implica que $\Omega_{\bar{x}_v}$ y $\Omega_{\bar{x}}$ tienen la misma matriz $\Phi_{\bar{x}}$ de autovectores. Por lo que en la implementación, se puede computar directamente $\Phi_{\bar{x}}$ por medio de la descomposición de $\Omega_{\bar{x}_v}$ en vez de descomponer $\Omega_{\bar{x}}$. De este modo, la matriz $P_{\bar{x}}$ de transformación PCA es:

$$P_{\bar{x}} = \Phi_{\bar{x}}^T. \quad (5.7)$$

Una vez obtenida la matriz de transformación PCA, podemos aplicarla al conjunto de datos $\bar{X}_v$ para tener:

$$\bar{Y}_v = P_{\bar{x}}\bar{X}_v = P_{\bar{x}}\bar{X} + P_{\bar{x}}V = \bar{Y} + V_y. \quad (5.8)$$

Donde $\bar{Y} = P_{\bar{x}}\bar{X}$ es la matriz diagonal que representa al conjunto de datos $\bar{X}$ decorrelacionado y $V_Y = P_{\bar{x}}V$ es el conjunto de datos transformados del conjunto de datos del ruido. Debido a que $\bar{Y}$ y el ruido $\bar{V}_Y$ no están correlacionados, se puede deducir que la matriz de covarianzas de $\bar{Y}_v$ es:

$$\Omega_{\bar{Y}_v} = \frac{1}{n}\bar{Y}_v\bar{Y}_v^T = \Omega_{\bar{Y}} + \Omega_{v_y} \quad (5.9)$$

Donde $\Omega_{\bar{Y}} = \Lambda_{\bar{x}}$ es la matriz de covarianzas del conjunto de datos $\bar{Y}$ decorrelacionado y $\Omega_{v_y} = P_{\bar{x}}\Omega_v P_{\bar{x}}^T$ es la matriz de covarianzas del conjunto ruidoso $V_Y$.

La señal de mayor importancia del conjunto de datos no ruidoso $\bar{Y}$ se concentra en los componentes principales en el dominio de transformación PCA $\bar{Y}_v$, mientras la señal del ruido $V_Y$ se dispersa de forma uniforme por todo el conjunto de datos. Por lo que puede ser utilizada la técnica *linear minimum mean square-error estimation*

(LMMSE) para suprimir este ruido.

Debido a que $\bar{Y}_v$ está centralizado, el LMMSE de cada fila $\vec{Y}_k$ es obtenido al aplicar un factor en encogimiento $w$ a cada fila.

$$\vec{Y}_k = w_k.\vec{Y}_k. \tag{5.10}$$

Donde el coeficiente de encogimiento se calcula como:

$$w_k = \Omega_{\bar{y}}(k,k)/(\Omega_{\bar{y}}(k,k) + \Omega_{v_y}(k,k)). \tag{5.11}$$

En la expresión 5.11, se obtiene $\Omega_{\bar{y}}$ sustrayendo de $\Omega_{y_v}$ la matriz de covarianzas del ruido $\Omega_{v_y}$. En los componentes principales de menor "importancia", $\Omega_{\bar{y}}(k,k)$ es mucho menor que $\Omega_{v_y}(k,k)$ por lo que $w_k$ es muy cercano a cero. Por lo tanto, la mayor parte del ruido se suprimirá en $\vec{Y}_k$ al aplicar el operador linear $\vec{Y}_k = w_k.\vec{Y}_k$. En caso de que $\Omega_{y_v}(k,k) < 0$, simplemente se deja en 0, así $\Omega_y(k,k) = 0$ y el ruido en $\Omega_v^k$ es completamente removido.

Sea $\vec{Y}$ la matriz resultante de aplicar el operador linear $\vec{Y}_k = w_k.\vec{Y}_k$, Aplicando los mismos procedimientos pero en orden inverso se puede obtener el resultado de eliminar el ruido en $\bar{X}_v$. Valiéndonos de que $P_{\bar{x}}$ es una matriz ortonormal y $P_{\bar{x}}^{-1} = P_{\bar{x}}^T$

$$\tilde{X} = P_{\bar{x}}^T.\vec{Y} \tag{5.12}$$

Luego se le suman todos sus promedios $\mu_k$ al conjunto de datos limpios $\tilde{X}$.

Finalmente el bloque central $x_0$ puede ser extraído del conjunto $\tilde{X}$, y por consiguiente se puede obtener el pixel central del bloque que en un principio queríamos limpiar. Este mismo proceso debe realizarse en cada pixel, ya sea con $K = 3$ o $K = 5$, a lo largo y ancho de toda la imagen $I_v$ para realizar la limpieza completa.

## 5.2. Mejora de la eliminación de ruido en la segunda etapa

Al aplicar los procedimientos de la sección anterior, la mayor parte del ruido es removido, pero aun queda ruido residual que puede apreciarse a simple vista, por lo que es necesario ajustar el nivel del ruido y aplicar el mismo procedimiento de vuelta.

En las siguientes imágenes podemos ver la diferencia entre el resultado de la primera fase del procedimiento LPG-PCA adaptativo y el resultado de la segunda fase después de aplicar el mismo procedimiento con el nivel de ruido ajustado. Fig 5.1.a es la imagen original, Fig 5.1.b es la imagen luego de adicionar ruido blanco ($\sigma = 20$) a la imagen original, Fig 5.1.d es el resultado de la primera fase de la eliminación del ruido (PSNR = 30.52) y Fig 5.1.d es el resultado obtenido en la segunda fase del algoritmo (PSNR = 30.70). El PSNR (del ingles *Peak Signal-to-Noise Ratio*) es la relación existente entre la máxima energía posible de la señal y el ruido que la afecta, siendo su unidad de medida el decibelio. A pesar de que el PSNR aumenta considerablemente luego de la primera etapa de eliminación de ruido, es posible visualizar ruido residual a simple vista. Para la segunda etapa el PSNR aumentó de 30.34 a 30.97db, aunque no haya mucha diferencia en el PSNR, se puede apreciar visualmente la mejora.

El ruido residual está causado por dos motivos: (a) El primer motivo es el fuerte ruido en el conjunto de datos originales $\bar{X}_v$. Este fuerte ruido provoca una estimación no muy buena de la matriz de covarianzas $\Omega_{\bar{x}_v}$, lo que conduce a un sesgo de estimación de la matriz de transformación PCA, deteriorando la calidad de la eliminación de ruido. (b) El segundo motivo es que el fuerte ruido en la imagen original produce una mala precisión en el LPG, lo que conlleva a un sesgo en la estimación de la matriz de covarianzas $\Omega_{\bar{x}_v}$. Debido a estas causas, es necesario realizar de vuelta la

Figura 5.1: (a) Imagen original, (b) Imagen ruidosa (PSNR=22.1dB), (c) Imagen resultante de la primera etapa de la eliminación de ruido (PSNR=30.34dB), (d) Imagen resultante de la segunda etapa de la eliminación de ruido (PSNR=30.97dB).

eliminación de ruido para obtener mejores resultados. Para esto, se debe actualizar el nivel de ruido de modo a que el LPG tenga una mejor precisión para que el PCA pueda realizar una mejor estimación de $\Omega_{\bar{x}_v}$ , en consecuencia, una mejor estimación de la matriz de transformación PCA y poder obtener mejores resultados visuales.

Para conseguir el nivel adecuado al actualizar el valor del ruido comparamos la imagen ruidosa $I_v$ con la imagen ruidosa $\hat{I}$, resultado de la eliminación de ruido en la primera etapa. Podemos escribir $\hat{I} = I + v_s$, siendo $v_s$ el ruido residual. Entonces el objetivo es determinar qué valor posee $v_s$, denotado por $\sigma_s = \sqrt{E[v_s^2]}$, para utilizarlo en la segunda etapa del LPG-PCA. Al comparar las diferencias entre $\hat{I}$ y $I_v$ tenemos:

$$\tilde{I} = I_v - \hat{I}. \tag{5.13}$$

Que a su vez es:

$$E[\tilde{I}^2] = E[v^2] + E[v_s^2] - 2E[v.v_s] = \sigma^2 + \sigma_s^2 - 2E[v.v_s]. \tag{5.14}$$

$v_s$ puede verse como una versión suave del ruido $v$, que contiene los componentes de baja frecuencia de $v$. Así que si $\tilde{v} = v - v_s$, podemos ver que esta diferencia contiene los componentes de alta frecuencia de $v$, también tenemos que $E[v.v_s] = E[\tilde{v}.v_s] + E[v_s^2]$. Por cuestiones de simplicidad [LS10] sugiere remover $E[\tilde{v}.v_s]$ de $E[v.v_s]$, y nos queda $E[v.v_s] \approx E[v_s^2] = \sigma_s^2$.
Por lo que tenemos:

$$\sigma_s^2 \approx \sigma^2 - E[\tilde{I}^2]. \tag{5.15}$$

Aunque en la práctica no solo el ruido residual es incluido, también la estimación del error en la imagen limpia $I$. Por lo que nos queda que:

$$\sigma_s = c_s.\sqrt{\sigma^2 - E[\tilde{I}^2]}. \tag{5.16}$$

Donde $c_s$ es una constante menor a 1. Basándonos en las pruebas realizadas asignamos $c_s = 0{,}27$ ya que es el valor que otorga mejores resultados. También es actualizado el *threshold* para la selección de muestras ya que con la segunda etapa de eliminación de ruido se puede ser más estricto en cuando a la semejanza de las muestras con el bloque central debido a que se cuenta con un menor grado de error cuadrático.

## 5.3. Imágenes a color

Las imágenes a color están compuestas por los tres canales primarios, que son el verde, el rojo y el azul (RGB). Para llevar a cabo el método LPG-PCA adaptativo en imágenes a color, existen dos enfoques.

El primer enfoque es separar cada uno de los canales y realizar el método propuesto a cada uno de ellos. El inconveniente que presenta este método es que ignora la correlación espectral existente entre cada uno de los canales. El segundo enfoque no requiere separar cada canal, mas bien, se modela cada pixel como un cubo de $K$x$K$x3, en donde cada matriz $K$x$K$ representa el modelado del pixel de cada canal correspondiente. Así, la eliminación de ruido se lleva a cabo igual que en imágenes de escala de grises, pero en variables vectoriales de $3K^2$, por lo que el LPG selecciona las muestras dentro de una ventana $L$x$L$x3. Aunque este enfoque no ignora la correlación espectral existente entre cada canal, verde, rojo y azul, se requiere un esfuerzo de cómputo mucho mayor que el primer enfoque debido a la dimensionalidad de las muestras. También, este aumento de dimensionalidad implica una mayor cantidad de muestras necesarias para llevar a cabo la eliminación de ruido, por lo que el cálculo de la matriz de transformación PCA podría no ser bien estimada, llevando a una reducción de la calidad en la eliminación de ruido.

Debido a estos inconvenientes y a la simplicidad del primer enfoque, en este trabajo optamos por separar cada canal de las imágenes a color para llevar a cabo la eliminación de ruido como si se tratara de una imagen a escala de grises, y al final unir los resultados de cada canal para formar de vuelta la imagen a color.

# Capítulo 6

# Resultados experimentales

Para imágenes en escala de grises, de forma a probar el método, el primer paso es agregarle el ruido blanco, se cargan las matrices que contendrán las variables vectoriales correspondientes a cada muestra para $K = 5$ y $K = 3$, luego se divide la imagen en columnas que serán procesadas en paralelo.

Cada proceso aplicara el LPG-PCA adaptativo pixel a pixel a lo largo de la imagen y rango de anchura asignado. Entonces cada pixel que ha sido tratado, se carga en una nueva matriz, esto se hace para que la limpieza sea uniforme ya que se asume que el ruido tiene media cero y desviación estándar $\sigma$ a lo largo de toda la imagen ruidosa.

Por defecto, el LPG toma bloques con $K = 5$ y en caso de no obtener suficientes muestras se reduce el tamaño de la ventana a $K = 3$, entonces se pasa la matriz de muestras al PCA para que lleve a cabo la eliminación de ruido.

El resultado del PCA es todo el bloque central $K$x$K$ limpio, no solo el pixel central, el cual sera cargado en la nueva imagen que se está formando, de modo a que cada pixel es tratado entre 9 y 25 pasadas dependiendo del tamaño de la ventana $K$. Por lo que el resultado final es el promedio de todas las limpiezas realizadas por pixel. Este método también es utilizado en [LS10].

Una vez recorrida toda la imagen, se juntan los procesos para actualizar el nivel de ruido, así como también se actualiza el *threshold* para la selección de muestras, que basándonos en las pruebas realizadas, con $T = 5$ se obtienen mejores resultados. Entonces el método LPG-PCA adaptativo se vuelve a llevar a cabo.

Para imágenes a color el procedimiento es prácticamente el mismo, solo que primero se separan los canales, rojo, verde y azul y luego se divide en tres procesos dándole a cada canal el mismo tratamiento que a una imagen de escala de grises.

Aunque visualmente pueda apreciarse la mejora de la calidad en las imágenes luego de la segunda etapa del algoritmo LPG-PCA adaptativo propuesto, es necesario tener alguna medida de calidad. La métrica de calidad más ampliamente utilizada es el MSE, calculando el promedio de la intensidad cuadrada de los pixeles distorsionados con la imagen de referencia, junto con la relación de Pico de Señal a Ruido (PSNR). Es un método bastante sencillo de implementar y tiene un claro significado, cuanto mayor el PSNR, menor el error medio cuadrático.

El verdadero objetivo de cualquier método de eliminación de ruido es mejorar la calidad visual y la fidelidad de una imagen ruidosa, pero este método no es muy eficiente para expresar la calidad percibida por el ojo humano. Por lo tanto, también utilizamos el *Structural similarity* (SSIM) como medida de calidad, ya que este método es utilizado para predecir la calidad percibida, cuanto mayor SSIM, mayor calidad percibida por el ojo humano.

Para medir la calidad de las imágenes a color, se debe tener en cuenta cada uno de los canales RGB, lo que resulta un poco ambiguo, es por esto que para tener una única medida se pasan las imágenes a escala de grises.

Figura 6.1: Imágenes utilizadas en las pruebas: Cameraman, Lenna, Monarch, House, Einstein, Peppers, Barbara (color) y Parrot (color)

Tabla 6.1: PSNR y SSIM de las imágenes luego de la eliminación de ruido con el método “Eliminación de ruido blanco con LPG-PCA adaptativo en dos etapas"

| Imágenes | Cameraman | Lenna | House | Monarch |
|---|---|---|---|---|
| Primera Etapa | | | | |
| $\sigma = 10$ | 34.52-0.58 | 32.92-0.81 | 35.56-0.62 | 33.82-0.83 |
| $\sigma = 20$ | 30.34-0.45 | 29.10-0.68 | 31.86-0.48 | 29.75-0.72 |
| $\sigma = 30$ | 27.86-0.39 | 26.78-0.58 | 29.29-0.40 | 27.41-0.64 |
| Segunda Etapa | | | | |
| $\sigma = 10$ | 34.83-0.58 | 32.98-0.81 | 35.84-0.62 | 34.00-0.84 |
| $\sigma = 20$ | 30.97-0.48 | 29.33-0.68 | 32.82-0.49 | 30.23-0.75 |
| $\sigma = 30$ | 28.67-0.42 | 27.16-0.58 | 30.87-0.41 | 28.11-0.68 |
| | | | | |
| Imágenes | Peppers | Einstein | Barbara | Parrot |
| Primera Etapa | | | | |
| $\sigma = 10$ | 34.37-0.76 | 33.89-0.75 | 36.34-0.87 | 36.04-0.74 |
| $\sigma = 20$ | 30.66-0.63 | 30.30-0.58 | 31.96-0.75 | 32.05-0.66 |
| $\sigma = 30$ | 28.19-0.53 | 28.06-0.46 | 29.45-0.67 | 29.53-0.58 |
| Segunda Etapa | | | | |
| $\sigma = 10$ | 34.58-0.76 | 34.01-0.75 | 36.40-0.87 | 36.07-0.76 |
| $\sigma = 20$ | 31.21-0.65 | 30.76-0.58 | 32.15-0.76 | 32.31-0.67 |
| $\sigma = 30$ | 29.05-0.58 | 28.95-0.48 | 29.88-0.69 | 30.06-0.61 |

El primer valor corresponde al PSNR y el siguiente después del guion medio al SSIM.

Tabla 6.2: Comparación de los métodos LPG-PCA adaptativo y LPG-PCA con tamaño fijo de ventana K

| Imágenes | Camera-man | Lenna | Peppers | Monarch | Parrot | House |
|---|---|---|---|---|---|---|
| LPG-PCA adaptativo | | | | | | |
| $\sigma = 10$ | 34.03-0.58 | 32.22-0.81 | 34.58-0.76 | 34.00-0.84 | 35.73-0.76 | 35.84-0.62 |
| $\sigma = 20$ | 30.14-0.48 | 28.57-0.68 | 30.44-0.65 | 29.31-0.75 | 32.30-0.67 | 32.05-0.49 |
| $\sigma = 30$ | 27.90-0.42 | 26.36-0.58 | 29.05-0.58 | 27.35-0.68 | 30.05-0.61 | 30.11-0.41 |
| LPG-PCA | | | | | | |
| $\sigma = 10$ | 33.84-0.58 | 29.88-0.75 | 34.12-0.76 | 33.82-0.86 | 34.74-0.75 | 35.64-0.62 |
| $\sigma = 20$ | 30.09-0.47 | 27.42-0.66 | 30.36-0.65 | 29.78-0.79 | 31.18-0.65 | 31.85-0.51 |
| $\sigma = 30$ | 27.71-0.42 | 25.91-0.58 | 28.74-0.61 | 27.45-0.73 | 29.12-0.59 | 30.68-0.43 |

El primer valor corresponde al PSNR y el siguiente después del guion medio al SSIM.

En la tabla 6.2 se puede apreciar que para ciertas imágenes y ciertos niveles de ruido, con un método se obtienen mejores resultados que con el otro. Esto se debe a que el método LPG-PCA adaptativo preserva mejor los bordes aunque estos sean irregulares, ya que se puede obtener una mayor cantidad de muestras al reducir el tamaño de la ventana $K$. Sin embargo, dependiendo de la complejidad de la estructura de la imagen, puede que aunque se reduzca el tamaño de la ventana $K$ no se obtenga una cantidad considerable de muestras, por lo que resulta conveniente en este caso tomar las muestras más similares aunque superen el *threshold* definido en el *Block Matching Method*, como lo hace el método LPG-PCA con ventana $K$ fija. El problema de este método es que las muestras tomadas que superan el *threshold* pueden provocar que los resultados sean borrosos, dificultando la preservación de los bordes o ciertos detalles que podrían mantenerse utilizando el método propuesto.

Figura 6.2: (a) Imagen original, (b) Imagen ruidosa (PSNR=22.1dB), (c) Imagen resultante de la primera etapa de la eliminación de ruido (PSNR=29.10dB), (d) Imagen resultante de la segunda etapa de la eliminación de ruido (PSNR=29.33dB).

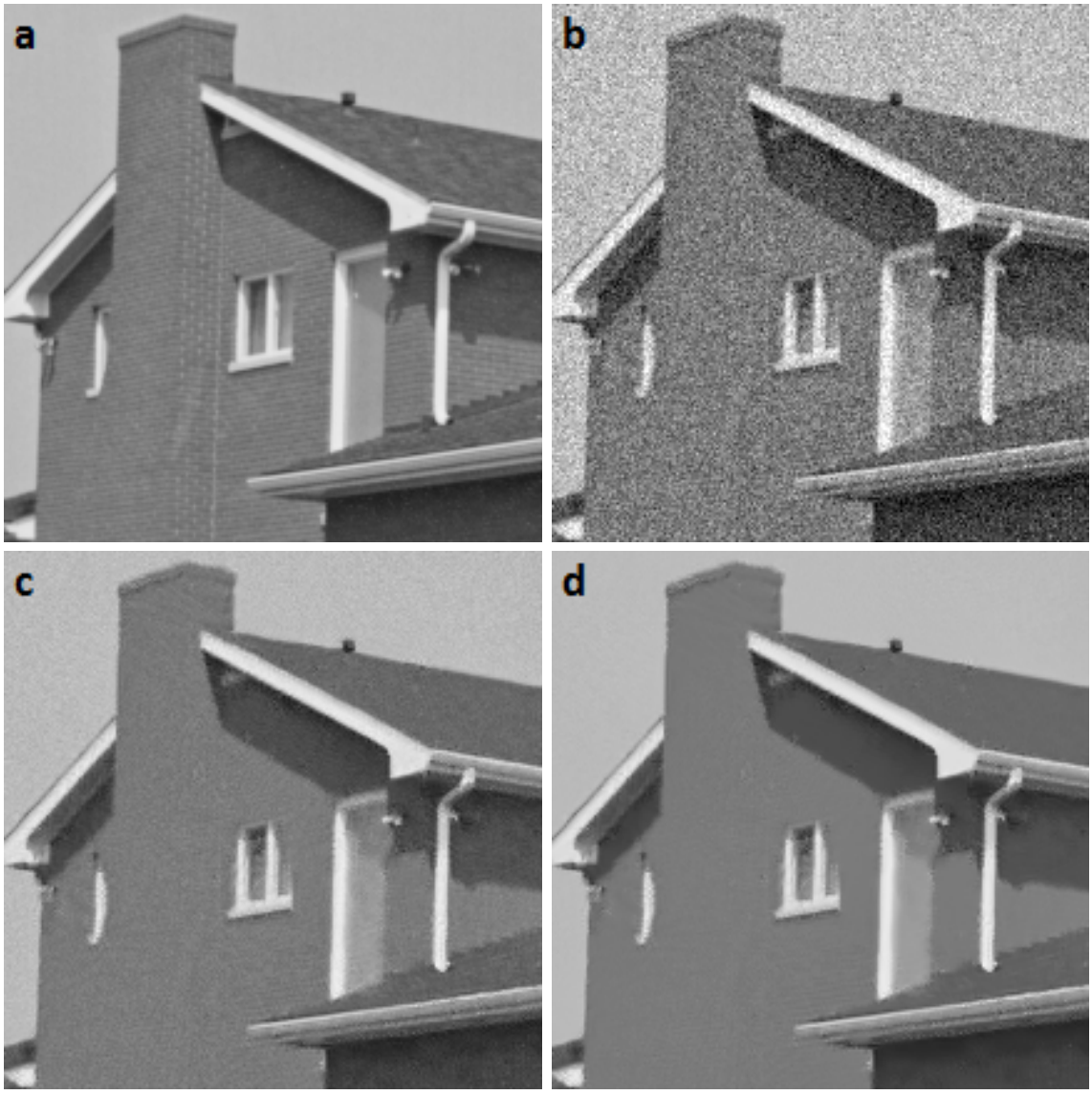


Figura 6.3: (a) Imagen original, (b) Imagen ruidosa (PSNR=22.1dB), (c) Imagen resultante de la primera etapa de la eliminación de ruido (PSNR=31.86dB), (d) Imagen resultante de la segunda etapa de la eliminación de ruido (PSNR=32.82dB).

Figura 6.4: (a) Imagen original, (b) Imagen ruidosa (PSNR=22.1dB), (c) Imagen resultante de la primera etapa de la eliminación de ruido (PSNR=29.75dB), (d) Imagen resultante de la segunda etapa de la eliminación de ruido (PSNR=30.23dB).

Figura 6.5: (a) Imagen original, (b) Imagen ruidosa (PSNR=22.1dB), (c) Imagen resultante de la primera etapa de la eliminación de ruido (PSNR=30.66dB), (d) Imagen resultante de la segunda etapa de la eliminación de ruido (PSNR=31.21dB).

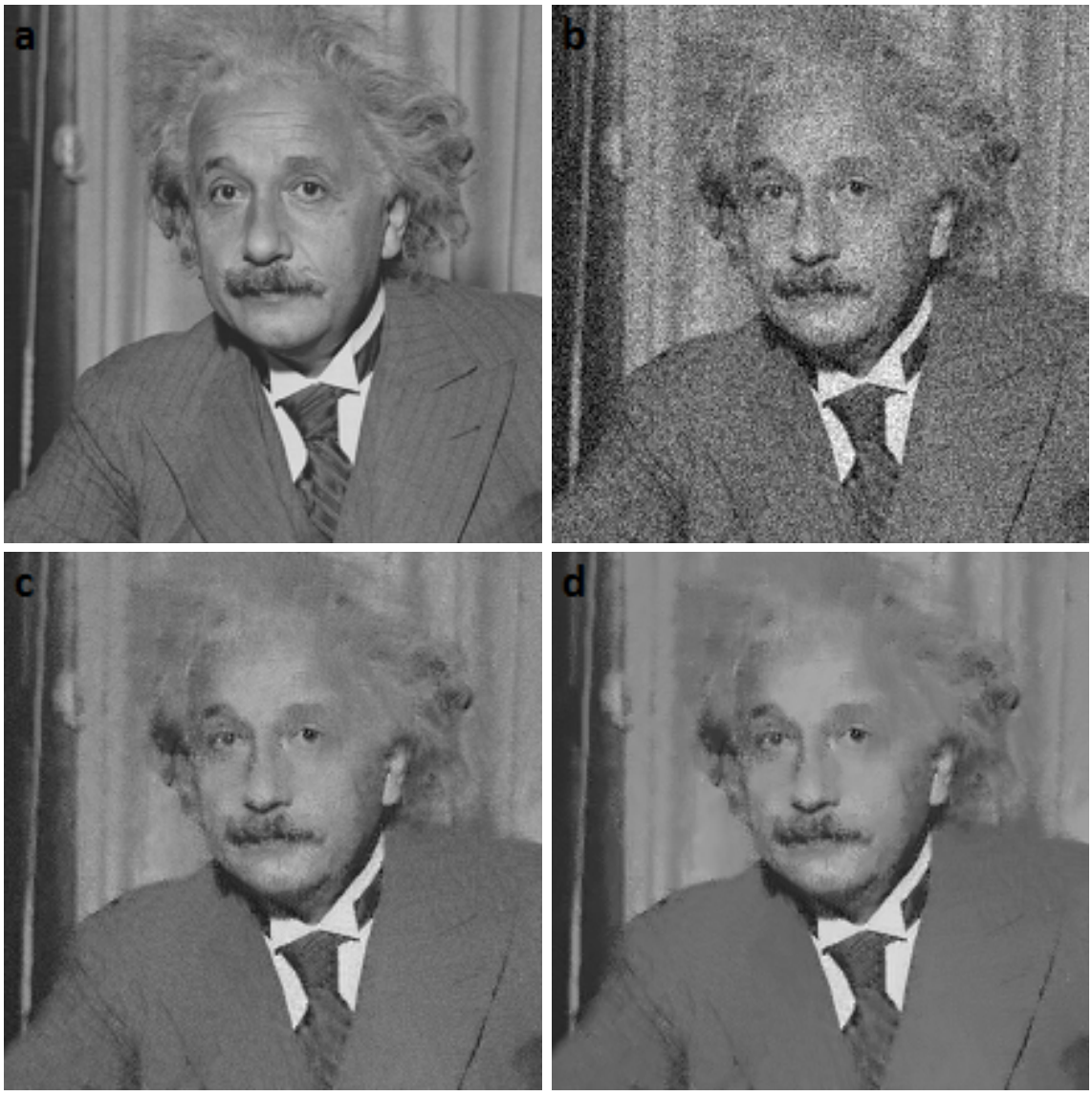


Figura 6.6: (a) Imagen original, (b) Imagen ruidosa (PSNR=22.1dB), (c) Imagen resultante de la primera etapa de la eliminación de ruido (PSNR=30.30dB), (d) Imagen resultante de la segunda etapa de la eliminación de ruido (PSNR=30.76dB).

Figura 6.7: (a) Imagen original, (b) Imagen ruidosa (PSNR=22.1dB), (c) Imagen resultante de la primera etapa de la eliminación de ruido (PSNR=31.96dB), (d) Imagen resultante de la segunda etapa de la eliminación de ruido (PSNR=32.15dB).

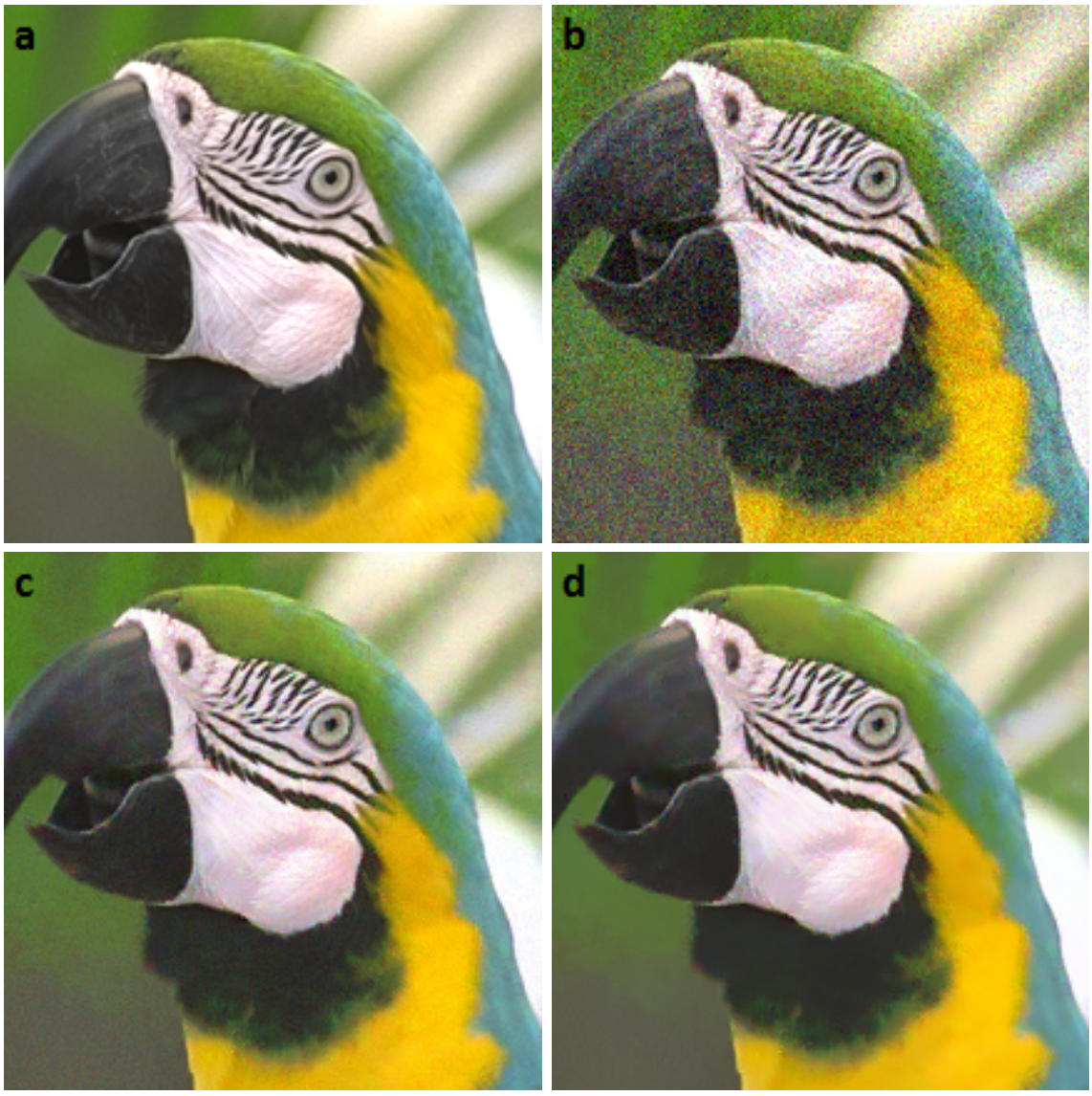


Figura 6.8: (a) Imagen original, (b) Imagen ruidosa (PSNR=22.1dB), (c) Imagen resultante de la primera etapa de la eliminación de ruido (PSNR=32.05dB), (d) Imagen resultante de la segunda etapa de la eliminación de ruido (PSNR=32.31dB).

# Capítulo 7

# Conclusión

Este método propone una técnica adaptativa de reducción del ruido blanco en imágenes, usando LPG-PCA con el tamaño de muestras adaptativo, en dos etapas.

La técnica PCA se utilizó para de-correlacionar la señal del ruido, estimando la matriz ortonormal de transformación entrenada por el algoritmo LPG y utilizarlo como nueva base para representar al conjunto de datos. Debido a que en la imagen se encuentran distintos patrones en su estructura, la selección de muestras es necesaria.

Basandonos en las pruebas realizadas, $K = 5$ y $L = 41$ son los criterios que arrojan mejores resultados. En áreas donde solo se obtiene una pequeña cantidad de muestras, se demuestra que vale la pena variar a $K = 3$ para que el LPG pueda obtener un mayor número de muestras. Esto mejora la calidad de eliminación del ruido después de reducir el coeficiente en la matriz de transformación PCA. Por lo que la ventaja de cambiar dinámicamente el tamaño de K se ha demostrado.

En la primera etapa, la mayor parte del ruido se elimina; en la segunda se actualiza el nivel de ruido, mejorando la precisión del LPG y permitiendo una mejor estimación en la matriz de covarianza y por consiguiente en la matriz de transformación PCA, permitiendo obtener mejores resultados visuales.

# Apéndice A

# Abstract presentado en la Conference of Computational Interdisciplinary Science CCIS 2016

Noise deteriorates image quality, and white noise in particular is present in almost any image process. In this paper, we focus on removing white noise through Principal Component Analysis (PCA) using several stages and step scales on the image.

To this end, for a specific pixel to be cleaned, we work with two windows $K$ and $L$, with $K$ centered and included in $L$. Window $K$ helps to preserve the local structure of the image, while window $L$ is a training region for the PCA. Several alternatives of $K$ into $L$ are considered.

We choose the samples that are adjacent to the pixel to be cleaned, and the block matching method is implemented. This ensures that the samples preserve local image structure for computing the PCA transformation matrix, by guaranteeing preservation of edges after coefficient shrinkage in the PCA transformation matrix. Hence, the PCA is extracted from the covariance matrix of the sample matrix observing that since the image is noise-corrupted, it cannot be directly computed. At each pixel, various window configurations may be implemented based on the local complexity of the image structure, specially in regions where there is an irregular edge.

With the proposed method, we achieve improvements in the quality and extent of de-noising in complex images that contain irregular edges. Specifically, by means of

a reduction in the size of window $K$, we can obtain a greater number of samples and thus obtain better results.

**Keywords**: Denoising, Principal component analysis (PCA), Multiscale.

# Apéndice B

# Codigo del software Eliminación de ruido blanco con LPG-PCA adaptativo en dos etapas

## Software desarrollado en Python 2.7 con el plugin PyDev de Eclipse

## B.1. Algoritmo de eliminación de ruido blanco en imagenes de escala de grises (Gray-Image-Denoising)

```
import numpy as np
from scipy import misc
import lpg
from psnr import getNewNoise,getPsnr
from multiprocessing import Process
from addNoise import addNoise
from recorrer import denoisingImage


#Funcion que une los procesos de las limpiezas para producir una unica imagen
def juntarProcesos(oIm,anchoP,c0,c1,c2,c3):
```

```
    denoisedImage = np.zeros(oIm.shape)
    [alto,ancho] = denoisedImage.shape
    for m in range(0,alto):
        for n in range(0,ancho):
            if n < anchoP+2:
                denoisedImage[m][n] = c0[m][n]
            elif n < anchoP*2+2 and n > anchoP:
                denoisedImage[m][n] = c1[m][n]
            elif n < anchoP*3+2 and n > anchoP*2:
                denoisedImage[m][n] = c2[m][n]
            else:
                denoisedImage[m][n] = c3[m][n]
    return denoisedImage


#Funcion que separa en procesos la eliminación de ruido
def procesarParalelo(anchoP, denoisedIm,ssm,ssm3,K,KK,L,cotaK,cotaL,sigma,iniAnch
                                                    anchoP,cantSamples,cond,indice):
    jobs = []
    process = 4
    for n in range(process):
        if n == 0:
            iniAncho = 0
        elif n == 1:
            iniAncho = anchoP
        elif n == 2:
            iniAncho = anchoP*2
        else:
            iniAncho = anchoP*3
```

```
        p = Process(target=denoisingImage, args=(denoisedIm,ssm,ssm3,K,KK,L,cotaK
                                                 anchoP,cantSamples,cond,n+indice
        jobs.append(p)

    for n in range(process):
        #Se inicia la limpieza
        jobs[n].start()
    for n in range(process):
        jobs[n].join()

if __name__ == '__main__':
    # Se inicializan las variables
    K = 5
    L = 41
    #Cantidad maxima de muestras que puede obtenerse
    cantSamples = (L-K+1)**2
    #Desviacion tipica del ruido
    sig = 10
    sigma = sig**2
    KK = K**2
    #Referencia utilizada para no sobrepasar los limites de las ventanas
    cotaK = (K-1)/2
    cotaL = (L-1)/2
    #Factor de ajuste
    fa = 0.18

    #Se lee dos veces porque si no, se sobreescribe la imagen
    oIm = misc.imread('Images/Parrot.tif', flatten=1)
```

```
orIm = misc.imread('Images/Parrot.tif', flatten=1)
misc.imshow(oIm)
[alto,ancho] = oIm.shape
nIm = addNoise(oIm,sig)
misc.imshow(nIm)
print "Ruido en el input",sigma
#Threshold
T = 25
#Se define la condicion para el lpg
cond = T + 2*sigma
print "Condicion",cond
misc.toimage(nIm, cmin=0, cmax=255).save('noisedParrot.tif')
denoisedIm = np.zeros(oIm.shape)
#Se cargan las muestras para K=5 y K=3
ssm = lpg.cargarTotalSamples(nIm, cotaK, K, alto, ancho)
ssm3 = lpg.cargarTotalSamples(nIm, cotaK-1, K-2, alto, ancho)
[alto,ancho] = ssm.shape
#Ancho de cada bloque de limpieza
anchoP = ancho/4

# Se divide en 4 procesos la limpieza de la imagen
procesarParalelo(anchoP,denoisedIm,ssm,ssm3,K,KK,L,cotaK,cotaL,sigma,
                                                anchoP,cantSamples,cond,0)

#Se leen las imagenes de los procesos para la construccion de la imagen
c0 = misc.imread('denoisedP0.tif',flatten=1)
c1 = misc.imread('denoisedP1.tif',flatten=1)
c2 = misc.imread('denoisedP2.tif',flatten=1)
```

```
c3 = misc.imread('denoisedP3.tif',flatten=1)


denoisedImage=juntarProcesos(oIm,anchoP,c0,c1,c2,c3)


print "Mejora en la primera pasada",getPsnr(denoisedImage,orIm),"db"
misc.imsave('parallel.tif',denoisedImage[20:alto-20,20:ancho-20])
misc.imsave('denoisedParallel.tif',denoisedImage)


#SEGUNDA FASE
#Se lee la imagen de salida de la primera pasada para limpiar
dnIm = misc.imread('denoisedParallel.tif', flatten=1)
denoisedIm = np.zeros(oIm.shape)
#Se calcula el nuevo nivel de ruido
sigma = getNewNoise(dnIm,nIm,sigma, fa)
print "Nuevo valor de ruido",sigma
#Threshold
T = 5
#Se define la condicion para el lpg
cond = T + 2 * sigma
print "Nueva condicion",cond
[alto,ancho] = dnIm.shape
#Se cargan las muestras para K=5 y K=3
ssm = lpg.cargarTotalSamples(dnIm, cotaK, K, alto, ancho)
ssm3 = lpg.cargarTotalSamples(dnIm, cotaK-1, K-2, alto, ancho)
[alto,ancho] = ssm.shape
#Ancho de cada bloque de limpieza
anchoP = ancho/4
ini = 0
```

```
    # Se divide en 4 procesos la limpieza de la imagen
    procesarParalelo(anchoP,denoisedIm,ssm,ssm3,K,KK,L,cotaK,cotaL,sigma,
                                                    anchoP,cantSamples,cond,4)


    #Se leen las imagenes de los procesos para la onstruccion de la imagen
    c0 = misc.imread('denoisedP4.tif',flatten=1)
    c1 = misc.imread('denoisedP5.tif',flatten=1)
    c2 = misc.imread('denoisedP6.tif',flatten=1)
    c3 = misc.imread('denoisedP7.tif',flatten=1)


    denoisedImage=juntarProcesos(oIm,anchoP,c0,c1,c2,c3)


    misc.imsave('finalDenoised.tif',denoisedImage)
    misc.imsave('denoised.tif',denoisedImage[20:alto-20,20:ancho-20])
    print "Mejora final",getPsnr(denoisedImage,orIm),"db"
    misc.imshow(denoisedImage)
```

## B.2. Algoritmo de eliminación de ruido blanco (LPG-PCA adaptativo)

```
import numpy as np
from scipy import misc
import time
import lpg
from pca import pca
```

```
#Se recorre la imagen limpiando pixel por pixel
def denoisingImage(denoisedIm,ssm,ssm3,K,KK,L,cotaK,cotaL,sig,iniAncho,anchoP,can
    MINSAMPLES = 4
    tic = time.clock()
    name = "denoisedP"+str(n)+".tif"
    [alto,ancho] = ssm.shape
    ref = np.zeros(shape=denoisedIm.shape)
    anchoP = iniAncho+anchoP

    print "alto: ",alto," anchoP: ",anchoP," sig: ",sig," ssm: ",ssm.shape

    for x in range(0,alto):
        for y in range(iniAncho,anchoP):
            #Se cargan las muestras correspondientes al pixel para K = 5
            [sm,nroMuestra,nm] = lpg.cargarSamples1(ssm,K,KK,L,x,y,cotaK,cotaL,al

            if nm < MINSAMPLES :
                #Se cargan las muestras correspondientes al pixel para K = 3
                [sm3,nroMuestra,nm] = lpg.cargarSamples1(ssm3,K-2,9,L-2,x+1,y+1,c
                #Se realiza la limpieza con el pca y se obtiene la ventana limpia
                denoisedIm[x+2-1:x+2+2,y+2-1:y+2+2] += pca(sm3, nroMuestra, sig)
                #Se suma la cantidad de veces que se limpia cada pixel
                ref[x+2-1:x+2+2,y+2-1:y+2+2] += 1
            else:
                #Se realiza la limpieza con el pca y se obtiene la ventana limpia
                denoisedIm[x+2-2:x+2+3,y+2-2:y+2+3] += pca(sm, nroMuestra, sig)
                #Se suma la cantidad de veces que se limpia cada pixel
                ref[x+2-2:x+2+3,y+2-2:y+2+3] += 1
```

```
    #Se divide cada pixel por la cantidad de limpiezas para obtener el promedio
    for a in range(0,alto):
        for b in range(iniAncho,anchoP):
            denoisedIm[a+2][b+2] = denoisedIm[a+2][b+2]/ref[a+2][b+2]


    toc = time.clock()
    print (toc-tic)/60,"minutos",name
    #Se guarda la porcion de la imagen limpia correspondiente a este proceso
    misc.toimage(denoisedIm, cmin=0, cmax=255).save(name)
```

## B.3. Analisis de componentes principales (pca)

```
from __future__ import division
import numpy as np
import math


def pca(sm, nroMuestra, sigma):
    sm = sm.astype('float')
    f, c = sm.shape
    fil = int(math.sqrt(f))
    if c == 1:#Si hay una sola muestra no se puede realizar la limpieza
        return sm[0:f,nroMuestra].reshape(fil,fil)
    prom = sm.mean(axis=1)#Promedio de filas
    sm = (sm.transpose() - prom).transpose()#Centralizacion
    cov = 1/c*sm.dot(sm.transpose())#Matriz de covarianzas
    v = np.linalg.svd(cov)[2]#Descomposicion de valores singulares
    P = v.transpose()#Matriz de transformacion ortonormal
```

```
    Yv = P.dot(sm)#Dominio del pca
    f,c = Yv.shape
    covYv = np.diag(np.diag(1/c*Yv.dot(Yv.transpose())))#Covarianzas
    en el dominio del PCA
    covV = np.identity(f)*sigma#Covarianza del ruido
    covVy=covV#Covarianza del ruido en el dominio pca
    Yk  = np.zeros((f,c))
    covY=covYv-covVy#Se sustrae la matriz de covarianzas del ruido
    for m in range(0, f):#Se setean a 0 los valores negativos
        if(covY[m][m]<0):
            covY[m][m] = 0
    for m in range(0, f):#Se minimiza la varianza
        w = covY[m][m]/(covY[m][m]+covVy[m][m])
        #print covVy[m][n]
        for n in range(0, c):
            Yk[m][n] = w*Yv[m][n]
    sm = P.transpose().dot(Yk)#Proceso de recuperacion de la imagen
    sm = (sm.transpose() + prom).transpose()#Se suman los promedios
    sustraidos
    return sm[0:f,nroMuestra].reshape(fil,fil)
```

## B.4. Agrupacion Local de Pixeles (lpg)

```
import numpy as np
from qs import quickSort


#Se carga la matriz de vectores que contiene todas las muestras de
```

```
la imagen
def cargarTotalSamples(im,cotaK,K,alto,ancho):
    #Nuevas dimensiones
    nAlto = alto - 2*cotaK
    nAncho = ancho - 2*cotaK
    sampleMatrix = np.zeros(shape=(nAlto,nAncho),dtype=object)
    sampleVector = np.zeros((K,K))
    [alto,ancho] = sampleMatrix.shape
    #Se fijan las cotas finales a partir de las nuevas dimensiones
    finAncho = ancho+cotaK
    finAlto = alto+cotaK
    for m in range(cotaK,finAlto):
        for n in range(cotaK,finAncho):
            #Se convierte a un array esta region de la imagen
            sampleVector = np.array(im[m-cotaK:m+cotaK+1,n-cotaK:
            n+cotaK+1]).reshape(K**2)
            sampleMatrix[m-cotaK][n-cotaK] = sampleVector
    return sampleMatrix

def cargarSamples(sm,K,KK,L,x,y,cotaK,cotaL,alto,ancho,cantSamples,T):
    #Se atiende que no se sobrepasen los limites de la imagen
    infx = max(x-cotaL+cotaK,0)
    supx = min(alto,x+cotaL-cotaK+1)
    infy = max(y-cotaL+cotaK,0)
    supy = min(ancho,y+cotaL-cotaK+1)
    #Se crea la matriz de muestras en base a la cantidad maxima de
    muestras posibles mas una fila para el emc
    samples = np.zeros(shape=(KK+1,cantSamples))
```

```
    col = 0


    for m in range(infx,supx):
        for n in range(infy,supy):
            #Se calcula la distancia cuadratica
            emc = np.sum((sm[x][y]-sm[m][n])**2)
            emc = emc/KK
            #Se verifica si se cumple la condicion para agregar a la
            matriz de muestras
            if emc < T:
                #Se carga por columna
                samples[0:KK,col] = sm[m][n]
                #Se carga su emc correspondiente
                samples[KK,col] = emc
                col += 1
    samples = samples[:,0:col]
    #Se ordenan las muestras en base al emc
    quickSort(samples)
    #Cantidad de muestras seleccionadas
    c = 10*KK
    if col > c:
        return (samples[0:KK,0:c],0,c)
    #Se eliminan las columnas que contienen 0s
    return (samples[0:KK,0:col],0,col)
```

## B.5. Imagenes a color (colorDenoising)

```
import numpy as np
```

```
from scipy import misc

import lpg

from psnr import getNewNoise,getPsnr

from multiprocessing import Process

import cv2

from codigo1 import denoisingImage

from addNoise import colorNoise


#Funcion que junta cada canal para formar la imagen a color

def juntarCanales(im,c0,c1,c2,n=''):

    denoisedImage = np.zeros(im.shape)

    denoisedImage[:,:,0] = c0

    denoisedImage[:,:,1] = c1

    denoisedImage[:,:,2] = c2

    misc.imsave("denoisedColor"+str(n)+".tif",denoisedImage)

    misc.imshow(denoisedImage)


#Funcion que inicializa las variables para la eliminacion de ruido de un canal de

def procesar(colorChannel,K,KK,L,cantSamples,sigma,cotaK,cotaL,alto,ancho,cond,n)

    #Nombre del colorChannel para guardar en el disco duro

    nName = "noisedChannel"+str(n)+".tif"

    misc.toimage(colorChannel, cmin=0, cmax=255).save(nName)


    denoisedIm = np.zeros(colorChannel.shape)


    #Se cargan las muestras para K=5 y K=3

    ssm = lpg.cargarTotalSamples(colorChannel, cotaK, K, alto, ancho)

    ssm3 = lpg.cargarTotalSamples(colorChannel, cotaK-1, K-2, alto, ancho)
```

```
[alto,ancho] = ssm.shape
#Recorre la imagen para la eliminacion de ruido
denoisingImage(denoisedIm,ssm,ssm3,K,KK,L,cotaK,cotaL,sigma,0,ancho,cantSampl


dnName = "denoisedP"+str(n)+".tif"
denoisedIm = misc.imread(dnName, flatten = 1)
psnr = getPsnr(denoisedIm,colorChannel)
print "Mejora en la primera pasada de colorChannel ",n," ",psnr," db"


#SEGUNDA FASE
#Se lee la imagen de salida de la primera pasada para limpiar
dnIm = misc.imread(dnName, flatten=1)
denoisedIm = np.zeros(dnIm.shape)
#Se calcula el nuevo nivel de ruido
sigma = getNewNoise(dnIm,colorChannel,sigma)
#Threshold
T = 5
#Se define la condicion para el lpg
cond = T + 2 * sigma
[alto,ancho] = dnIm.shape
#Se cargan las muestras para K=5 y K=3
ssm = lpg.cargarTotalSamples(dnIm, cotaK, K, alto, ancho)
ssm3 = lpg.cargarTotalSamples(dnIm, cotaK-1, K-2, alto, ancho)
[alto,ancho] = ssm.shape
#Recorre la imagen para la eliminacion de ruido
denoisingImage(denoisedIm,ssm,ssm3,K,KK,L,cotaK,cotaL,sigma,0,ancho,cantSampl
dnName = "denoisedP"+str(n+3)+".tif"
denoisedIm = misc.imread(dnName, flatten = 1)
```

```
    psnr = getPsnr(denoisedIm,colorChannel)
    print "Mejora en la segunda pasada de colorChannel ",n+3," ",psnr,"db"


if __name__ == '__main__':
    # Se inicializan las variables
    K = 5
    L = 41
    #Cantidad maxima de muestras que puede obtenerse
    cantSamples = (L-K+1)**2
    #Desviacion tipica del ruido
    sig = 30
    sigma = sig**2
    KK = K**2
    cotaK = (K-1)/2
    cotaL = (L-1)/2



    im = misc.imread('Images/Barbara.tif')
    nImage = colorNoise(im,sig)


    misc.toimage(nImage, cmin=0, cmax=255).save("noisedBarbara.tif")
    noised = misc.imread("noisedBarbara.tif")
    misc.imshow(noised)
    #Se separan los los canales de cada color
    r,g,b = cv2.split(nImage)
    #Se carga en un vector para iterar
    colorChannels = np.array([r,g,b])
```

```
#Solo se necesita la dimension de alguno de los canales
[alto,ancho] = g.shape


#Threshold
T = 25
#Se define la condicion para el lpg
cond = T + 2*sigma
print "Condicion",cond


# Se divide en 3 procesos la limpieza de la imagen
jobs = []
process = 3
for n in range(process):
    p = Process(target=procesar, args=(colorChannels[n],K,KK,L,cantSamples,si
    jobs.append(p)


for n in range(process):
    #Se inicia la limpieza
    jobs[n].start()
for n in range(process):
    jobs[n].join()


#Se leen las imagenes de los procesos para la construccion de la imagen de la
im = misc.imread('Images/Barbara.tif')
c0 = misc.imread('denoisedP0.tif',flatten=1)
c1 = misc.imread('denoisedP1.tif',flatten=1)
c2 = misc.imread('denoisedP2.tif',flatten=1)
```

```
    juntarCanales(im,c0,c1,c2,1)


    #Se leen las imagenes de los procesos para la construccion de la imagen final
    im = misc.imread('Images/Barbara.tif')
    c0 = misc.imread('denoisedP3.tif',flatten=1)
    c1 = misc.imread('denoisedP4.tif',flatten=1)
    c2 = misc.imread('denoisedP5.tif',flatten=1)


    juntarCanales(im,c0,c1,c2,1)
```

## B.6. Agregar ruido (addNoise)

```
from __future__ import division
import numpy as np


#Funcion que agrega ruido a la imagen en escala de grises
def addNoise(im,sig):
    [alto,ancho] = im.shape
    for m in range(alto):
        for n in range(ancho):
            r = np.random.normal(0, sig)
            i=im[m,n];
            j=i+r;
            #Se atiende a que no se pasen los limites permitidos para
            los valores de los pixeles
            if j<0:
                j=0
            if j>255:
```

```
                j=255;


            im[m,n]=j;
    return im


#Funcion que agrega ruido blanco a imagenes a color
def colorNoise(image,sigma):
    row,col,ch= image.shape
    mean = 0
    gauss = np.random.normal(mean,sigma,(row,col,ch))
    gauss = gauss.reshape(row,col,ch)
    noisy = image + gauss
    return noisy
```

## B.7. Calidad de la imagen (calidad)

```
from skimage.measure import structural_similarity as ssim
import matplotlib.pyplot as plt
import numpy as np
import math
from scipy import misc


def mse(imageA, imageB):
# Error medio cuadratico
    err = np.sum((imageA.astype("float") - imageB.astype("float")) **
    2)
    err /= float(imageA.shape[0] * imageA.shape[1])
    return err
```

```
def compare_images(imageA, imageB, title):
# Se computa el error medio cuadratico y el ssim de las imagenes
    m = mse(imageA, imageB)
    s = ssim(imageA, imageB)
    if m == 0:
        psnr = 0
    else:
        psnr = 20*math.log10(255/math.sqrt(m))

    # Se configura la imagen
    fig = plt.figure(title)
    plt.suptitle("MSE: %.2f, SSIM: %.2f, PSNR: %.2f" % (m, s,psnr))

    # Muestra la primera imagen
    ax = fig.add_subplot(1, 2, 1)
    plt.imshow(imageA, cmap = plt.cm.gray)
    plt.axis("off")

    # Muestra la segunda imagen
    ax = fig.add_subplot(1, 2, 2)
    plt.imshow(imageB, cmap = plt.cm.gray)
    plt.axis("off")

    # Muestra las imagenes
    plt.show()

original = misc.imread("Images/Comparar20p/cameraman.tif", flatten=1)
```

```
parallel = misc.imread("Images/Comparar20p/DenoisedParallel.tif",
flatten=1)
final = misc.imread("Images/Comparar20p/finalDenoised.tif", flatten=1)

# Inicializa la figura
fig = plt.figure("Images")
images = ("Original", original), ("Paralelo", parallel), ("Final",
final)

# Se itera sobre las imagenes
for (i, (name, image)) in enumerate(images):
    # Muestra la imagen
    ax = fig.add_subplot(1, 3, i + 1)
    ax.set_title(name)
    plt.imshow(image)#, cmap = plt.cm.gray)
    plt.axis("off")

# Muestra las imagenes
plt.show()

# Compra las imagenes
compare_images(original, parallel, "Original vs. Primera pasada")
compare_images(original, final, "Original vs. Segunda pasada")
```

## B.8. Nivel de ruido (psnr)

```
from __future__ import division
from decimal import *
```

```
from scipy import misc
import numpy as np
import math

#Funcion que retorna el valor de ruido de la imagen limpia en primera
pasada
def getNewNoise(imo,imn,noise):
    [alto,ancho] = imo.shape
    area = (alto-20)*(ancho-20)
    sigma = 0
    for m in range(20,236):
        for n in range(20,236):
            sigma = sigma + (imo[m,n]-imn[m,n])**2
    sigma = sigma/area
    #Factor de ajuste
    fa = 0.27
    #Calculo del nuevo ruido
    newSigma = (fa*math.sqrt(noise-sigma))**2
    print "Factor de ajuste",fa
    return newSigma

#Funcion que retorna el PSNR de la imagen. No se cuentan los 20
pixeles correspondientes a cada borde porque no cuentan con (L-K+1)^2
muestras
def getPsnr(imo,imn):
    [alto,ancho] = imo.shape
    #misc.imshow(imo)
    area = (alto-20)*(ancho-20)
```

```
    sigma = 0
    for m in range(20,236):
        for n in range(20,236):
            sigma = sigma + (imo[m,n]-imn[m,n])**2-imnx[m,n])**2)
    sigma = sigma/area
    return 20*math.log10(255/math.sqrt(sigma))
```

## B.9. Ordenamiento (qs)

```
def quickSort(alist):
    quickSortHelper(alist,0,alist.shape[1]-1,alist.shape[0])


def quickSortHelper(alist,first,last,f):
    if first<last:


        splitpoint = partition(alist,first,last,f)
        quickSortHelper(alist,first,splitpoint-1,f)
        quickSortHelper(alist,splitpoint+1,last,f)



def partition(alist,first,last,f):
    temp = np.zeros([f])
    pivotvalue = alist[f-1,first]
    leftmark = first+1
    rightmark = last
    done = False
    while not done:
        while leftmark <= rightmark and alist[f-1,leftmark] <=
```

```
    pivotvalue:

        leftmark = leftmark + 1


    while alist[f-1,rightmark] >= pivotvalue and rightmark >=

    leftmark:

        rightmark = rightmark -1


    if rightmark < leftmark:

        done = True

    else:

        temp[0:f] = alist[0:f,leftmark]

        alist[0:f,leftmark] = alist[0:f,rightmark]

        alist[0:f,rightmark] = temp[0:f]


temp[0:f] = alist[0:f,first]

alist[0:f,first] = alist[0:f,rightmark]

alist[0:f,rightmark] = temp[0:f]


return rightmark
```

# Bibliografía


[JD02] Emmanuel J. Candès Jean-Luc Starck y David L. Donoho. «The Curvelet Transform for Image Denoising». En: *IEEE Transaction on Image Processing* 11.6 (2002), págs. 670-684.

[CK07] G.Y. Chen y B. Kégl. «Image denoising with complex ridgelets». En: *Pattern Recognition* 49.2 (2007), págs. 578-585.

[JS03] Martin J. Wainwright Javier Portilla Vasily Strela y Eero P. Simoncelli. «Image denoising using scale mixtures of Gaussians in the wavelet domain». En: *IEEE Transaction on Image Processing* 12.11 (2003), págs. 1338-1351.

[EA06] Michael Elad y Michal Aharon. «Image Denoising Via Sparse and Redundant Representations Over Learned Dictionaries». En: *IEEE Transaction on Image Processing* 15.12 (2006), págs. 3736-3745.

[MB06] Michael Elad Michal Aharon y Alfred Bruckstein. «K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation». En: *IEEE Transaction on Image Processing* 54.11 (2006), págs. 4311-4322.

[AE07] Vladimir Katkovnik Alessandro Foi y Karen Egiazarian. «Pointwise Shape-Adaptive DCT for High-Quality Denoising and Deblocking of Grayscale and Color Images». En: *IEEE Transaction on Image Processing* 16.5 (2007).

[TM98] C. Tomasi y R. Manduchi. «Bilateral Filtering for Gray and Color Images». En: *Proceedings of the 1998 IEEE International Conference on Computer Vision, Bombay, India* (1998), págs. 839-846.

[KB06] Charles Kervrann y Jérôme Boulanger. «Optimal Spatial Adaptation for Patch-Based Image Denoising». En: *IEEE Transaction on Image Processing* 15.10 (2006), págs. 2866-2878.

[Egi07] Karen Egiazarian. «Image denoising by sparse 3D transform-domain collaborative ltering». En: *IEEE Transaction on Image Processing* 16.8 (2007), págs. 2080-2095.

[LS10] David Zhang Lei Zhang Weisheng Dong y Guangming Shi. «Two-stage image denoising by principal component analysis with local pixel grouping». En: *Elsevier-Pattern Recognition*. Vol. 43. 2010.

[Tho68] James R. Thompson. «Some Shrinkage Techniques for Estimating the Mean». En: *Journal of the American Statistical Association* 63.321 (1968), págs. 113-122.

[YP12] Dr. M.V. Subramanyam Y. Murali Mohan Babu y Dr. M.N. Giri Prasad. «PCA based image denoising». En: *Signal & Image Processing : An International Journal (SIPIJ)* 3.2 (2012).

[PP06] Aleksandra Pizurica y Wilfried Philips. «Estimating the probability of the presence of a signal of interest in multiresolution single- and multiband image denoising». En: *IEEE Transaction on Image Processing* 15.3 (2006), págs. 654-665.

[Mur10] Shogo Muramatsu. «SSIM Image Quality Metric for Denoised Images». En: *Advances in Visualization, Imaging and Simulation* (2010).

[ZS04] Hamid R. Sheikh Zhou Wang Alan C. Bovik y Eero P. Simoncelli. «Image Quality Assessment: From Error Visibility to Structural Similarity». En: *IEEE Transaction on Image Processing* 13.4 (2004).

[Ric09] Mark Richardson. «Principal Component Analysis». En: (2009).

[Str07] Gilbert Strang. *Algebra Lineal y sus aplicaciones*. Thomson, 2007.