Computación Eficiente y Exacta de Momentos Geométricos en Imágenes en Escala de Grises[1]

TD 24
Carlos Gómez Palacios, Iván Lineros Vera y Alejandro Rafael Irimia Mesa

Resumen

Se propone un algoritmo que permite una rápida y segura computación de momentos geométricos en imágenes en escala de gris. Nuestra idea es obtener dado una imagen que le proporcionemos el cálculo de los momentos geométricos, el centroide y los momentos invariantes de dicha imagen

Key words: computación, escala de gris, momentos geométricos, centroide, momentos invariantes.

1. Introducción

A lo largo de la historia de las imágenes digitales se han utilizado los descriptores de imágenes para obtener información y propiedades descriptivas sobre ellas, y su principal contribución en la ciencia ha sido: *El uso propio como descriptor del objeto que lo hará distinguible del resto de imágenes, para el uso de codificación de imágenes y para tareas de clasificación de patrones.*

Los tipos de descriptores que nos podemos encontrar son variados, pero existen especialmente tres grandes campos que son:

- **Descriptores Topológicos**: Las propiedades topológicas son útiles para descripciones globales de regiones del plano imagen, es decir, definida en forma simple, serán aquellas propiedades de una figura a las que no le afecta ningún deformación.

- **Descriptores Geométricos**: Las propiedades geométricas es una representación simple para poder hacerlo como un tipo de descriptor, entre las propiedades geométricas que nos podemos encontrar están... perímetro, área, longitud.

- **Descriptores Estadísticos**: Es en este tipo de descriptor en donde nos vamos a centrar durante todo nuestro trabajo, concretamente en el cálculo de los momentos de una imagen.

Podemos definir **los momentos de la imagen** como particulares medidas ponderadas de las intensidades de la imagen, de tal forma que el resultado tiene ciertas propiedades.

2. Planteamiento teórico

El objetivo de este trabajo es realizar un sistema que sea capaz de calcular de una forma rápida y segura computación los momentos geométricos en imágenes en escala de gris.

Para ello partiremos en principio del algoritmo clásico, visto en teoría, que servirá para el cálculo de los momentos de una imagen y lo que haremos al final será una comparación entre los dos algoritmos, de tal forma que obtendremos un algoritmo que realizara el cálculo exacto y rápido sin llegar a cometer errores para el cálculo de los momentos geométricos.

2.1. *Algoritmo Tradicional para el cálculo de los momentos:*

La teoría de los momentos proporciona una interesante y alternativa para la representación de formas de objetos, ya que si tenemos un objeto en una región A que viene dado por los puntos en los que f(x,y) es mayor que 0 vamos a tener que él cálculo de los momentos geométricos

Los **momentos cartesianos de orden (p+q)** de una imagen digital I en escala de grises será:

$$M_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^p y^q f(x, y)\, \mathrm{d}x\, \mathrm{d}y. \qquad M_{pq} = \sum_{x=1}^{N} \sum_{y=1}^{N} x^p y^q f(x, y),$$

2.2. *Imágenes Binarias*

Para el caso de imágenes binarias la ecuación anterior se transforma de la siguiente forma:

$$M_{pq} = \sum_{x} \sum_{y} x^p y^q, \quad \forall x, y: f(x, y) = 1.$$

Sin embargo, el cálculo de los momentos geométricos a través de la ecuación anterior es bastante lento. El tiempo de cálculo se puede reducir por un factor significativo, si se describe la imagen en el proceso, por un conjunto de áreas rectangulares de pixeles homogénea, que conocemos como bloques. El método utilizado para dividir las imágenes es el método IBR y se aplica solamente en imágenes binarias.

Una vez que la representación del bloque de una imagen binaria se extrae, el cálculo de los momentos geométricos se puede acelerar en el sentido de que la ecuación anterior se transforma en una forma menos compleja.

El algoritmo de IBR se compone de una sola pasada de la imagen y un proceso de contabilidad, más precisamente el algoritmo se describe mediante los siguientes pasos:

2.2.1. *Algoritmo – IBR*

Los pasos que sigue el algoritmo IBR son los siguientes:

1. Considerar cada línea “y” de la imagen f y encontrar intervalos objetos de nivel de línea “y”
2. Comparación de los intervalos “y” los bloques que tienen los pixeles de la línea “y-1”.
3. Si un intervalo no coincide con ningún bloque, este es el comienzo de un nuevo bloque
4. Si un bloque contenido en el intervalo, el fin del bloque esta en la línea “y”.

Como resultado de la aplicación del algoritmo anterior, es un conjunto de bloques rectangulares, teniendo el valor de la intensidad misma de “1”. Después de la extracción de bloques, la imagen puede ser redefinía en términos de bloques, en lugar de pixeles individuales.

$$f(x,y)=\{b_i:i=0,1,\ldots,k-1\},$$

donde k es el numero de bloques de la imagen. Cada bloque se describe por dos pixeles, la parte superior izquierda y los pixeles de la esquina abajo a la derecha del bloque.

2.3. *Algoritmo PIBR*

Ahora implementaremos el algoritmo PIBR, tal y como se describe en el articulo. Lo primero que tendremos que hacer previamente es extraer los bloques de la imagen y luego hacemos el cálculo del momento geométrico en base al cálculo de momentos binarios de cada bloque.

Para conseguir esto lo que haremos es que para cada intensidad almacenada de la imagen, obtendremos los bloques definidos para esa intensidad y de ahí calcularemos el momento binario de orden (p+q)

$$\begin{aligned}
M_{pq} &= \sum_x \sum_y x^p y^q f(x,y) = \sum_x \sum_y x^p y^q \left(\sum_{i=1}^{n} f_i(x,y) \right) \\
&= \sum_x \sum_y x^p y^q (f_1(x,y) + f_2(x,y) + \cdots + f_n(x,y)) \\
&= \sum_x \sum_y x^p y^q f_1(x,y) + \sum_x \sum_y x^p y^q f_2(x,y) + \cdots \\
&\quad + \sum_x \sum_y x^p y^q f_n(x,y) \\
&= f_1 \sum_{x_1} \sum_{y_1} x_1^p y_1^q + f_2 \sum_{x_2} \sum_{y_2} x_2^p y_2^q \\
&\quad + \cdots + f_n \sum_{x_n} \sum_{y_n} x_n^p y_n^q \\
&= f_1 M_{pq}^1 + f_2 M_{pq}^2 + \cdots + f_n M_{pq}^n,
\end{aligned}$$

La fórmula que utilizaremos para calcular el momento binario es la que sigue. Puede comprobarse que los momentos se calculan en base a las coordenadas del bloque sin tener que realizar un recorrido completo de la matriz

$$M_{pq} = \sum_{i=0}^{k-1} m_{pq}^{b_i} = \sum_{i=0}^{k-1} \left[\left(\sum_{x=x_{1b_i}}^{x_{2b_i}} x^p \right) \left(\sum_{y=y_{1b_i}}^{y_{2b_i}} y^q \right) \right],$$
$$\forall x, y: f(x, y) = 1,$$

Antes de poder calcular los momentos geométricos con nuestro algoritmo hay que hacer un preprocesamiento para extraer los bloques.

192	192	95	95
221	192	95	95
221	28	28	28
234	234	28	95

Donde los resultados que obtenemos son los siguientes como bloque de la imagen:

a

192	192	0	0
0	192	0	0
0	0	0	0
0	0	0	0

b

0	0	95	95
0	0	95	95
0	0	0	0
0	0	0	95

c

0	0	0	0
0	0	0	0
0	28	28	28
0	0	28	0

d

0	0	0	0
221	0	0	0
221	0	0	0
0	0	0	0

e

0	0	0	0
0	0	0	0
0	0	0	0
234	234	0	0

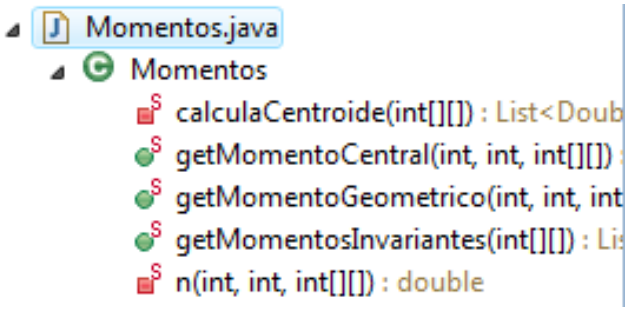
Proposicion1: En el momento (p+q) en el momento geométrico de una imagen en escala de grises es igual a la “Intensidad ponderada” suma de el mismo orden en momentos geométrico de una serie de cortes binarios {0,1}.

3*. Resolución práctica*

A la hora de la implementación debemos destacar ciertas características de las imágenes que deben tener si queremos tratarlas. Nuestro algoritmo solo admite imágenes en escala de grises. Otro detalle a tener en cuenta son las dimensiones de la imagen de entrada, dado que trabajamos con bloques de NxN, si queremos que se trate a toda la imagen en la codificación debemos introducir una imagen cuadrada.

En cuanto a la reutilización de código del trabajo anterior, en este proyecto no hemos podido reutilizar nada. El algoritmo de PIBR lo hemos metido dentro de un paquete llamado útiles.
En cuanto al lenguaje utilizado hemos usado el Java para luego en un futuro poder insertar nuestros algoritmos en ImageJ como plugin. A la hora de la implementación en Java realizamos dos paquetes para la elaboración de dichos algoritmos.

Vamos ahora a colocar una captura de todo nuestro trabajo. A continuación de ella iremos comentando lo que hemos realizado en cada método.



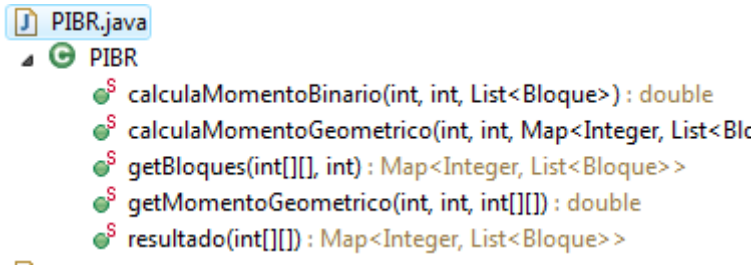
getMomentoCentral (int, int, int[ ] [ ]): Sera un método que nos calcula los momentos central con el algoritmo tradicional.

getMomentoGeometrico (int, int, int [ ] [ ]): Sera un método que nos calcula los momentos geométricos de una imagen.

getMomentosInvariantes (int [ ] [ ]): Sera un método que nos calcula los siete momentos invariantes de una imagen.

calculaCentroide (int [ ] [ ]): Nos calcula el centroide a partir del momento central.

La clase PIBR tendrá lo siguientes métodos:



calculoMomentoBinario (int, int, List<Bloque>): Sera un método que nos calcula los momentos binario de una imagen.

calculoMomentoGeometrico (int, int, Map<Integer, List<Bloque>>): Sera un método que nos calcula los momentos geométricos de una imagen.

getBloques (int [ ] [ ], int): Sera un método que nos calcula los bloques de una imagen.

getMomentoGeometrico (int, int, int [ ] [ ]): Nos obtiene el momento geométrico a partir de los bloques.

También tendremos dos métodos más que nos calcula los bloques de una imagen, además una clase que nos permite manejar todas las características que posee una imagen, por ejemplo, su matriz de intensidad, sus pixeles…

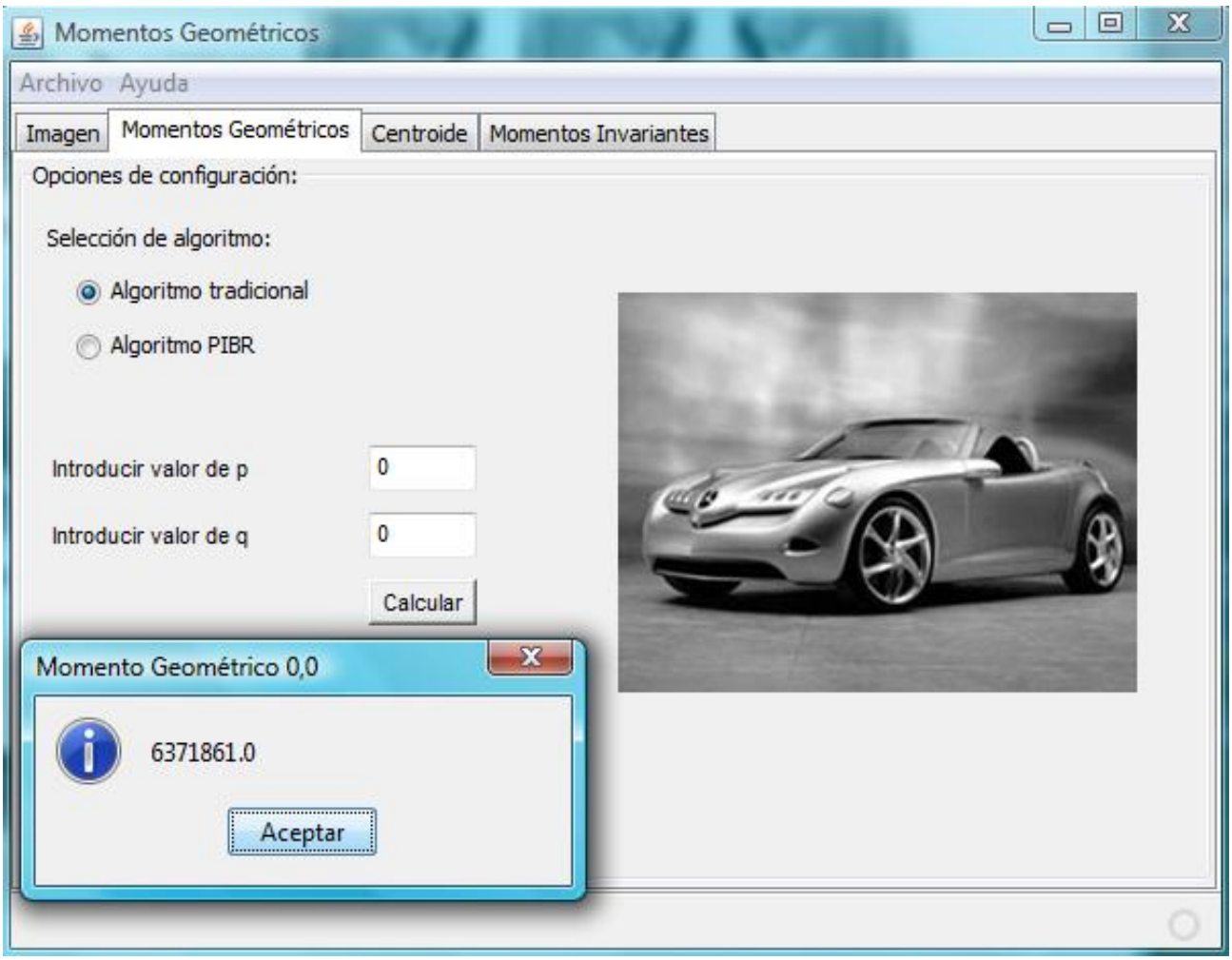
Las librerías que hemos utilizado han sido las librerías propias que se encuentra en la API de Java, además de utilizar las librerías de Math para operar con matrices y operaciones aritméticas en las distintas formulas.

4. *Experimentación*

A la hora de la experimentación vamos a probar con distintas imágenes de prueba de distinto formato en cuanto a dimensiones y tipos de imágenes.

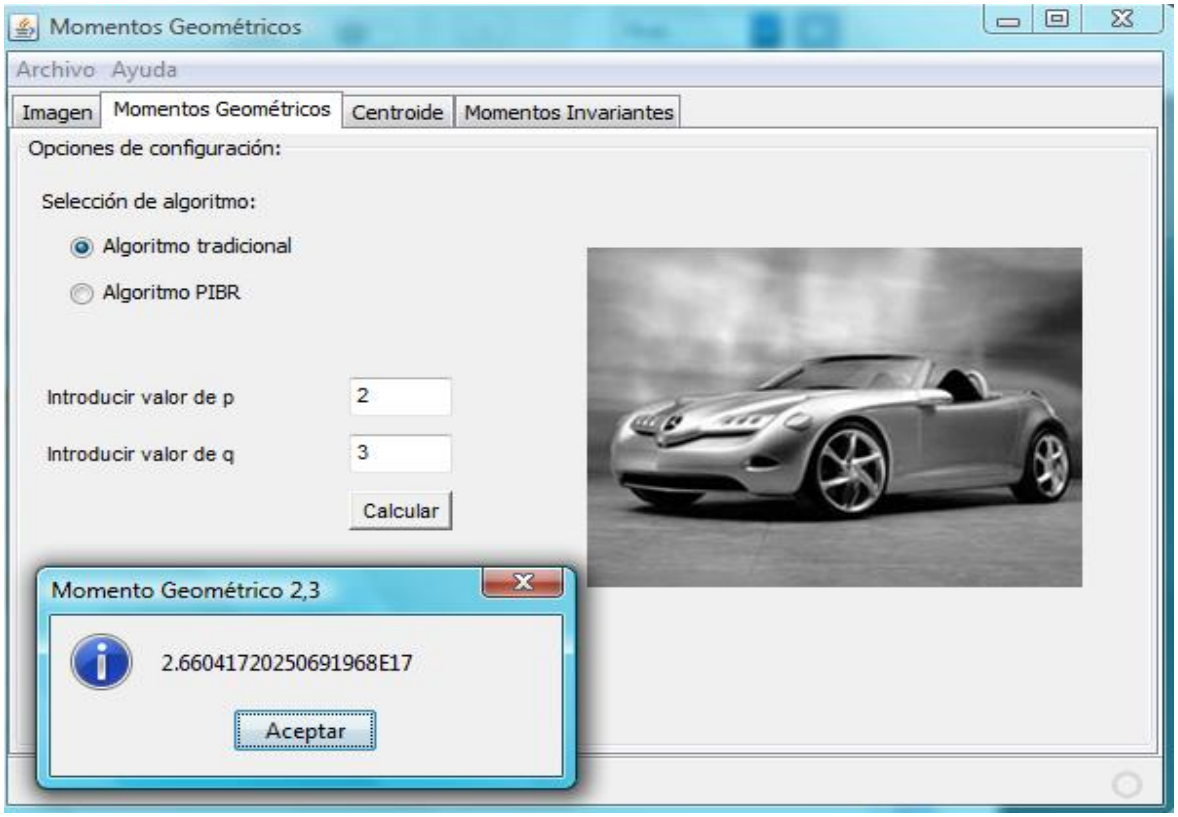
Vamos a cargar nuestra imagen dentro de nuestra aplicación y vamos a tener diferentes opciones para elegir.

Si elegimos el cálculo de los momentos geométricos con el algoritmo tradicional tendremos lo siguiente:

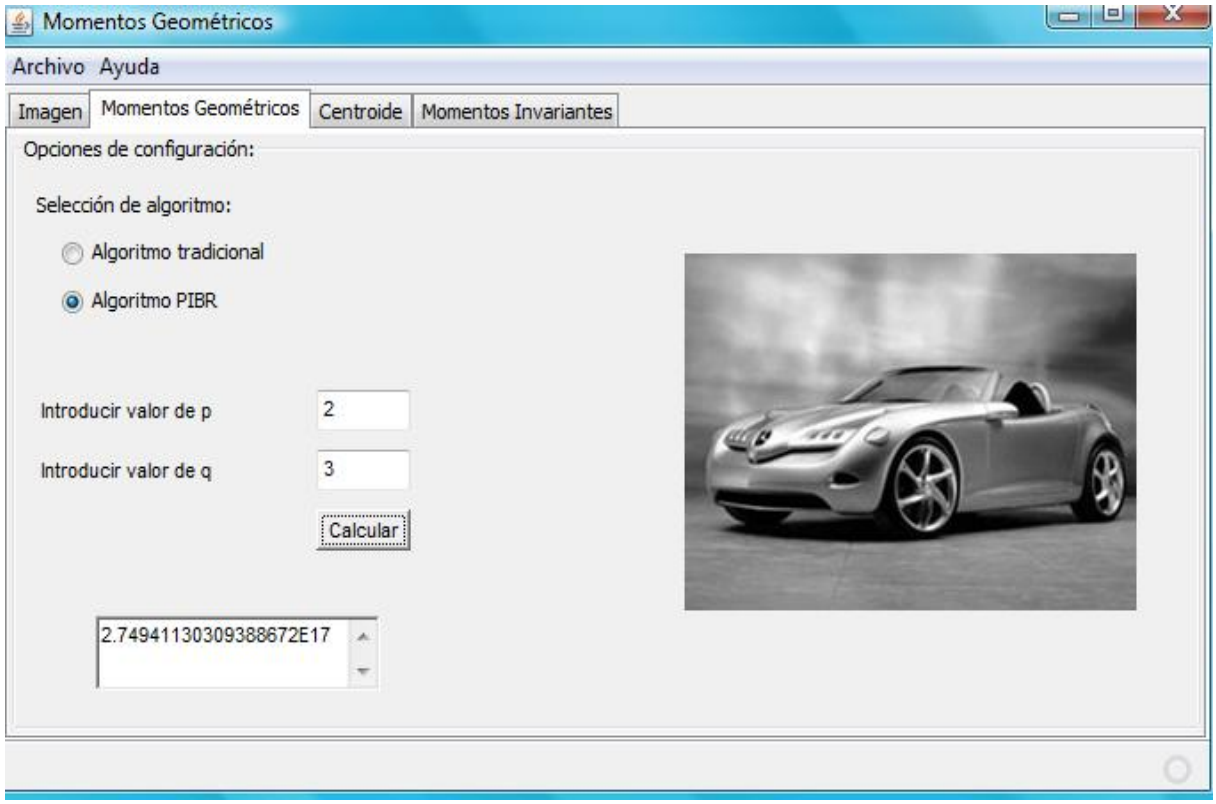


En esta primera prueba hemos calculado el momento geométrico de orden 0 que corresponde con el área de la figura.

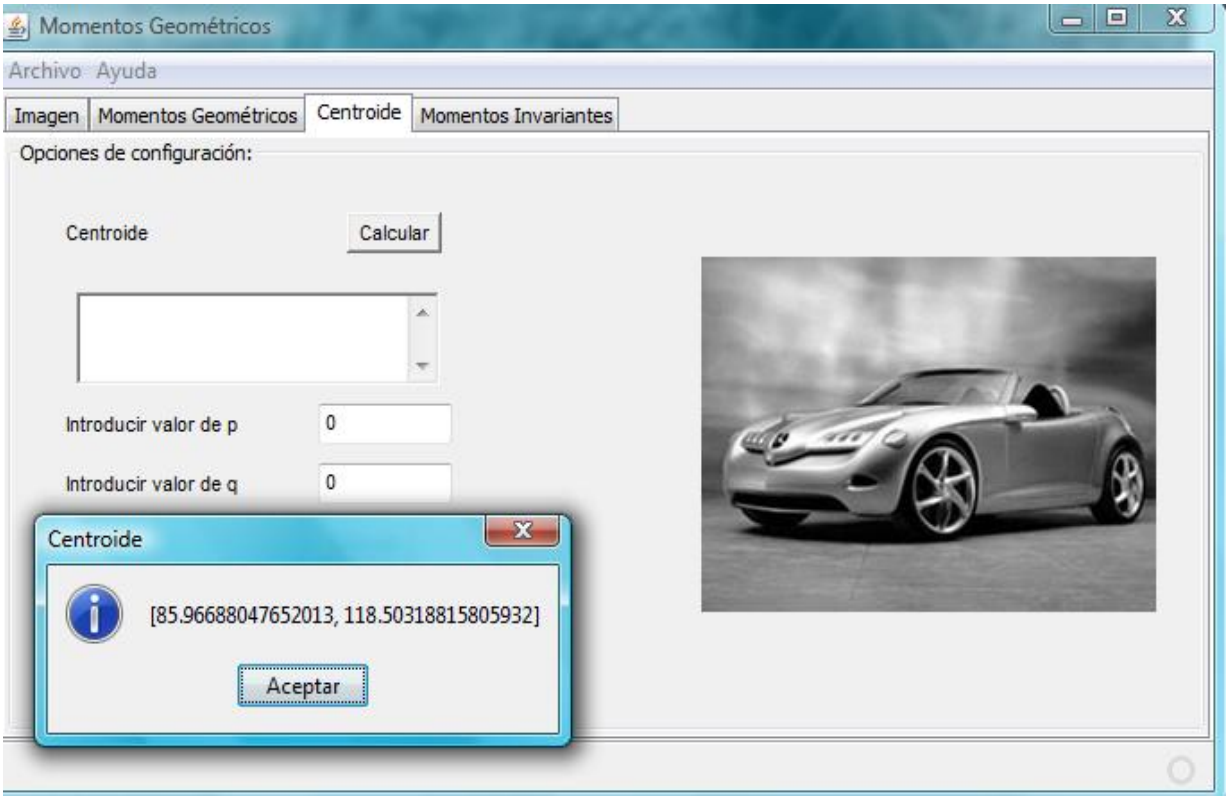
Si queremos calcular otro tipo de momento geométrico como el momento de orden 5 tendremos lo siguiente:



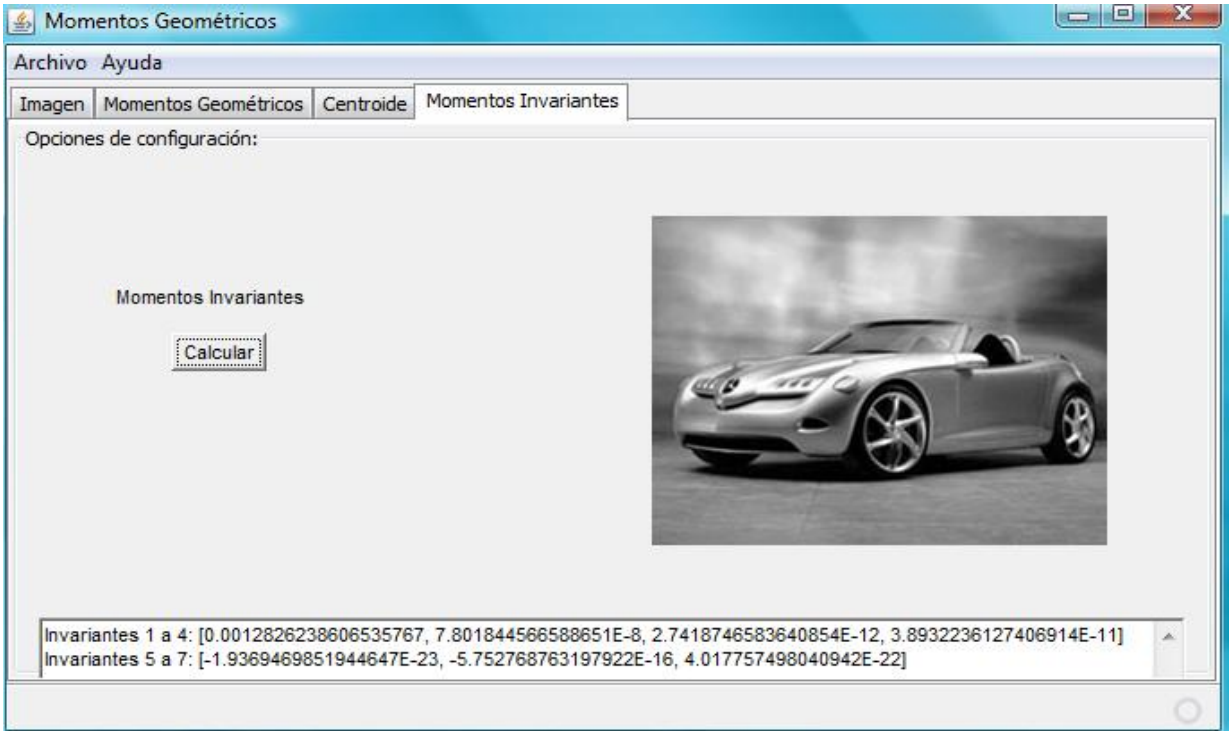
Calculando el momento geométrico de orden 5 con el algoritmo PIBR tendremos:



Si queremos calcular el centroide de nuestra imagen le daremos a la opción de centroide y nos calculara directamente el centroide de nuestra imagen.



Por último podremos calcular los momentos invariantes de dicha imagen, a través de la opción de momentos invariantes de nuestra aplicación y nos devolverá los siete momentos invariantes.



5. *Manual de Usuario*

Para la correcta configuración de la aplicación siga los siguiente pasos:

1) Descomprime el fichero en la carpeta deseada.
(Le aconsejamos en el directorio "c" o en el escritorio).

2) Abre el fichero "momentos geométricos" y a continuación la carpeta "dist".

3) Para iniciar la aplicación haga doble click sobre el fichero

momentos geométricos.jar

4) Las imágenes de prueba que usted puede utilizar se encuentra en la carpeta

"src/imagenes" una vez dentro de la aplicacion solamente tendra que entrar dentro de dichar carpeta.

NOTA.
No separe el ejecutable (Momentos geométricos.jar) del código fuente.

Los resultados obtenidos para nuestro trabajo lo expresamos a continuación:

Resultados de ejecución
Testing3.java

Inicio del Testing de pruebas

Imagen de lena cargada correctamente
Tamanyo de imagen: 256x256 pixeles

Descomposición de imagen
Tiempo: 2649439258ns
Bloques: 60218
Intensidades: 229
Calculo de momentos geometricos
M(0,0)(inicial): 8095286.0
Tiempo: 23978620ns
M(0,0)(PIBR): 8826966.0
Tiempo: 33857097ns
M(0,1)(inicial): 1.077574637E9
Tiempo: 21419211ns
M(0,1)(PIBR): 1.169863871E9
Tiempo: 28498102ns
M(0,2)(inicial): 1.87080453851E11
Tiempo: 14702969ns
M(0,2)(PIBR): 2.00783848309E11
Tiempo: 29103518ns
M(0,3)(inicial): 3.6065724486155E13
Tiempo: 64021359ns
M(0,3)(PIBR): 3.8297998411613E13
Tiempo: 77294088ns
M(2,0)(inicial): 1.67239335411E11
Tiempo: 10987706ns
M(2,0)(PIBR): 1.71874019579E11
Tiempo: 32927558ns
M(2,1)(inicial): 2.2857215072335E13
Tiempo: 13191449ns

M(2,1)(PIBR): 2.364982423276E13
Tiempo: 29675118ns
M(2,2)(inicial): 4.038193849688571E15
Tiempo: 9958463ns
M(2,2)(PIBR): 4.177993427459516E15
Tiempo: 38485955ns
M(2,3)(inicial): 7.860603760979369E17
Tiempo: 63439825ns
M(2,3)(PIBR): 8.1160497129756518E17
Tiempo: 82952083ns
M(3,0)(inicial): 3.1740250177215E13
Tiempo: 67071498ns
M(3,0)(PIBR): 3.2286081668523E13
Tiempo: 83012282ns
M(3,1)(inicial): 4.340183073424579E15
Tiempo: 64192386ns
M(3,1)(PIBR): 4.43843319878679E15
Tiempo: 75836396ns
M(3,2)(inicial): 7.6554340223561869E17
Tiempo: 64923022ns
M(3,2)(PIBR): 7.8321161381449434E17
Tiempo: 75831840ns
M(3,3)(inicial): 1.4852159646999137E20
Tiempo: 117794651ns
M(3,3)(PIBR): 1.5177615419088693E20
Tiempo: 124701019ns

Imagen de restaurante cargada correctamente
Tamanyo de imagen: 550x367 pixeles

Descomposición de imagen
Tiempo: 14081374074ns
Bloques: 171964
Intensidades: 252
Calculo de momentos geometricos
M(0,0)(inicial): 2.9108977E7
Tiempo: 31123672ns
M(0,0)(PIBR): 3.1051721E7
Tiempo: 101490506ns
M(0,1)(inicial): 8.289692556E9
Tiempo: 31972730ns
M(0,1)(PIBR): 8.795113757E9
Tiempo: 104428666ns
M(0,2)(inicial): 3.028457751844E12
Tiempo: 31595072ns
M(0,2)(PIBR): 3.192043361179E12
Tiempo: 109795063ns
M(0,3)(inicial): 1.236636355304544E15
Tiempo: 192118853ns
M(0,3)(PIBR): 1.295905387330367E15
Tiempo: 252890713ns
M(2,0)(inicial): 1.423290768997E12
Tiempo: 31292787ns
M(2,0)(PIBR): 1.485335269289E12
Tiempo: 108283698ns
M(2,1)(inicial): 4.11931324920535E14
Tiempo: 30520007ns
M(2,1)(PIBR): 4.33414462787941E14
Tiempo: 97705033ns
M(2,2)(inicial): 1.53946528256008768E17

Tiempo: 32007131ns
M(2,2)(PIBR): 1.61876645736802624E17
Tiempo: 103377454ns
M(2,3)(inicial): 6.412052943168077E19
Tiempo: 189856385ns
M(2,3)(PIBR): 6.721322525610294E19
Tiempo: 263290764ns
M(3,0)(inicial): 4.03694860190185E14
Tiempo: 189820057ns
M(3,0)(PIBR): 4.18612628902859E14
Tiempo: 277284785ns
M(3,1)(inicial): 1.17793457256078928E17
Tiempo: 198468589ns
M(3,1)(PIBR): 1.2318066392698952E17
Tiempo: 229357806ns
M(3,2)(inicial): 4.436925650383988E19
Tiempo: 188240696ns
M(3,2)(PIBR): 4.641224757234715E19
Tiempo: 229163905ns
M(3,3)(inicial): 1.8591896144378027E22
Tiempo: 352219282ns
M(3,3)(PIBR): 1.940390719857029E22
Tiempo: 378672410ns

Imagen de coche cargada correctamente
Tamanyo de imagen: 800x600 pixeles

Descomposición de imagen
Tiempo: 50032415002ns
Bloques: 295160
Intensidades: 256
Calculo de momentos geometricos
M(0,0)(inicial): 6.2984536E7
Tiempo: 74912976ns
M(0,0)(PIBR): 6.5715076E7
Tiempo: 228244906ns
M(0,1)(inicial): 2.3575795246E10
Tiempo: 120749839ns
M(0,1)(PIBR): 2.4575915737E10
Tiempo: 183802429ns
M(0,2)(inicial): 1.1909667055294E13
Tiempo: 77170589ns
M(0,2)(PIBR): 1.2344327255483E13
Tiempo: 185354539ns
M(0,3)(inicial): 6.855978997344946E15
Tiempo: 456335752ns
M(0,3)(PIBR): 7.066062829629181E15
Tiempo: 527086223ns
M(2,0)(inicial): 6.638717788664E12
Tiempo: 78677995ns
M(2,0)(PIBR): 6.747735976754E12
Tiempo: 181955967ns
M(2,1)(inicial): 2.598298277600993E15
Tiempo: 79503194ns
M(2,1)(PIBR): 2.648287527826753E15
Tiempo: 183132362ns
M(2,2)(inicial): 1.37298895776849818E18
Tiempo: 77644395ns
M(2,2)(PIBR): 1.39747202301135949E18
Tiempo: 180299477ns

M(2,3)(inicial): 8.192903532010658E20
Tiempo: 448590197ns
M(2,3)(PIBR): 8.321311757684432E20
Tiempo: 539963441ns
M(3,0)(inicial): 2.998719377962194E15
Tiempo: 455680513ns
M(3,0)(PIBR): 3.029495690538788E15
Tiempo: 391639274ns
M(3,1)(inicial): 1.18665512439445555E18
Tiempo: 448935855ns
M(3,1)(PIBR): 1.20180585692536934E18
Tiempo: 387778759ns
M(3,2)(inicial): 6.322857030643666E20
Tiempo: 450037070ns
M(3,2)(PIBR): 6.400206052244927E20
Tiempo: 389401809ns
M(3,3)(inicial): 3.794633175624769E23
Tiempo: 835280760ns
M(3,3)(PIBR): 3.8365119564298634E23
Tiempo: 734518951ns

Imagen de paisaje cargada correctamente
Tamanyo de imagen: 1024x768 pixeles

Descomposición de imagen
Tiempo: 168333842813ns
Bloques: 534147
Intensidades: 256
Calculo de momentos geometricos
M(0,0)(inicial): 6.9510176E7
Tiempo: 133684226ns
M(0,0)(PIBR): 7.1633418E7
Tiempo: 374166483ns
M(0,1)(inicial): 2.8681429118E10
Tiempo: 123356184ns
M(0,1)(PIBR): 2.950906618E10
Tiempo: 365112163ns
M(0,2)(inicial): 1.7677171687534E13
Tiempo: 130389980ns
M(0,2)(PIBR): 1.8039295868784E13
Tiempo: 357928226ns
M(0,3)(inicial): 1.2845885316974496E16
Tiempo: 746895960ns
M(0,3)(PIBR): 1.3016905764397428E16
Tiempo: 913328392ns
M(2,0)(inicial): 1.2697932911981E13
Tiempo: 127332911ns
M(2,0)(PIBR): 1.287560403941E13
Tiempo: 351997303ns
M(2,1)(inicial): 5.295006914444774E15
Tiempo: 129845678ns
M(2,1)(PIBR): 5.379632262020391E15
Tiempo: 371365364ns
M(2,2)(inicial): 3.3107321422732375E18
Tiempo: 133853839ns
M(2,2)(PIBR): 3.3525813605455165E18
Tiempo: 373470807ns
M(2,3)(inicial): 2.437273532257604E21
Tiempo: 761212203ns
M(2,3)(PIBR): 2.458962691953522E21

Tiempo: 913119739ns
M(3,0)(inicial): 6.742238684375701E15
Tiempo: 739381072ns
M(3,0)(PIBR): 6.805682789512992E15
Tiempo: 745650554ns
M(3,1)(inicial): 2.8362683811526487E18
Tiempo: 743685263ns
M(3,1)(PIBR): 2.8681839178257812E18
Tiempo: 795189702ns
M(3,2)(inicial): 1.7799670906618244E21
Tiempo: 732921520ns
M(3,2)(PIBR): 1.7962959967649103E21
Tiempo: 741943346ns
M(3,3)(inicial): 1.312912341797856E24
Tiempo: 1367270813ns
M(3,3)(PIBR): 1.321606771630908E24
Tiempo: 1306398444ns

Fin del Testing de pruebas

Todos estos resultados lo hemos obtenido con un equipo con las siguientes características

CPU-Z TXT Report

Binaries

CPU-Z version	1.56

Processors

Number of processors	1
Number of threads	2

APICs

Processor 0
- -- Core 0
  - -- Thread 0 0
  - -- Thread 1 1

Processors Information

Processor 1	ID = 0
Number of cores	1 (max 1)
Number of threads	2 (max 2)
Name	Intel Pentium 4 524
Codename	Prescott
Specification	Intel(R) Pentium(R) 4 CPU 3.06GHz
Package (platform ID)	Socket 775 LGA (0x4)
CPUID	F.4.9
Extended CPUID	F.4
Core Stepping	G1
Technology	90 nm
Core Speed	3067.2 MHz

Multiplier x FSB	23.0 x 133.4 MHz
Rated Bus speed	533.4 MHz
Stock frequency	3066 MHz
Instructions sets	MMX, SSE, SSE2, SSE3, EM64T
L1 Data cache	16 KBytes, 8-way set associative, 64-byte line size
Trace cache	12 Kuops, 8-way set associative
L2 cache	1024 KBytes, 8-way set associative, 64-byte line size
FID/VID Control	no

Thread dumps

CPU Thread 0

APIC ID	0
Topology	Processor ID 0, Core ID 0, Thread ID 0
Type	01001003h
Max CPUID level	00000005h
Max CPUID ext. level	80000008h
Cache descriptor	Level 1, D, 16 KB, 2 thread(s)
Cache descriptor	Level 2, U, 1 MB, 2 thread(s)
Cache descriptor	Level 1, T, 12 KB, 2 thread(s)

CPUID				
0x00000000	0x00000005	0x756E6547	0x6C65746E	0x49656E69
0x00000001	0x00000F49	0x00020800	0x0000651D	0xBFEBFBFF
0x00000002	0x605B5001	0x00000000	0x00000000	0x007C7040
0x00000003	0x00000000	0x00000000	0x00000000	0x00000000
0x00000004	0x00004121	0x01C0003F	0x0000001F	0x00000000
0x00000004	0x00004143	0x01C0103F	0x000003FF	0x00000000
0x00000005	0x00000040	0x00000040	0x00000000	0x00000000
0x80000000	0x80000008	0x00000000	0x00000000	0x00000000
0x80000001	0x00000000	0x00000000	0x00000001	0x20000000
0x80000002	0x20202020	0x20202020	0x20202020	0x6E492020
0x80000003	0x286C6574	0x50202952	0x69746E65	0x52286D75
0x80000004	0x20342029	0x20555043	0x36302E33	0x007A4847
0x80000005	0x00000000	0x00000000	0x00000000	0x00000000
0x80000006	0x00000000	0x00000000	0x04006040	0x00000000
0x80000007	0x00000000	0x00000000	0x00000000	0x00000000
0x80000008	0x00003024	0x00000000	0x00000000	0x00000000

MSR 0x0000001B	0x00000000	0xFEE00900
MSR 0x00000017	0x00120000	0x00000000
MSR 0x0000002C	0x00000000	0x17110917
MSR 0x000001A0	0x00000004	0x22A60089

CPU Thread 1

APIC ID	1
Topology	Processor ID 0, Core ID 0, Thread ID 1
Type	01001003h
Max CPUID level	00000005h
Max CPUID ext. level	80000008h
Cache descriptor	Level 1, D, 16 KB, 2 thread(s)
Cache descriptor	Level 2, U, 1 MB, 2 thread(s)
Cache descriptor	Level 1, T, 12 KB, 2 thread(s)

CPUID				
0x00000000	0x00000005	0x756E6547	0x6C65746E	0x49656E69
0x00000001	0x00000F49	0x01020800	0x0000651D	0xBFEBFBFF

0x00000002	0x605B5001	0x00000000	0x00000000	0x007C7040
0x00000003	0x00000000	0x00000000	0x00000000	0x00000000
0x00000004	0x00004121	0x01C0003F	0x0000001F	0x00000000
0x00000004	0x00004143	0x01C0103F	0x000003FF	0x00000000
0x00000005	0x00000040	0x00000040	0x00000000	0x00000000
0x80000000	0x80000008	0x00000000	0x00000000	0x00000000
0x80000001	0x00000000	0x00000000	0x00000001	0x20000000
0x80000002	0x20202020	0x20202020	0x20202020	0x6E492020
0x80000003	0x286C6574	0x50202952	0x69746E65	0x52286D75
0x80000004	0x20342029	0x20555043	0x36302E33	0x007A4847
0x80000005	0x00000000	0x00000000	0x00000000	0x00000000
0x80000006	0x00000000	0x00000000	0x04006040	0x00000000
0x80000007	0x00000000	0x00000000	0x00000000	0x00000000
0x80000008	0x00003024	0x00000000	0x00000000	0x00000000

MSR 0x0000001B	0x00000000	0xFEE00800
MSR 0x00000017	0x00120000	0x00000000
MSR 0x0000002C	0x00000000	0x17110917
MSR 0x000001A0	0x00000004	0x22A60089

Chipset

Northbridge	SiS 661FX rev. 11
Southbridge	SiS 964 rev. 36
Graphic Interface	AGP
AGP Revision	3.0
AGP Transfer Rate	8x
AGP SBA	supported, enabled
Memory Type	
Memory Size	1024 MBytes

Memory SPD

DIMM #	1
SMBus address	0x50
Memory type	DDR
Manufacturer (ID)	Nanya Technology (7F7F7F0B00000000)
Size	512 MBytes
Max bandwidth	PC3200 (200 MHz)
Part number	NT512D64S88B0GY-5T
Serial number	828B0881
Manufacturing date	Week 39/Year 06
Number of banks	1
Data width	64 bits
Correction	None
Registered	no
Buffered	no
Nominal Voltage	2.50 Volts
EPP	no
XMP	no
JEDEC timings table	CL-tRCD-tRP-tRAS-tRC @ frequency
JEDEC #1	2.5-3-3-7-n.a. @ 166 MHz
JEDEC #2	3.0-3-3-8-n.a. @ 200 MHz
DIMM #	2
SMBus address	0x51

Memory type	DDR
Manufacturer (ID)	Nanya Technology (7F7F7F0B00000000)
Size	512 MBytes
Max bandwidth	PC3200 (200 MHz)
Part number	NT512D64S88B0GY-5T
Serial number	D28908A2
Manufacturing date	Week 39/Year 06
Number of banks	1
Data width	64 bits
Correction	None
Registered	no
Buffered	no
Nominal Voltage	2.50 Volts
EPP	no
XMP	no
JEDEC timings table	CL-tRCD-tRP-tRAS-tRC @ frequency
JEDEC #1	2.5-3-3-7-n.a. @ 166 MHz
JEDEC #2	3.0-3-3-8-n.a. @ 200 MHz

DIMM # 1

SPD registers

	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00	80 08 07 0D 0B 01 40 00 04 50 65 00 82 08 00 01
10	0E 04 18 01 02 20 C0 60 70 00 00 3C 28 3C 28 80
20	60 60 40 40 00 00 00 00 00 37 46 30 28 50 00 01
30	00 00 00 00 00 00 00 00 00 00 00 00 00 00 10 C7
40	7F 7F 7F 0B 00 00 00 00 0D 4E 54 35 31 32 44 36
50	34 53 38 38 42 30 47 59 2D 35 54 20 00 06 27 82
60	8B 08 81 88 00 00 00 00 00 00 00 00 00 00 00 00
70	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
80	00 00 00 00 00 00 00 4D 4D 30 36 36 32 37 30 4B
90	42 00 00 00 00 00 00 44 4E 54 36 32 36 32 38 35
A0	32 00 00 00 00 00 01 89 01 4E 53 54 55 00 34 2D
B0	30 38 00 00 00 00 00 01 4E 53 54 55 00 34 2D 30
C0	38 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D0	80 00 00 00 88 00 00 00 00 00 00 00 00 00 00 80
E0	00 00 00 88 88 01 02 00 00 03 00 00 00 00 00 00
F0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 88

DIMM # 2

SPD registers

	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00	80 08 07 0D 0B 01 40 00 04 50 65 00 82 08 00 01
10	0E 04 18 01 02 20 C0 60 70 00 00 3C 28 3C 28 80
20	60 60 40 40 00 00 00 00 00 37 46 30 28 50 00 01
30	00 00 00 00 00 00 00 00 00 00 00 00 00 00 10 C7
40	7F 7F 7F 0B 00 00 00 00 0D 4E 54 35 31 32 44 36
50	34 53 38 38 42 30 47 59 2D 35 54 20 00 06 27 D2
60	89 08 A2 88 00 00 00 00 00 00 00 00 00 00 00 00
70	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
80	00 00 00 00 00 00 00 4D 4D 30 36 36 32 37 30 4B
90	42 00 00 00 00 00 00 44 4E 54 36 32 36 32 38 35
A0	32 00 00 00 00 00 01 31 01 4E 53 54 55 00 34 2D
B0	30 38 00 00 00 00 00 01 4E 53 54 55 00 34 2D 30
C0	38 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D0	80 00 00 00 88 00 00 00 00 00 00 00 00 00 00 80
E0	00 00 00 88 88 01 02 00 00 03 00 00 00 00 00 00
F0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 88

6. Conclusiones

Las conclusiones que hemos sacado después de la realización del trabajo han sido que el objetivo fundamental de nuestro trabajo que era el cálculo de los momentos geométricos ha sido conseguido, además de ver las principales propiedades que tienen los momentos estadísticos dentro de una imagen, además de haber cumplido el cálculo de los momentos geométricos con el algoritmo PIBR.

Hemos visto que tiempo de extracción de los bloques es mayor que el que se muestre en el documento del trabajo, pero dado el pseudocódigo del algoritmo ha sido imposible poder mejorarlo.

Por último el algoritmo tradicional del cálculo de los momentos sigue siendo más rápido que el cálculo de dichos momentos utilizando los bloques que resulta de la ejecución del algoritmo PIBR.

Las posibles mejorar son el poder aplicar no solamente a imágenes en escala de grises, sino también a imágenes en color.

Cambiar la ecuación que calcula los momentos binarios por otra más eficiente, en el trabajo se menciona una ecuación pero no está descrita completamente.

7. *Tiempos*

Cuaderno de registro de tiempos

Fecha	Inicio	Fin	Tiempo Interrupción	Δ tiempo	Actividad	Comentarios	Miembros del grupo
14/12/2010	13:00	14:00	0	60	TP	Test Previo	Iván.
15/12/2010	12:00	13:50	30	80	AN	Análisis de documentación	Alejandro, Carlos e Iván.
19/12/2010	17:00	18:00	10	50	PA	Programacion del algoritmo	Carlos e Iván.
04/01/2011	10:00	12:40	30	130	PA	Programacion del algoritmo	Alejandro.
04/01/2011	16:00	18:40	20	140	PA	Programacion del algoritmo	Alejandro y Carlos.
05/01/2011	10:00	12:00	30	90	PrA	Pruebas del algoritmo	Alejandro, Carlos e Iván.
05/01/2011	16:00	20:30	90	180	DI	Diseño de interfaz	Carlos.
07/01/2011	10:00	13:00	50	130	DI	Diseño de interfaz	Carlos.
07/01/2011	16:00	19:00	40	140	DI	Diseño de interfaz	Carlos.
08/01/2011	10:00	13:00	30	150	PI	Pruebas de interfaz	Carlos.
08/01/2011	16:00	19:00	50	130	PI	Pruebas de interfaz	Carlos.
10/01/2011	11:00	13:30	60	90	D	Documentación	Iván.
10/01/2011	16:00	19:30	60	150	D	Documentación	Alejandro e Iván.
12/01/2011	16:00	19:30	60	150	D	Documentación	Iván.
13/01/2011	15:00	19:30	90	180	D	Documentación	Alejandro, Carlos e Iván.
14/01/2011	09:30	11:00	20	70	MI	Mejoras de interfaz	Carlos.
14/01/2011	10:30	13:00	30	120	P	Presentación	Iván.
14/01/2011	18:00	20:00	0	120	PA	Programacion del algoritmo	Alejandro.
15/01/2011	09:30	11:00	20	70	MI	Mejoras de interfaz	Carlos.
15/01/2011	9:00	14:00	40	260	PA	Programacion del algoritmo	Alejandro.
15/01/2011	15:00	22:00	80	340	PA	Programacion del algoritmo	Alejandro.
15/01/2011	13:30	15:00	0	90	P	Presentación	Iván.
16/01/2011	11:00	13:55	15	160	PA	Programacion del algoritmo	Alejandro.
16/01/2011	21:00	3:00	40	320	P	Presentación	Iván.
17/01/2011	16:00	19:00	30	120	UD	Ultimar detalles	Alejandro, Carlos e Iván.

Tiempo de Alejandro = 1770

Tiempo de Carlos = 1520

Tiempo de Iván = 1500

8. Referencia

[1]G.A. Papakostas, E.G. Karakasis, D.E. Koulouriotis " Efficient and accurate computation of geometric moments on gray-scale images."

[2] R. Krishnamoorthi. Transform coding of monochrome images with statistical design of experiments approach to separate noise, Pattern Recognition Letters., 28 (7) (2007) 771–777.

[3] http://ev.us.es/

[4] J.Flusser, "Refined moment calculation using image block representation", IEEE Trans. Image Process 9 (11) (2000)

[5] es.wikipedia.org