UNIVERSIDAD DE CANTABRIA

FACULTAD DE CIENCIAS

Análisis de Imagen y Estereología aplicadas al conteo de partículas

(Image analysis and stereology applied to population size estimation)

TRABAJO FIN DE GRADO PARA
ACCEDER AL GRADO EN FÍSICAS

Autor: Ignacio López García
Director: Marcos Cruz Rodríguez
OCTUBRE, 2019

Agradecimientos

Este trabajo no podría haberlo terminado si no fuera por toda la ayuda y el apoyo recibido por muchas personas a las que quiero dedicar este apartado.

En primer lugar, me gustaría dar las gracias a Marcos por proponerme este proyecto tan interesante hace ya casi un año. Han sido meses de investigación y simulaciones en las que hemos descubierto innovaciones considerables. Gracias también por tu empuje en que me volcase a culminar y plasmar sobre el papel estos descubrimientos.

En segundo lugar, quiero agradecer a INGRAM MICRO por entender mi situación particular en las últimas fechas y facilitarme la reducción a la mitad de mi jornada laboral. Sin su ayuda, no habría tenido tiempo virtual para terminar este proyecto en las fechas previstas.

Por último, a mi familia, por su apoyo constante e incondicional en todo lo logrado para llegar hasta aquí. Sobre todo, por su compresión estos meses en los que apenas hemos podido vernos. Por vosotros.

ÍNDICE GENERAL

1 - RESUMEN Y PALABRAS CLAVE

El problema de dimensionar poblaciones de gran tamaño es muy recurrente en muchos ámbitos científicos, sociales o biológicos. En la actualidad, nos encontramos con varios métodos muy extendidos tales como el conteo manual exhaustivo, el método de densidad o la visión automatizada por computadora. Todos ellos conllevan limitaciones importantes en cuanto a precisión y/o tiempo de ejecución.

En este trabajo, nos centraremos en el problema de estimar una población de partículas (personas, aves, árboles, etc.) a partir de imágenes aéreas fijas. El método *CountEm*, nos permite hacerlo con errores bajos y predecibles de manera insesgada independientemente del tamaño de la población. Sin embargo la varianza muestral puede variar dependiendo de la distribución espacial de la población. Vamos a simular patrones de partículas con diferentes distribuciones espaciales y con la ayuda del método de Mone Carlo, analizaremos los coeficientes de error empírico para cada una de ellas. Teniendo en cuenta los resultados obtenidos, proponemos adaptar el protocolo de elección de los parámetros iniciales del estimador de *CountEm*.

Palabras clave: *quadrat, estimación del tamaño poblacional, muestreo sistemático, método de Monte Carlo, método insesgado.*

Population size estimation is relevant to biological and social sciences. Current methods include exhaustive manual counting, computer vision, or the density method. Unfortunately, all of them have important limitations in precision and/or counting time.

Here we address population size estimation with the CountEm method on fixed images. The CountEm method allows unbiased population size estimation with low and predictable errors, irrespective of population size. The sampling variance depends on the spatial distribution of the population. We simulate point populations with different spatial distributions and calculate their empirical coefficients of error. Based on the results, we propose to adapt the protocol to select sampling parameters in the *CountEm* method.

Keywords: Quadrat, population size estimation, systematic sampling, Monte Carlo method, unbiased method.

2 - INTRODUCCIÓN

La estimación del tamaño de una población de N elementos es un problema relevante en estudios de ámbito científico, biológico y social. Dependiendo del contexto de la población, el método que se utiliza para calcular N varía totalmente.

Por ejemplo, en muchos estudios en el ámbito de la biología, la tarea de estimar una población se realiza de maneral manual. Si nos encontramos en un espacio abierto y con suficiente luz, podríamos en principio identificar todos los elementos de interés y realizar el conteo a partir de fotografías tomadas desde el aire. Esta es una práctica habitual en los censos de aves en parques naturales [1]. Sin embargo, este proceso resulta tedioso, lento y difícil de verificar.

Cuando nos encontramos ante poblaciones de gran tamaño, el conteo manual queda descartado y se recurre a diferentes técnicas de muestreo con distintos enfoques como el método tradicional de densidad [2]. Este método, es ampliamente utilizado por los medios de comunicación, la policía o los organizadores de diversos eventos multitudinarios. Se basa en el estudio específico del espacio donde se halla la población y la cantidad de espacio que ocupa cada individuo. Se calcula el número de personas que hay por metro cuadrado y multiplicando por el área total del evento, se estima el número total de personas. Sin embargo, con esta técnica de muestreo no se dan criterios claros sobre la región elegida para calcular la densidad, cómo corregir los efectos de borde, etc.

Otros métodos muy extendidos son los basados en el conteo automatizado por ordenador. Funciona bien en algunos casos particulares como lo son los que presentan patrones regulares de partículas no superpuestas que yacen sobre fondos homogéneos [3-4]. Aun cumpliendo estas premisas, los algoritmos automatizados generalmente están sesgados y presentan un bajo rendimiento [5].

En 2015, científicos de la Universidad de Cantabria propusieron un método insesgado de estimación del tamaño de la población denominado *CountEm* [6]. Está basado en principios bien conocidos de muestreo geométrico en estereología y se puede aplicar a cualquier tipo de partículas independientemente del tamaño y patrón de distribución. El único requisito necesario es que las partículas deben ser identificables sin ambigüedad, de tal manera que se pueda realizar el conteo manual de una muestra pequeña de partículas (típicamente entre 50 y 100) con el fin de poder estimar poblaciones de cualquier tamaño y un coeficiente de error empírico inferior al 10% [6-7]. El coeficiente de error empírico, que definiremos de forma más precisa en las siguientes secciones, es equivalente al error relativo y lo calculamos hallando la desviación típica de un gran número de estimaciones replicadas por el método de Monte Carlo.

En 2018, los mismos científicos de la Universidad de Cantabria publicaron otro artículo sobre *CountEm* [7]. En él, utilizaron un total de 51 imágenes con distribuciones aleatorias [Figura 1] con el propósito de verificar la validez de los criterios prácticos discutidos anteriormente, así como analizar los parámetros óptimos de la rejilla usada como estimador para el muestreo.

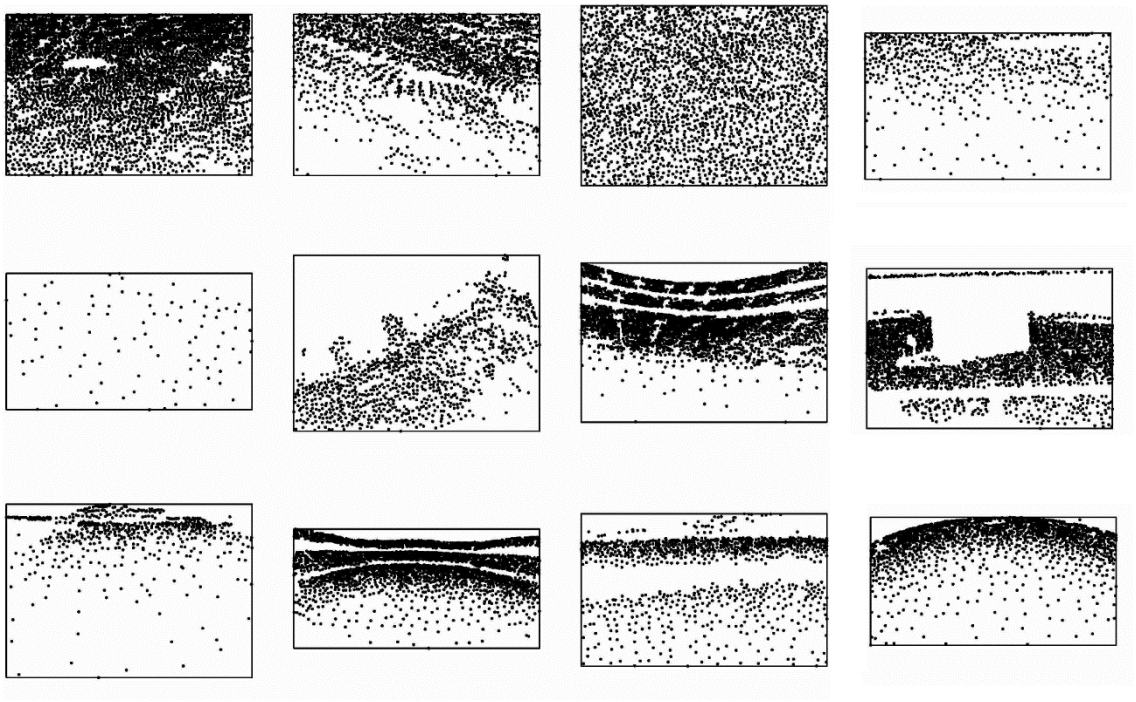

Figura 1. Estas imágenes fueron obtenidas del conjunto de datos UCF [10]. Todas ellas son fotografías de aglomeraciones de personas y cada punto representa las coordenadas de una persona. Estas personas fueron contadas una a una en un largo proceso de conteo manual, por lo que la población total de individuos es conocida.

Si observamos las imágenes utilizadas y los resultados que se obtuvieron del coeficiente de error empírico [Figura 11, que se explicará en apartados posteriores], se puede intuir que dichos resultados fluctúan, no solo por los parámetros de precisión de nuestro estimador, sino también por el patrón de dispersión. Así, se decidió clasificar las imágenes en diferentes patrones geométricos que se repiten frecuentemente en la distribución de elementos de una población, por ejemplo, de aves [Figura 8] y, a través del procedimiento de Monte Carlo, analizar los coeficientes de error empírico para cada una de ellas. Teniendo en cuenta los resultados obtenidos, vamos a proponer un cambio en el protocolo de elección de los parámetros iniciales del estimador de *CountEm*.

2.1 - EL MÉTODO COUNTEM

Vamos a suponer una población de partículas fija y determinista, representada por un conjunto discreto y acotado de elementos, es decir, que puede estar contenida en un disco de radio finito. Antes de continuar con el funcionamiento, vamos a describir la notación utilizada en el artículo [7]:

- *N*: número de partículas de la población objetivo.
- *T*: separación entre las ventanas de muestreo o *quadrats*.
- *t*: longitud del lado del *quadrat*.
- *Q*: número total de partículas contadas en los *quadrats*.
- *n*: número de *quadrats* no vacíos.
- $\widehat{N}$: nuestro estimador del tamaño de la población.

Para estimar el tamaño de una población *N*, *CountEm* usa un estimador de conteo sistemático $\widehat{N}$, formado por una rejilla de ventanas de muestreo con una orientación arbitraria en el plano. De esta forma, evitamos que nuestra rejilla adopte el mismo patrón de dirección que las partículas y así añadir errores de sesgo a nuestra medida. Finalmente, $\widehat{N}$ se calcula multiplicando el número de partículas por el cociente de las áreas de los cuadrados de lado t y T [Figura 2.B].

$$\widehat{N} = \frac{T^2}{t^2} Q = \frac{A}{a} Q \tag{1}$$

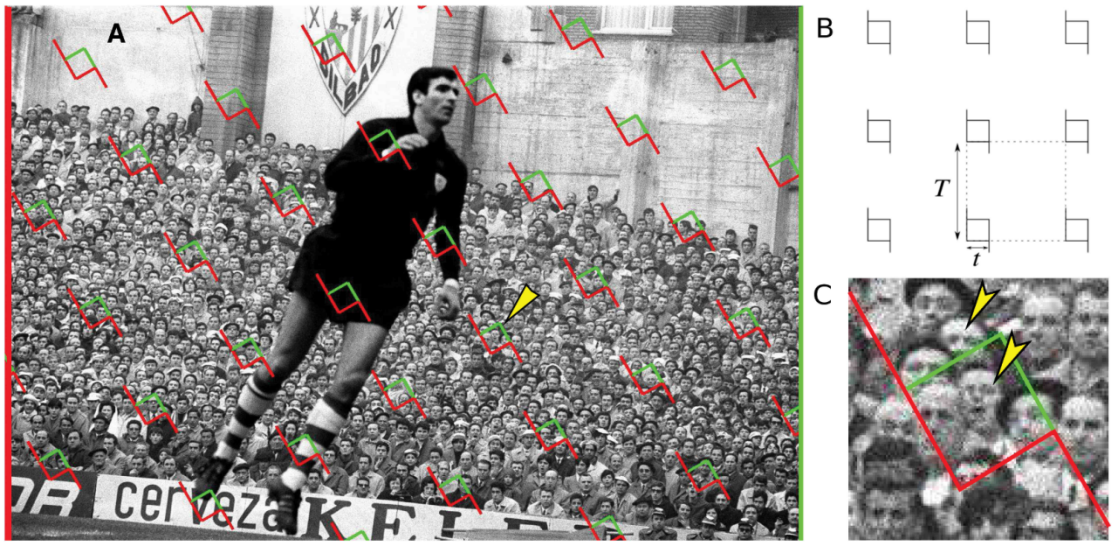

Figura 2. (A) Superponemos una rejilla cuadrada de *quadrats* al azar sobre la imagen para estimar el número total de espectadores. El cuadrante marcado está ampliado en (C). (B) La rejilla está formada por *quadrats* de parámetros T y t, como se indica. (C) Regla de la línea prohibida para eliminar efectos de borde en el conteo [8]. Solo las cabezas señaladas en amarillo se encuentran en el interior del *quadrat*, las demás golpean el borde prohibido de color rojo. Imagen extraída de [7].

Se han dado algunos criterios prácticos para elegir los parámetros *T* y *t* de la rejilla [6]; sin embargo, en la práctica, la elección de los valores iniciales de estos parámetros no siempre es obvia. Por eso, se simplificaron estas pautas [7] utilizando una parametrización más conveniente:

- *f*: fracción de muestreo, $f = \frac{t^2}{T^2}$
- n_0: número inicial de cuadrantes de la rejilla, $n_0 = \frac{B_x \cdot B_y}{T^2}$ donde B_X y B_Y representan el ancho y la altura de la imagen en píxeles, respectivamente.
- Por lo tanto: $\widehat{N} = \frac{Q}{f}$

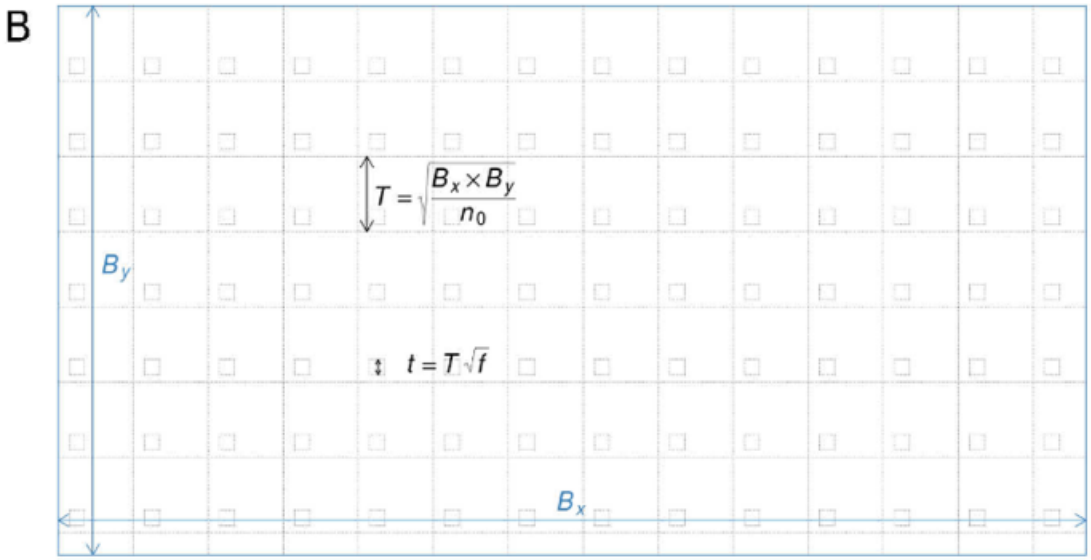

Figura 3. Relación entre los parámetros *{f, n_0}* y *{t, T}*. El tamaño del cuadrado de lado T de la rejilla, es igual al área total de la rejilla $B_x \cdot B_y$ dividido por el número inicial de cuadrantes, n_0. El tamaño del quadrat de lado t, es el tamaño del cuadrado de lado *T* multiplicado por la raíz cuadrada de la fracción de muestreo, *f*. Imagen extraída de [8].

Con este nuevo protocolo, los pasos a seguir por el usuario para estimar los individuos de una población serían los siguientes:

- Recortar la imagen, excluyendo las regiones vacías. Este paso es opcional en cuanto a la obtención de los resultados, pero es muy recomendable para aumentar la eficiencia y evitar contar un número elevado de *quadrats* vacíos.
- Elegir valores adecuados de *f* y n_0. Típicamente, podemos comenzar con $n_0 = 100$ y $f = 0.04$.
- Superponemos nuestra rejilla de *quadrats* al azar. Podemos, de manera voluntaria, inclinar un ángulo fijo para evitar alineación de *quadrats* con partículas.

- Mediante una inspección visual superficial [Figura 4], verificamos que Q y n se encuentren entre los siguientes rangos según el coeficiente de error deseado:

$$Q \approx 50 \ \& \ n \approx 20 \ \Rightarrow \ CE_e\left(\widehat{N}\right) < 15\%$$

$$Q \approx 100 \ \& \ n \approx 50 \ \Rightarrow \ CE_e\left(\widehat{N}\right) < 10\%$$

- Contamos el número total de partículas en los *quadrats*, Q, teniendo en cuenta la regla de la línea prohibida y aplicamos la ecuación (1) para obtener $\widehat{N}$.

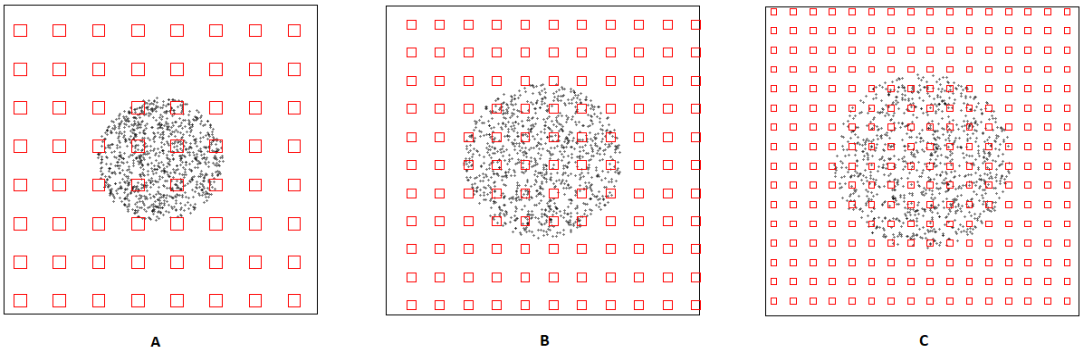

Figura 4. (A) disposición de nuestra rejilla cuadrada de *quadrats* con n_0 = 49 y $f = 0.4$. Tenemos entre 10 y 12 *quadrats* no vacíos (n). (B) Como n era demasiado bajo, aumentamos n_0 hasta 121. Si, por el contrario, Q pareciera demasiado bajo, aumentaríamos f. En (C) hemos ajustado ambos parámetros de forma que tenemos n superior a 50 (aproximadamente 64) y Q algo superior a 100. Con esta elección, podemos obtener errores entre el 5% y el 10%.

Utilizando estas nuevas parametrizaciones de n_0 y f, se reduce el error de manera más intuitiva que con los parámetros anteriores, incluso para usuarios no experimentados. La fuerza de este protocolo se verificó estudiando al detalle los rangos de error correspondientes a diferentes conjuntos de parámetros. En el apartado de presentación de datos y simulaciones, profundizaremos más en la idoneidad de esta elección.

2.2 – VARIANZA DE $\widehat{N}$, ERROR DE ESTIMACIÓN

En una situación práctica, es posible usar varios estadísticos como estimadores puntuales para un parámetro de la población. Dos características son valiosas en un estimador puntual. Primero, la distribución muestral del estimador se debe centrar sobre el verdadero valor del parámetro por estimar. Al estimador en el que la media de su distribución es igual al verdadero valor del parámetro, lo llamamos insesgado. La segunda característica deseable, es que la varianza (medida de la dispersión) de la distribución muestral debe ser tan pequeña como sea posible.

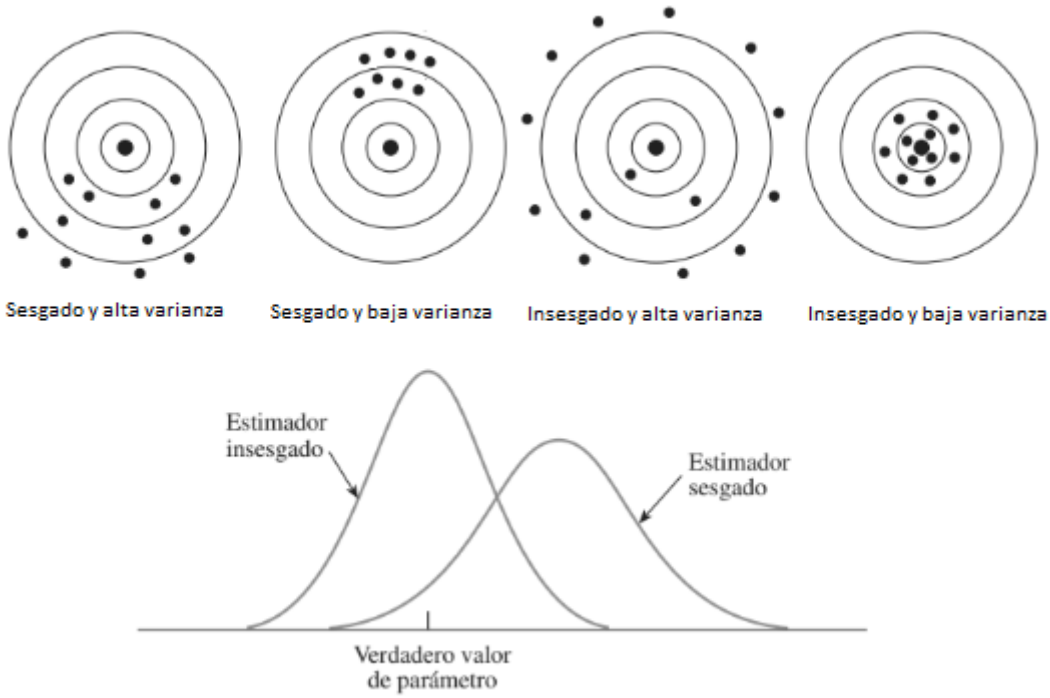

Figura 5. En esta representación podemos observar la diferencia entre el sesgo y la varianza de un estimador. Con el ejemplo de las dianas, se entiende fácilmente que, aunque nuestro estimador sea muy preciso (tenga poca varianza), si no está centrado en el punto del verdadero valor del parámetro (incurre en errores de sesgo), el error cometido puede ser elevado.

Ya ha sido demostrado que el método *CountEm* es insesgado [6] por lo que, en nuestro trabajo, el error cometido dependerá únicamente de la varianza. Normalmente, para determinar si un estimador es mejor que otro, observamos su comportamiento en el muestreo repetido, descrito por su distribución muestral. En nuestro caso, la técnica de muestreo repetido usada será el método de Monte Carlo.

Vamos a generar un total de $k^2 = 32 \cdot 32 = 1024$ superposiciones repetidas de nuestra rejilla de *quadrats* sobre la población de partículas. Para cada *k*, se calculará la población total de la muestra con la ecuación (1):

$$\widehat{N}_k = \left(\frac{T}{t}\right)^2 \cdot Q_k \tag{2}$$

Y la media empírica, la varianza y el Coeficiente de error al cuadrado de $\widehat{N}$, se obtienen, respectivamente, de la siguiente manera:

$$E_e\left(\widehat{N}\right) = k^{-2} \sum_{k=1}^{k^2} \widehat{N}_k \tag{3}$$

$$Var_e\left(\widehat{N}\right) = k^{-2} \sum_{k=1}^{k^2} \left[\widehat{N}_k - E_e\left(\widehat{N}\right)\right]^2 \tag{4}$$

$$CE_e^2\left(\widehat{N}\right) = \frac{Var_e(\widehat{N})}{N^2} \tag{5}$$

Estos valores empíricos, estarán muy cercanos a los respectivos valores reales si *k* es suficientemente alto. En [6] y [7] se consideró suficiente tomar *k*=32.

3 - DATOS Y SIMULACIONES

Anteriormente, hablábamos de los motivos que nos llevaron a realizar el estudio de la varianza a partir de los resultados publicados sobre el estimador de población de *CountEm*. En esta sección, vamos a describir con detalle los datos de partida, metodología de simulación y herramientas de trabajo.

Para la realización de este trabajo, hemos tenido a nuestra disposición una base de datos de más de 200 imágenes de decenas de miles de aves en las cuales, al igual que con las aglomeraciones de personas del conjunto de datos UCF [10], conocíamos el número exacto de la población. Pudimos observar que las aves, como muchas otras especies animales, no siempre definen estructuras geométricas arbitrarias. Después de analizar diferentes estructuras, según si las aves se estaban desplazando por el aire, alimentándose formando cúmulos o posadas alrededor de un estanque de agua; hemos decidido clasificar 4 estructuras diferentes que adoptan las aves frecuentemente para su estudio [Figura 6].

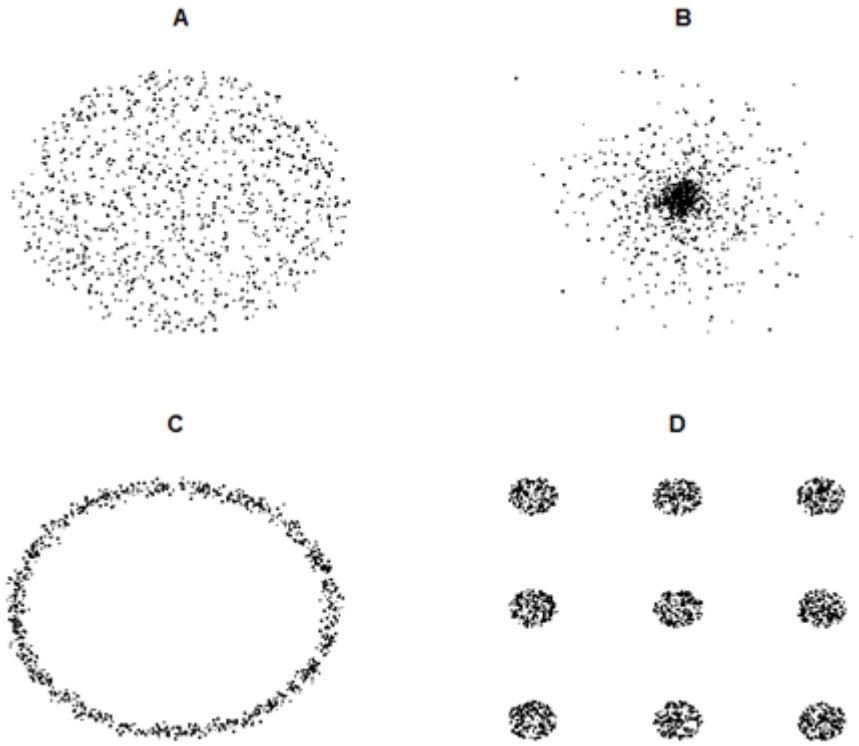

Figura 6. Después de centenares de miles de simulaciones, observamos que, a estas distribuciones concretas, la varianza les afecta de manera bastante diferente. (A) se corresponde con una distribución uniforme de los individuos y es donde mejor resultados obtiene el método. (B) se corresponde con un clúster atómico o cúmulo en el que los individuos se apelotonan sobre todo alrededor de un punto fijo y, según nos alejamos, el número de individuos está más disperso. (C) se corresponde con distribuciones alrededor de un contorno donde el centro no hay apenas individuos y, por último, (D) se corresponde con varios grupos de individuos y algo mucho más interesante, la posibilidad de aplicar nuestro método sobre 9 fotos al mismo tiempo con una sola rejilla estimadora.

A continuación, vamos a mostrar una comparativa entre los patrones que más se repetían en nuestra base de datos y las figuras geométricas que hemos simulado. También, mostraremos las fotos originales, de tal manera que se pueda tener una impresión visual de lo adecuado que ha sido elegir estos patrones y no otros.

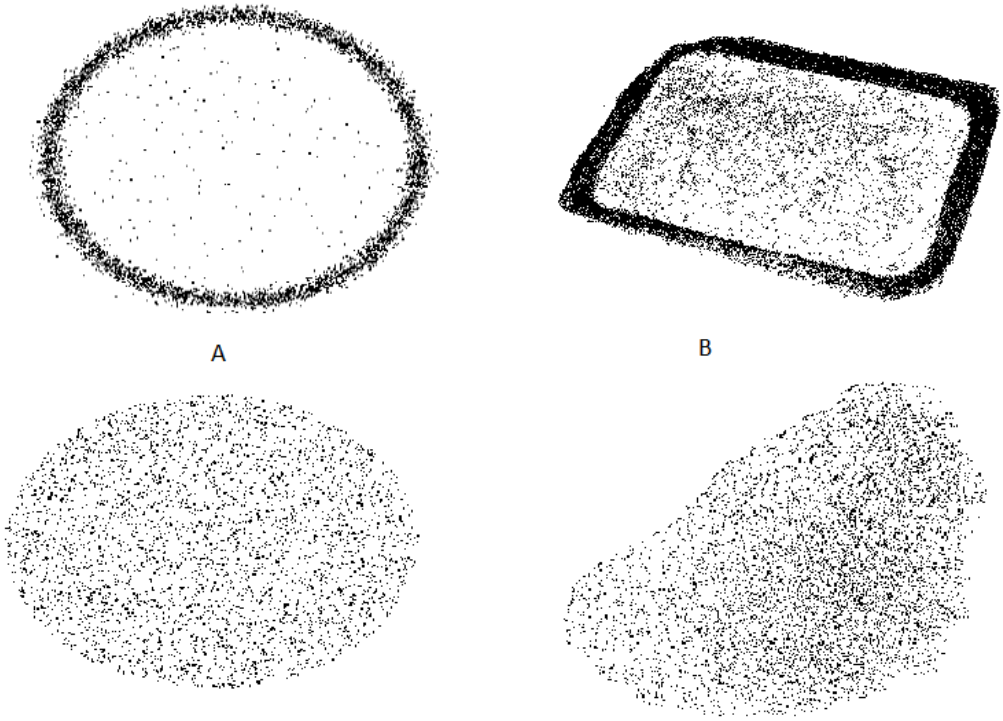

Figura 7. En (A) tenemos dos patrones de dispersión simulados por ordenador. La de arriba, es una de las agrupaciones más habituales que hemos observado en nuestra base de datos. Es muy habitual que las aves formen esos contornos alrededor del estanque cuando las aguas están muy frías o simplemente no se están bañando. Mientras que la de abajo, simula una fotografía aérea de un grupo de aves volando en bandada. En (B) tenemos la representación de dos fotos originales y las coordenadas de las aves obtenidas por conteo manual. Coordenadas extraídas del conjunto de datos Eastern Canada Flocks (ECA-Flocks, Canadian Wildlife Service) que se publicará próximamente.

Figura 8. Fotos originales tomadas desde el aire de las representaciones de puntos aportadas en la Figura 7.B. Imágenes extraídas del conjunto de datos Eastern Canada Flocks (ECA-Flocks, Canadian Wildlife Service) que se publicará próximamente.

Las simulaciones que hemos llevado a cabo, las hemos realizado gracias a la librería de código abierto *spatstat* que se puede instalar y agregar al entorno de programación estadístico R. *Spatstat* proporciona una herramienta para analizar patrones de puntos tanto bidimensionales como tridimensionales (nosotros nos centraremos en los primeros). Contiene más de 2000 funciones para trazar datos espaciales, análisis de datos exploratorios, ajuste de modelos, simulación, y muestreo espacial; por lo que es una herramienta idónea para nuestros propósitos.

A continuación, explicaremos brevemente cómo hemos simulado estos patrones. Todos ellos están formados por 1000 partículas construidas con diferentes funciones de *spatstat*, salvo el mosaico que contiene una población total de 9000 partículas al estar formado por 9 representaciones individuales de 1000. El código de cada una de las figuras simuladas, así como el de la función con el muestreo repetido de Monte Carlo, se pueden estudiar de manera pormenorizada en el apéndice.

Los discos que representan distribuciones de partículas uniformes los hemos construido con la función *runifdisc(n, r)* de R; la cual, nos devuelve una muestra de *n* puntos aleatorios distribuidos de manera uniforme en un disco de radio *r*.

El clúster atómico, o cúmulo, lo hemos construido a partir de generar 2 distribuciones normales con diferente σ (desviación estándar), pero mismo valor central, tanto en el eje *x* como el eje *y* [Figura 9]. La función que nos devuelve esta distribución de partículas es *rnorm(n, mean, sd)* donde *n* es el numero de puntos, mean el valor central y *sd* la desviación estándar σ.

Figura 9. Utilizando 2 veces la función *rnorm()* en torno al mismo valor central pero con distinto σ, podemos lograr distribuciones como el cúmulo (B) de la Figura 6. En nuestro caso, hemos puesto $n = 500$ para cada función para obtener una población total de 1000 partículas. Mientras que σ se eligió de manera que el diámetro del cúmulo fuese del mismo tamaño que el del disco uniforme: *rnorm(500, 0, 1)* y *rnorm(500, 0, 5).*

Para simular el contorno de partículas en forma de anillo, hemos utilizado la función ya conocida *rnorm(n, mean, sd)* y la función *runif(n, a, b)* que genera *n* puntos uniformemente aleatorios entre los valores de *a* y de *b*. Podemos construir un anillo de 1000 partículas a través del siguiente código:

*theta = runif(1000, 0, 2*pi)*
r = rnorm(1000, 20, 1)
*X = r * sin(theta)*
*Y = r * cos(theta)*

Eligiendo en la función *rnorm()* el mismo radio que hemos usado tanto en el disco uniforme como en el cúmulo. Cabe destacar aquí, que si escogemos un radio pequeño y una sigma grande (con lo que aumentaríamos el grosor del anillo) estaríamos dibujando estructuras geométricas similares al disco uniforme, a un donut con un agujero diminuto, lo cual carece de todo interés para este trabajo. Por ello, el grosor del anillo se ha mantenido constante en todas las simulaciones.

En el caso del mosaico, nos encontramos en una situación similar que con el anillo y es que podríamos separar los grupos de partículas tanto como quisiéramos y complicar el estudio hasta límites que no tienen interés alguno. Precisamente por ello, por el interés de nuestras simulaciones, hemos construido mosaicos cuyas estructuras apenas están separadas por el tamaño de una muestra de igual tamaño [Figura 10]. Que suceda esto es algo muy habitual si la foto de nuestras aves no está recortada y focalizada en la población a estudiar. Además, con esto es suficiente para obtener resultados por encima de lo deseado en cuanto a la varianza del estimador.

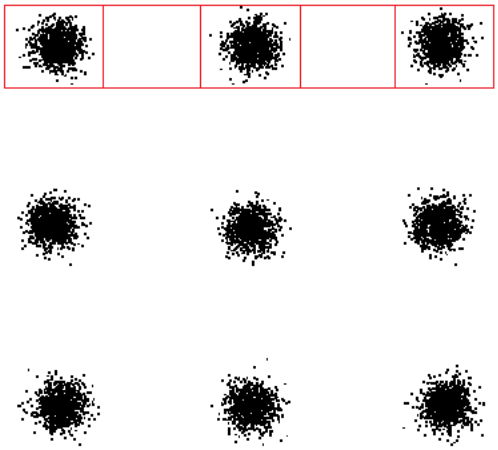

Figura 10. Como se puede observar en esta representación individual del mosaico de 9 estructuras diferentes de población igual a 1000, la separación entre ambas está diseñada de forma que no supone una distancia mayor a la inclusión de una imagen intermedia.

4 - RESULTADOS Y CONCLUSIONES

Comenzamos este apartado, recordando las conclusiones obtenidas en el artículo que supone el origen de este trabajo. Las simulaciones se realizaron con 2 rejillas diferentes, una de 50 ventanas de muestreo y otra de 100, ambas presentaban resultados muy similares, siendo ligeramente mejores cuando el número de *quadrats* era n_0 = 100. Como ya comentamos en el apartado 2.1 y también se puede observar en la Figura 11, basta con contar un número de partículas Q = 100 con el método *CountEm* para lograr obtener errores de estimación del orden, o inferiores, al 10%.

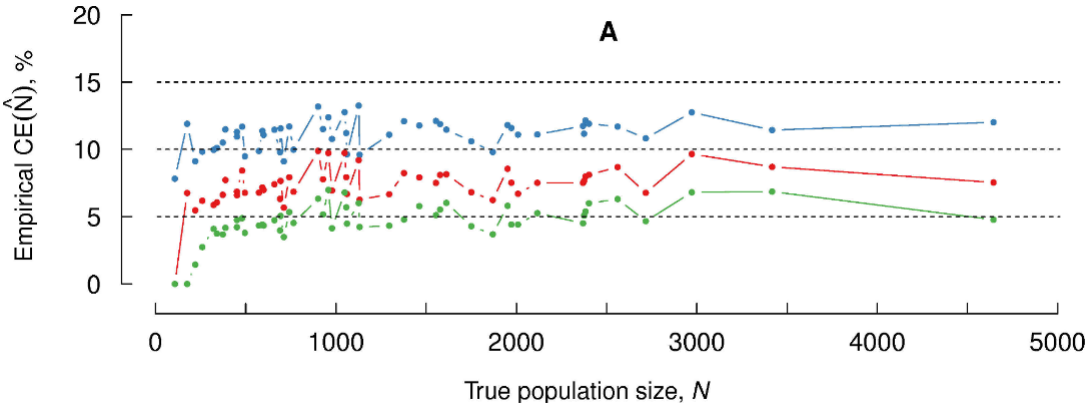

Figura 11. Gráfica que nos muestra el coeficiente de error empírico obtenido en cada una de las 51 imágenes de aglomeraciones de personas con una rejilla de 100 ventanas de muestreo. Los colores azul, rojo y verde representan tamaños de muestra Q = 50, Q = 100 y Q = 200 respectivamente. Gráfica extraída de [7].

Solo nos falta elegir el parámetro de la fracción de muestreo. Si nos fijamos bien en la ecuación que lo define, $f = \frac{t^2}{T^2}$ no es más que la fracción de área muestreada respecto al total. Es decir, si f = 1, estaríamos contando todos los puntos la imagen y si f = 0, ninguno (lo que sería absurdo). Por lo tanto, f representa de alguna manera el tanto por uno de puntos que contamos respecto del total. Si en todas nuestras simulaciones tenemos una población de 1000 partículas, y queremos contar 100, la fracción de muestreo que vamos a elegir es de f = 0.1.

Recordemos antes de continuar que, sobre cada patrón simulado, vamos a estimar la población de partículas 1024 veces para calcular la varianza empírica correspondiente, como hemos explicado anteriormente. Las primeras simulaciones, las realizamos sobre 10 discos de partículas aleatorias distribuidas de manera uniforme. Este es el caso que más se asemeja a la gran mayoría de patrones en las imágenes que tenemos y, al mismo tiempo, donde mejores resultados obtenemos para la varianza con nuestro estimador [Figura 12]. La mayoría de las veces, no solo las bandadas de aves, sino también las aglomeraciones de personas en eventos forman estructuras en grupos ciertamente homogéneos. Por lo que este método con esta convención de parámetros de inicio es idóneo la gran mayoría de imágenes en las que queramos estimar la población de individuos.

Sin embargo, hay otro tipo de estructuras o patrones de dispersión menos habituales que también se repiten como las que vamos a considerar a continuación. Si observamos la Figura 12, podemos ver que la varianza aumenta considerablemente en el caso de estructuras en forma de anillo, cúmulos o el caso especial del mosaico.

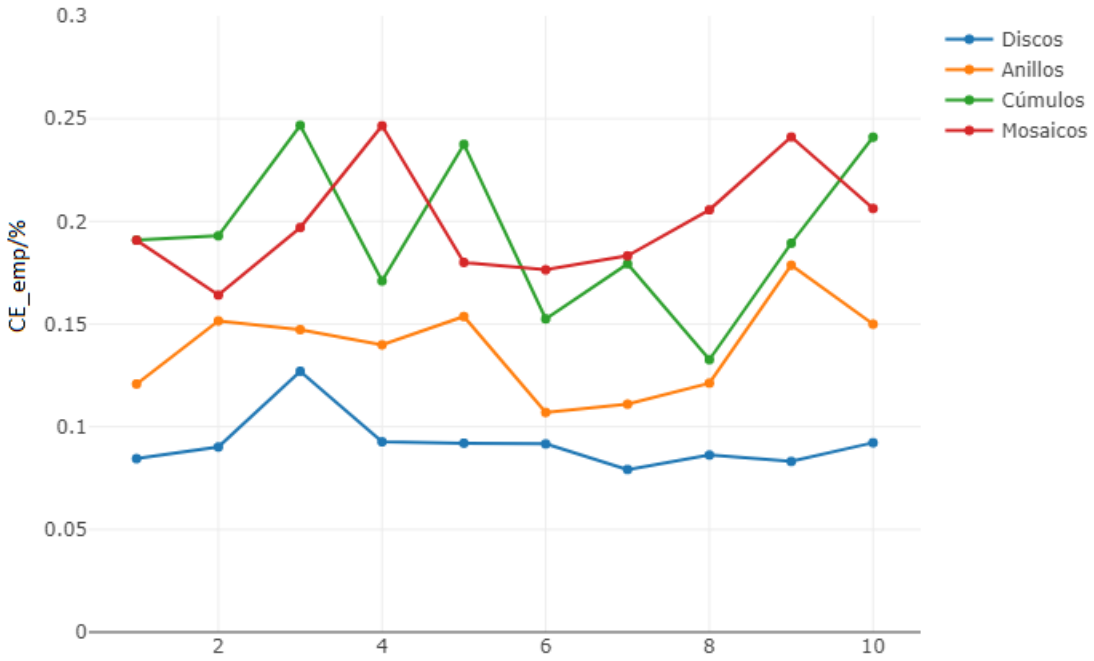

Figura 12. Error empírico obtenido para 40 patrones de dispersión diferentes. En azul, tenemos 10 patrones uniformes en forma de disco [Figura 6.A] donde se observa una varianza poco oscilante en torno al 10%. En amarillo, tenemos 10 patrones de dispersión en forma de anillo [Figura 6.C] donde la varianza es más oscilante, entre el 11% y el 18 % aproximadamente. En verde, tenemos 10 patrones de dispersión correspondientes a cúmulos de partículas [Figura 6.B], en los que la varianza oscila de manera más amplia que en el resto de los casos, entre el 14% y el 25%. Y, por último, en color rojo tenemos el caso especial del mosaico formado por 9 imágenes con 9 patrones de dispersión [Figura 6.D]. Aquí, es donde tenemos mayores resultados para la varianza, llegando a ser en más de una ocasión de entorno al 25%.

Si representamos ahora, la cantidad de ventanas de muestreo, o *quadrats* que han sido ocupadas por partículas en cada uno de los casos podemos intuir el origen del problema; y, es que, cuando tenemos un número muy pequeño de *quadrats* no vacíos no se obtienen buenos resultados con este estimador.

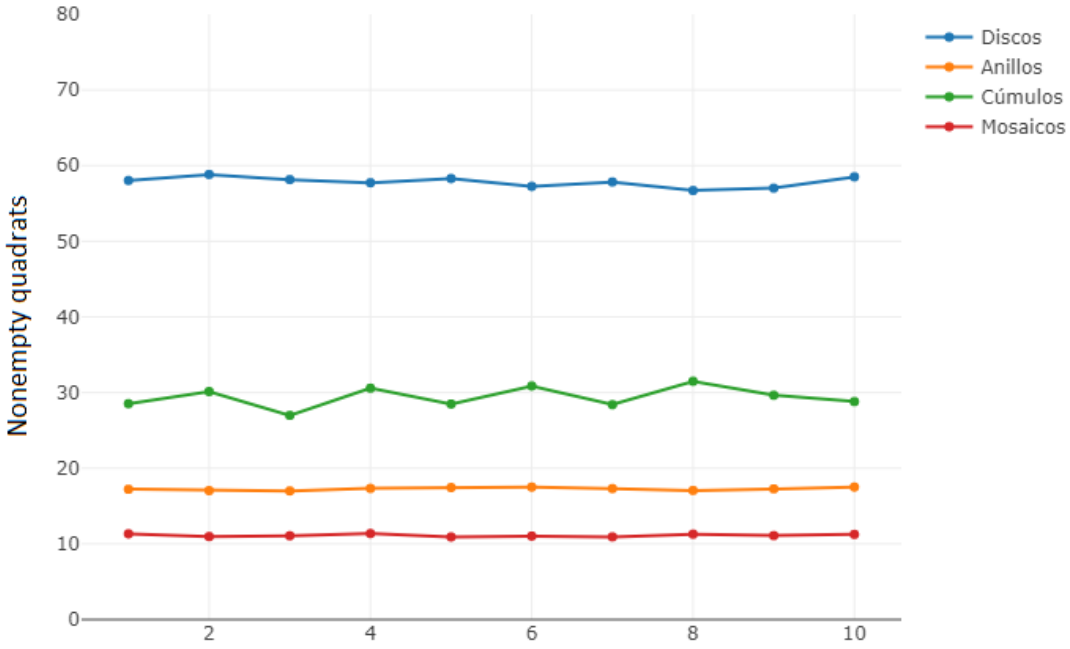

Figura 13. Representación del numero de ventanas de muestreo no vacío para cada patrón de dispersión simulado. En la leyenda de la figura se puede ver a que color corresponde cada tipo de patrón simulado. Como la rejilla era de 100 ventanas de muestreo, el número de *quadrats* vacíos será la resta de 100 menos los no varios que se muestran en la gráfica.

La solución que se nos ha ocurrido para poder elevar el número de ventanas de muestreo ocupadas por las partículas que forman cada uno de nuestros patrones de dispersión es, aumentar la división de la rejilla de n_0 = 100 a n_0 = 300. Veamos que sucede con esta nueva rejilla si la aplicamos sobre los mismo 40 patrones simulados anteriormente.

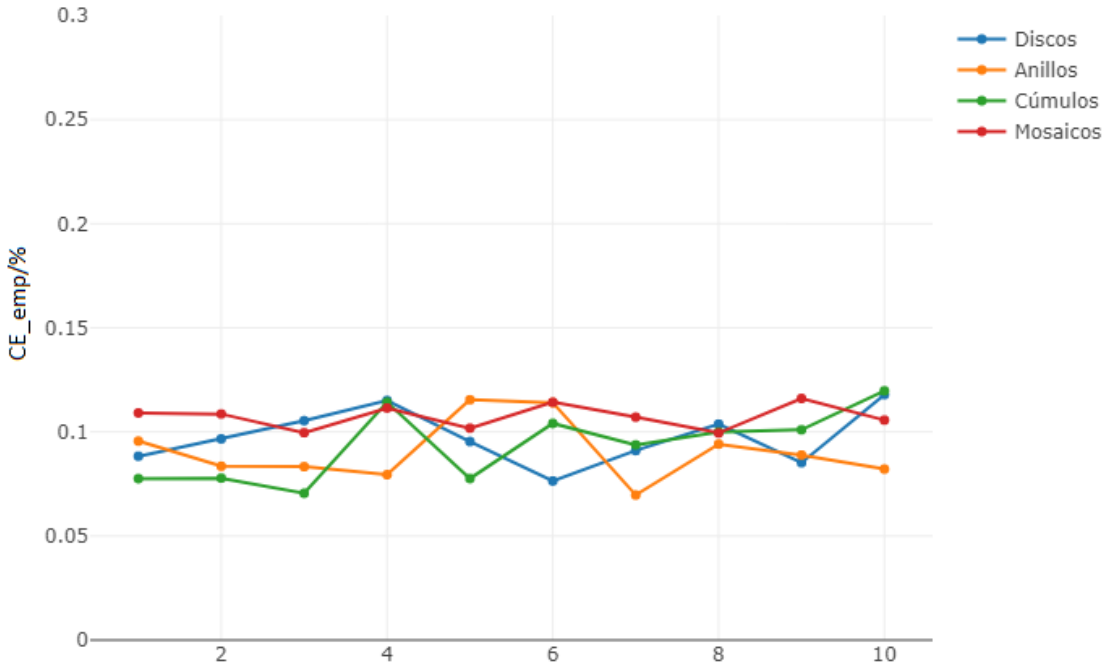

Figura 14. Error empírico obtenido para los mismos patrones de dispersión de la Figura 12. En la leyenda se puede ver a que color corresponde cada tipo de patrón simulado.

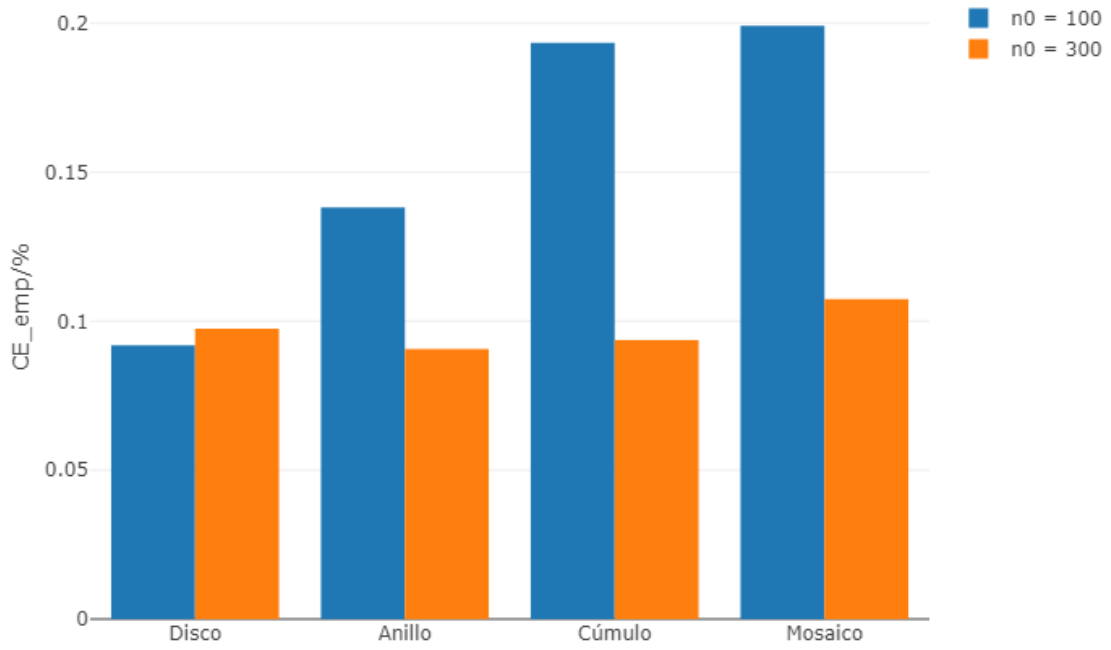

Figura 15. Diagrama de barras donde podemos observar la media, de las 10 figuras de cada patrón, del coeficiente de error empírico. En azul, la media del error usando nuestra rejilla de 100 ventanas de muestreo y en naranja usando la rejilla mejorada de 300 ventanas de muestreo.

De los datos que nos aportan las Figuras 14 y 15 podemos concluir que en los patrones de estructuras con dispersión uniforme obtenemos valores deseados de entorno al 10% con ambas rejillas, estas figuras son las más habituales a la hora de resolver el problema de estimar el número de partículas de una población dada; lo que habla muy bien del método estimador utilizado. Por otro lado, en cuanto a los anillos y a los cúmulos, desde la observación de los resultados obtenidos, proponemos un cambio en el protocolo de selección de los parámetros iniciales de la rejilla estimadora. En estos casos, vemos necesario la utilización de una rejilla de al menos 300 ventanas de muestreo.

En el caso del mosaico, proponemos 2 alternativas. En la sección 2.1, cuando hablábamos del protocolo establecido en el artículo [7], se afirmaba que recortar la foto era recomendable pero únicamente opcional, para evitar contar demasiados *quadrats* vacíos. Viendo los resultados en el caso del mosaico, recomendamos recortar la foto siempre que sea posible para evitar crear tanto espacio entre los patrones y, en este caso, perjudicar la calidad de la estimación y aumentar la varianza. La segunda alternativa sería, al igual que con los anillos y los cúmulos, cambiar el protocolo de la misma forma y aumentar el número de *quadrats* de nuestra rejilla a $n_0 = 300$.

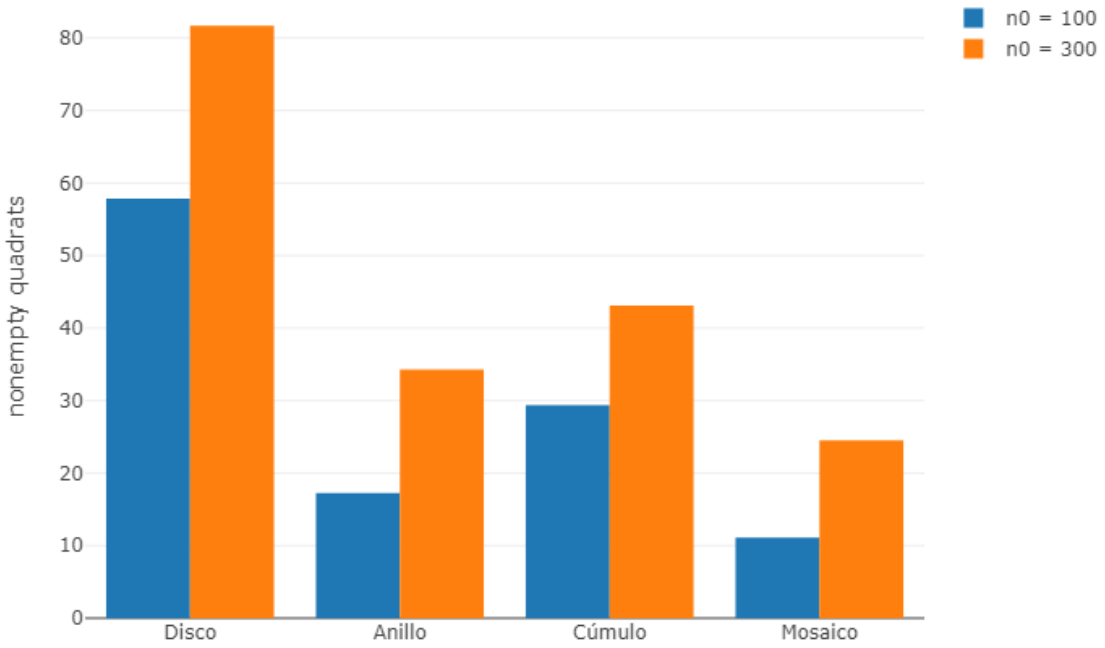

Figura 16. Representación gráfica de barras en la que se puede observar la cantidad media de ventanas de muestreo no vacías para cada figura, con la rejilla de 100 *quadrats* y con la de 300. Nos da una idea de la importancia que tiene conseguir un número cercano a 30 *quadrats* no vacíos para obtener errores del orden del 10%.

Vamos a terminar, viendo un caso práctico de utilización de *CountEm* sobre una foto real de aves en la que conocemos la población exacta. Se trata de un tipo de patrón de los que hemos estudiado que presentan dificultades con el antiguo protocolo del método.

La foto elegida, junto con las coordenadas de cada ave contadas a mano, las podemos ver en la Figura 18. Los resultados obtenidos con n_0 = 100 son:

- $E_e(\widehat{N})$ = 14549
- $CE_e\ (\widehat{N})$ = 0.15
- n = 15.6

Mientras que con n_0 = 300:

- $E_e(\widehat{N})$ = 14595
- $CE_e\ (\widehat{N})$ = 0.076
- n = 34.8

En este ejemplo práctico podemos confirmar todos los supuestos y conclusiones comentados anteriormente. Sabemos que la población de aves de la foto es igual a N = 14558 y la media de la distribución muestral para todos los $\widehat{N}$, calculada con la ecuación (3), es muy cercana al valor real de este parámetro tanto para n_0 = 100 como para n_0 = 300. Esto demuestra algo que ya sabíamos, que nuestro método es insesgado. Por otro lado, el coeficiente de error empírico, calculado con la ecuación (5), que obtenemos con la rejilla de 100 ventanas de muestreo es de aproximadamente el 15%, dato que conseguimos disminuir hasta el 7.6% aplicando el protocolo propuesto en este trabajo (n_0 = 300) para patrones de dispersión de este tipo.

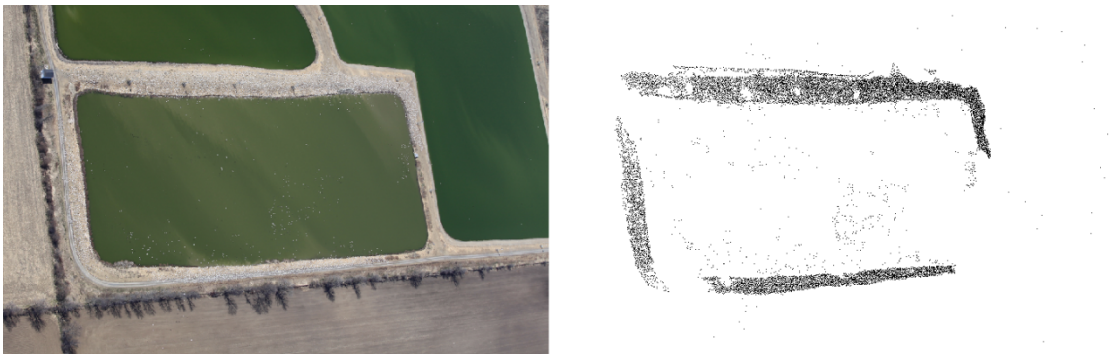

Figura 18. (Izquierda) Foto original tomada desde el aire de una población de 14558 aves posadas alrededor de un estanque de agua, formando un contorno con características similares al patrón de dispersión con forma de anillo. (Derecha) representación gráfica en R de las coordenadas de todas las aves contadas a mano. Imagen extraída del conjunto de datos Eastern Canada Flocks (ECA-Flocks, Canadian Wildlife Service) que se publicará próximamente.

5 - APÉNDICE

Código de programación:

```
############################## DISCO

library(spatstat)

Disco<-runifdisc(1000, 20)

Xdisco <- Disco$x
Ydisco <- Disco$y

Xmin<-min(Xdisco)
Ymin<-min(Ydisco)
Xmax<-max(Xdisco)
Ymax<-max(Ydisco)

Xdisco <- Xdisco - Xmin
Ydisco <- Ydisco - Ymin

coordenadas <- data.frame(Xdisco, Ydisco)
summary(coordenadas)

plot(Xdisco, Ydisco, main="A", pch=".", cex = 2, col = "black", xlab = "", ylab = "")

## fun_CE
sf<-0.1
ROInquadrats<-100
nruns <- 1024
Theta<-pi/3

source("fun_CE_individual.R")

Xmax <- max(Xdisco)
Xmin <- min(Xdisco)
Ymax <- max(Ydisco)
Ymin <- min(Ydisco)

P <- ppp(Xdisco-Xmin,Ymax-Ymin-(Ydisco-Ymin),c(0,Xmax-Xmin), c(0,Ymax-Ymin))

result<-funCE(sf,ROInquadrats,nruns,Theta,P)

############################## ANILLO

library(spatstat)

theta <- runif(1000, 0, 2*pi)
r <- rnorm(1000, 20, 1)
Xdonut <- r * sin(theta)
Ydonut <- r * cos(theta)


Xmin<-min(Xdonut)
```

```
Ymin<-min(Ydonut)
Xmax<-max(Xdonut)
Ymax<-max(Ydonut)

Xdonut <- Xdonut - Xmin
Ydonut <- Ydonut - Ymin

plot(Xdonut, Ydonut, main="A", pch=".", cex = 2, col = "black", axes = FALSE, xlab = "", ylab = "")


coordenadas <- data.frame(Xdonut, Ydonut)
summary(coordenadas)

##fun_CE
sf<-0.011
ROInquadrats<-100
nruns <- 529
Theta<-pi/3

source("fun_CE_individual.R")

Xmax <- max(Xdonut)
Xmin <- min(Xdonut)
Ymax <- max(Ydonut)
Ymin <- min(Ydonut)

P <- ppp(Xdonut-Xmin,Ymax-Ymin-(Ydonut-Ymin),c(0,Xmax-Xmin), c(0,Ymax-Ymin))

result<-funCE(sf,ROInquadrats,nruns,Theta,P)
result$npoints


############################## CUMULO

sigm1 <- 1
sigm2 <- 6
Xdisco<-c(rnorm(500, 0, sigm1), rnorm(500, 0, sigm2))
Ydisco<-c(rnorm(500, 0, sigm1), rnorm(500, 0, sigm2))

Xmin<-min(Xdisco)
Ymin<-min(Ydisco)
Xmax<-max(Xdisco)
Ymax<-max(Ydisco)


while (abs(Xmin) >= 3.5*sigm2 |  abs(Xmax) >= 3.5*sigm2 |  abs(Ymin) >= 3.5*sigm2 | abs(Ymax) >=
3.5*sigm2) {
  Xdisco<-c(rnorm(500, 0, sigm1), rnorm(500, 0, sigm2))
  Ydisco<-c(rnorm(500, 0, sigm1), rnorm(500, 0, sigm2))

  Xmin<-min(Xdisco)
  Ymin<-min(Ydisco)
  Xmax<-max(Xdisco)
```

```
 Ymax<-max(Ydisco)
}

Xdisco <- Xdisco - Xmin
Ydisco <- Ydisco - Ymin

coordenadas <- data.frame(Xdisco, Ydisco)
summary(coordenadas)

plot(coordenadas, main="Nube Disco", pch=".", cex = 2, asp = 1, col = "red")

## fun_CE
sf<-0.1
ROInquadrats<-300
nruns <- 1024
Theta<-pi/3

source("fun_CE_individual.R")

Xmax <- max(Xdisco)
Xmin <- min(Xdisco)
Ymax <- max(Ydisco)
Ymin <- min(Ydisco)

P <- ppp(Xdisco-Xmin,Ymax-Ymin-(Ydisco-Ymin),c(0,Xmax-Xmin), c(0,Ymax-Ymin))

result<-funCE(sf,ROInquadrats,nruns,Theta,P)


############################## MOSAICO

Library(spatstat)

X <- NULL
Y <- NULL
k <- 0
while(k<9){

 Disco<-runifdisc(1000, 4)

 Xdisco <- Disco$x
 Ydisco <- Disco$y

 Xmin<-min(Xdisco)
 Ymin<-min(Ydisco)
 Xmax<-max(Xdisco)
 Ymax<-max(Ydisco)

 i <- k %/% 3
 j <- k %% 3


 Xdisco <- Xdisco - Xmin + i*20
```

```
  Ydisco <- Ydisco - Ymin + j*20

  X <- c(X, Xdisco)
  Y <- c(Y, Ydisco)

  k <- k+1

}

coordenadas <- data.frame(X, Y)
colnames(coordenadas) <- c("X", "Y")
summary(coordenadas)

points(coordenadas, main="Nube Mosaico", pch=".", cex = 2, asp = 1, col = "red")


##fun_CE
sf<-0.011
ROInquadrats<-100
nruns <- 1024
Theta<-pi/3

source("fun_CE_individual.R")

Xmax <- max(X)
Xmin <- min(X)
Ymax <- max(Y)
Ymin <- min(Y)

P <- ppp(X-Xmin,Ymax-Ymin-(Y-Ymin),c(0,Xmax-Xmin), c(0,Ymax-Ymin))

result<-funCE(sf,ROInquadrats,nruns,Theta,P)
result$npoints

############################## FUNCION METODO DE MONTE CARLO

funCE <- function(sf,ROInquadrats,nruns,Theta,P){

nsx<-sqrt(nruns)
nsy<-sqrt(nruns)

xmin<-min(P$x)
ymin<-min(P$y)
xmax<-max(P$x)
ymax<-max(P$y)

originalsize<-c(xmax-xmin, ymax-ymin)
L<-sqrt(originalsize[1]*originalsize[2]/ROInquadrats)
l<-sqrt(sf*L^2)

Lx<-L
Ly<-L
Lprime_x<-l
Lprime_y<-l

#Area of fundamental box
A<-Lx*Ly
#Area of one quadrat
Aprime<-Lprime_x*Lprime_y
```

```
tau<-l/L
const<-(1-tau)^2/(3-2*tau)
alpha<-(1-tau)^2/(2-tau)/6

### DEFINE NEW IMAGE SIZE TO ADD BORDER AND MAKE GRID
imgdiag<-sqrt(originalsize[1]^2+originalsize[2]^2)
boxdiag<-sqrt(Lx^2+Ly^2)

#Number of boxes in image with border
Nx<-ceiling((imgdiag+2.*boxdiag)/Lx)
Ny<-ceiling((imgdiag+2.*boxdiag)/Ly)
Gridsize_x<-Nx*Lx
Gridsize_y<-Ny*Ly

#New imagesize
imagesize<-c(Gridsize_x,Gridsize_y)

#Margin
dd<-c((Gridsize_x-originalsize[1])/2,(Gridsize_y-originalsize[2])/2)

#Rotate points
Prot<-rotate(P,Theta, centre="midpoint")

#Translate points
Grid<-owin( c(0,Gridsize_x), c(0,Gridsize_y) )
Pnew<-ppp(Prot$x+dd[1],Prot$y+dd[2],window=Grid)

Deltax<-Lx/nsx
Deltay<-Ly/nsy

zx<-runif(1,0,Lx)
zy<-runif(1,0,Ly)

#Initialise xini, yini
xini_vec<-rep(0,nruns)
yini_vec<-rep(0,nruns)

#Initialize Variables
Nhat_per_run<-rep(0,nruns)
Q_per_run<-rep(0,nruns)
Q_per_q_run<-matrix(0,Nx*Ny,nruns)
nonzero_q_flag_per_run<-matrix(0,Nx*Ny,nruns)
nonzero_q_per_run<-rep(0,nruns)
areatothat_per_run<-rep(0,nruns)

#Var
varq1<-rep(0,nruns)
varN<-rep(0,nruns)  #interquadrat estimator
varNpoisson<-rep(0,nruns)
varCav<-rep(0,nruns)
varSplit<-rep(0,nruns)
varCavT<-rep(0,nruns)
varSplitT<-rep(0,nruns)

if(nruns ==1){
        plot(Pnew,pch=16,cex=0.4, main = " ")
        }
Q_per_q_run<-matrix(0,Nx*Ny,nruns)
k<-1

for (ii in 1:nsx)
{

    for (jj in 1:nsy)
    {

       kq<-1
        respint<-0
        respint_q<-0
        nonzero_q<-0
```

```
xini_vec[k]<-zx+(ii-1)*Deltax
yini_vec[k]<-zy+(jj-1)*Deltax
 xini<-xini_vec[k]
 yini<-yini_vec[k]
 for (i in 1:Nx)
{
     for (j in 1:Ny)
    {

         x1<-xini+(i-1)*Lx
         y1<-yini+(j-1)*Ly
         x2<-xini+Lprime_x+(i-1)*Lx
         y2<-yini+(j-1)*Ly
         x3<-xini+Lprime_x+(i-1)*Lx
         y3<-yini+Lprime_y+(j-1)*Ly
         x4<-xini+(i-1)*Lx
         y4<-yini+Lprime_y+(j-1)*Ly
        quadr<-owin( c(x1,x3), c(y1,y3) )
        if(nruns ==1){
               plot(quadr,add=T,border="red")
               }
        ### CHECK IF QUADRAT INTERSECTS REGION OF INTEREST
        overlap<-overlap.owin(Grid, quadr)
        if(overlap != 0)
        {
                  Q_per_q_run[kq,k]<- length(which (inside.owin(Pnew$x,Pnew$y,
                  quadr) == TRUE))
                  ### CHECK IF QUADRAT IS EMPTY
                  if (Q_per_q_run[kq,k] != 0)
                  {
                        nonzero_q<-nonzero_q+1
                        nonzero_q_flag_per_run[kq,k]<-1
                  }

        }
        Qi[i,k]<-Qi[i,k]+Q_per_q_run[kq,k]
         if (Q_per_q_run[kq,k] != 0)
        {
             ni[i,k]<-ni[i,k]+1
        }
         if ((j %% 2) == 0)
        {
             Qio[i,k]<-Qio[i,k]+Q_per_q_run[kq,k]
        }
         if ((j %% 2) == 1)
        {
             Qie[i,k]=Qie[i,k]+Q_per_q_run[kq,k]
        }
        ####### Transposed ##########
        tQi[j,k]<-tQi[j,k]+Q_per_q_run[kq,k]
         if (Q_per_q_run[kq,k] != 0)
        {
             tni[j,k]<-tni[j,k]+1
        }
         if ((i %% 2) == 0)
        {
             tQio[j,k]<-tQio[j,k]+Q_per_q_run[kq,k]
        }
         if ((i %% 2) == 1)
        {
             tQie[j,k]=tQie[j,k]+Q_per_q_run[kq,k]
        }



         #kq = quadrat counter
        kq<-kq+1
    }
    VarQi[i,k]<-const*(Qio[i,k]-Qie[i,k])^2
}
for (j in 1:Ny)
```

```
        {
        tVarQi[j,k]<-const*(tQio[j,k]-tQie[j,k])^2
        }
        #####
        nindex<-which(nonzero_q_flag_per_run[,k]!=0)
         nonzero_q_per_run[k]<-nonzero_q
         Q_per_run[k]<-sum(Q_per_q_run[,k])
        #####
        #NHAT
        Nhat_per_run[k]<-sum(Q_per_run[k])*A/Aprime
        #print(paste("Nhat = ", Nhat_per_run[k]))
        ######

        ## MEAN SPLIT + SPLIT TRANSPOSED
        varSplitT[k]<- 0.5*varSplitT[k] + 0.5*const*( (tQo[k]-tQe[k])^2-tnu[k] ) / tau^4
+ tnu[k] / tau^4

        #print(paste("std Cav = ", sqrt(varCav[k])))
        #print(paste("std Split = ", sqrt(varSplit[k])))
         ##DESIGN BASED: INTERQUADRAT VAR WITH ALL NONZERO QUADRATS
         varq1[k]<-var(Q_per_q_run[nindex,k])
         varN[k]<-nonzero_q_per_run[k]*varq1[k]/tau^4
        #print(paste("std Ind = ", sqrt(varN[k])))
         #####
         #POISSON
         varNpoisson[k]<-Q_per_run[k]/tau^4
k<-k+1
#k <- simulation counter
}
}

npoints<-length(P$x)

meanNhat<-mean(Q_per_run)*A/Aprime
var_Nhat<-var(Q_per_run)*A^2/Aprime^2

ce_emp<-sqrt(var_Nhat)/npoints
ce_Cav<-sqrt(mean(varCav))/npoints
ce_Split<-sqrt(mean(varSplit))/npoints
ce_CavT<-sqrt(mean(varCavT))/npoints
ce_SplitT<-sqrt(mean(varSplitT))/npoints
nonzero_q<-mean(nonzero_q_per_run)
Qmean<-mean(Q_per_run)

result <- list("ce_emp" = ce_emp,  "nonzeroq" = nonzero_q, "npoints" = npoints)

print(paste("Mean Nhat = ", meanNhat))
print(paste("CE_emp = ", ce_emp))
print(paste("CE_CavT = ", ce_CavT))
print(paste("nonzero_q = ", nonzero_q))
print(paste("Q = ", Qmean))

return(result)
}
```

6 - BIBLIOGRAFÍA

[1] A. Ancel, R. Cristofari, P. T. Fretwell, P. N. Trathan, B. Wienecke, M. Boureau, J. Morinay, S. Blanc, Y. Le Maho, C. Le Bohec, Emperors in hiding: When ice-breakers and satellites complement each other in antarctic exploration, PLOS ONE 9 (2014) 1-8

[2] H. Jacobs, To count a crowd, Columbia Journalism Review 6 (1967) 37.

[3] Lempitsky V, Zisserman A. Learning to Count Objects in Images. In: Advances in Neural Information Processing Systems; 2010. p. 1324–1332.

[4] Chabot D, Francis CM. Computer-automated bird detection and counts in high-resolution aerial images: a review. Journal of Field Ornithology. 2016;87(4):343–359.

[5] Zhang S, Benenson R, Omran M, Hosang J, Schiele B. ¿Qué tan lejos estamos de resolver la detección de peatones? En: Actas de la Conferencia IEEE sobre Visión por Computadora y Reconocimiento de Patrones; 2016. p. 1259-1267.

[6] M. Cruz, D. Gómez, L. M. Cruz-Orive, Efficient and unbiased estimation of population size, PLOS ONE 10 (2015) 1{14.

[7] M. Cruz, D. Gómez, L. M. Cruz-Orive, Efficient and unbiased estimation of population size, Enviado a PLOS ONE (2018).

[8] Gundersen HJG. Notes on the estimation of the numerical density of arbitrary profiles: the edge effect. J Microsc. 1977;111(2):219–223.

[9] Baddeley A, Rubak E, Turner R. Spatial Point Patterns: Methodology and Applications with R. CRC Press; 2015.

[10] Idrees H, Saleemi I, Seibert C, Shah M. Multi-Source Multi-Scale Counting in Extremely Dense Crowd Images. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE; 2013. p. 2547–2554.