Facultad de Ciencias

Servicio Web para el conteo de multitudes

A Crowd Counting Web Service

Trabajo de fin de grado
para acceder al

GRADO EN INGENIERÍA INFORMÁTICA

Autor: Jose Antonio Salazar Carriles
Codirector: Domingo Gómez Pérez
Codirectora: Ana Isabel Gómez Pérez

marzo de 2022

Agradecimientos

Quería agradecer este proyecto a mi familia por la paciencia que han tenido conmigo y por apoyarme en todo momento. También quería agradecer a mi amigo y compañero de carrera Alvaro Martinez por insistir en que acabara lo antes posible para poder así conseguir el Grado en Ingeniería Informática.

Servicio Web para el conteo de multitudes

Resumen

palabras clave: Análisis de imagen , Aplicación móvil, Aplicación web , Datos estadísticos, estimacion estadistica, Python, servidor web

Conocer el número total de partículas en una población, es decir, el tamaño de la población a partir de imágenes, puede ser aplicable en muchos de los problemas existentes en el mundo real. Contar personas en fotos es tedioso, difícil de comprobar y, muchas veces, innecesario. Un ejemplo se puede encontrar en esta foto. En muchas ocasiones, una estimación del número de personas y una estimación del error cometido son suficientes para cualquier propósito práctico y para ello existen diferentes métodos. Los más eficientes provienen del campo de la estereología.

En este proyecto, hemos realizado un servicio web que permite realizar una estimación del número de personas en una fotografía, mediante el conteo manual de una parte, consiguiendo una estimación lo mas precisa posible. Para ello se ha usado el lenguaje de programación Python y herramientas basadas en ese lenguaje como Kivy, Pillow y Flask. El software se puede descargar de Github.

A Crowd Counting Web Service

Abstract

keywords: Android application, Data analysis , Estimation algorithm, Image analysis , population size, Python, Web application

Knowing the total number of particles in a population, can be applicable in many of the problems existing in the real world. Counting people in photos is tedious, difficult to verify and, many times , unnecessary. An estimate of the number of people and an estimate of the error made is sufficient for any practical purpose. There are different methods for this purpose, the most efficient ones come from the field of stereology.

In this project, we have made an application where a user can upload a photo and return an estimate of the number of people in addition to other parameters. Python programming language has been choosen for the development of this service. The software can be downloaded from Github.

Índice

1. Introducción **1**
- 1.1. Contexto y antecedentes . . . 1
- 1.2. Métodos de conteo . . . 2
  - 1.2.1. Conteo mediante mediciones físicas . . . 3
  - 1.2.2. Aplicar la inteligencia artificial . . . 3
  - 1.2.3. Otras alternativas . . . 3
  - 1.2.4. El software Countem . . . 4
- 1.3. Motivación . . . 5
- 1.4. Objetivos . . . 5

2. Fundamentos Teóricos **6**
- 2.1. Muestra . . . 6
  - 2.1.1. Técnicas de Muestreo . . . 6
- 2.2. Media . . . 7
- 2.3. Mediana . . . 7
- 2.4. Estimación . . . 8
- 2.5. Varianza . . . 9
  - 2.5.1. Varianza Muestral . . . 10
  - 2.5.2. Varianza Simple . . . 10
  - 2.5.3. Varianza Compuesta . . . 11
- 2.6. Desviación Típica . . . 12
- 2.7. KdTrees . . . 12
- 2.8. Broadcasting y Multiproccesing . . . 14

3. Material Utilizado **15**
- 3.1. Tecnologías . . . 15
  - 3.1.1. Python . . . 15
  - 3.1.2. Latex . . . 17
- 3.2. Herramientas de apoyo . . . 17
  - 3.2.1. Github . . . 17
  - 3.2.2. Eclipse . . . 18
  - 3.2.3. Overleaf . . . 18
  - 3.2.4. Conda y Anaconda . . . 18
  - 3.2.5. Countem . . . 18

4. Metodología **19**
- 4.1. Metodología . . . 19
- 4.2. Planificación . . . 20

5. Análisis de Requisitos **21**
- 5.1. Requisitos Funcionales . . . 21
- 5.2. Requisitos No Funcionales . . . 22
- 5.3. Especificación de Casos de Uso . . . 22

6. Diseño e Implementación **24**
6.1. Diseño Arquitectónico . . . 24
6.2. Capa de Presentación . . . 25
6.3. Capa de Negocio . . . 29
6.4. Capa de Datos . . . 37

7. Pruebas **39**
7.1. Pruebas Unitarias . . . 39
7.1.1. Cálculo de la Varianza . . . 39
7.1.2. Diseño Interfaz (Kivy) . . . 40
7.1.3. Rellenar la matriz . . . 41
7.1.4. Extrapolar la matriz . . . 41
7.1.5. Recorte de imágenes . . . 41
7.1.6. Servidor Web . . . 42
7.1.7. Simulaciones . . . 42
7.2. Pruebas de Integración . . . 42
7.2.1. Interfaz . . . 42
7.2.2. Generador . . . 42
7.2.3. Recorte de imágenes . . . 43
7.2.4. Rellenar la matriz . . . 43
7.2.5. Extrapolar la matriz . . . 43
7.2.6. Pop up . . . 44
7.3. Pruebas de Sistema . . . 44
7.4. Pruebas de Aceptación . . . 45

8. Simulaciones **46**
8.1. Funcionamiento de las simulaciones . . . 46
8.2. Resultados y Análisis . . . 47

9. Conclusión y Trabajos Futuros **49**
9.1. Conclusiones . . . 49
9.2. Trabajos Futuros . . . 49

10. Anexos **51**
10.1. Teoría Adicional . . . 51
10.1.1. Broadcasting . . . 51
10.1.2. Multiproccesing . . . 51
10.2. Código . . . 52
10.3. Imágenes utilizadas en las simulaciones . . . 60
10.4. Tareas . . . 65

Referencias **71**

1 Introducción

Este primer capítulo expone el contexto en el que surge este proyecto, describiendo las circunstancias que motivan su realización y los objetivos que se deben satisfacer a la conclusión del mismo. La idea es ofrecer un servicio web que permita a los usuarios subir sus propias fotos y estimar el número de personas en la misma y el error con el mínimo esfuerzo y con unos resultados aproximados a la realidad.

1.1. Contexto y antecedentes

Cruz y González-Villa [2018] explican que una población es un conjunto finito de elementos separados o "partículas" de interés. Conocer el número total de partículas en una población, es decir, el tamaño de la población a partir de imágenes puede ser aplicable en muchos de los problemas existentes en el mundo real.

Ilustración 1: Imagen de una multitud. Tomada de Diario Público

En ámbitos sociológicos es importante conocer el número de personas de la población objeto de estudio como por ejemplo en manifestaciones, mítines políticos, conciertos, o maratones. Mientras que el tamaño de una manada, un rebaño, o un enjambre, por poner tres ejemplos del mundo animal, nos son de gran ayuda en ecología. Otro punto en el que es importante conocer el número de integrantes de una multitud son temas referentes a seguridad, ya que es vital tener una estimación cercana a la realidad del tamaño de una población para poder controlarla.

Contar el número de personas que hay en una fotografía es sencillo para una persona si se cuenta con una imagen de resolución suficiente en la que se pueda distinguir. Nuestro cerebro es muy bueno detectando patrones y tenemos una capacidad innata para observar las variaciones obvias entre dos personas: un niño de un metro de altura y un adulto de casi dos metros, o entre una persona con abrigo gris y una con abrigo rojo.

Desafortunadamente, es increíblemente pesado contar en fotografías si su número es del orden de decenas de miles, cientos de miles, o incluso, millones de cabezas. Al ser una tarea repetitiva el humano comete errores, llegando al caso de que se queden cabezas sin contar o alguna contadas de forma doble al no conseguir distinguirlas. Es complejo tener un resultado exacto de las personas que aparecen en una multitud, sobre todo a la hora de hablar de manifestaciones en las que cada organismo da sus respectivos datos, que, en la mayoría de los casos, suelen diferir respecto a los reales. Los medios de comunicación, los cuerpos de seguridad, las organizaciones del evento, etcétera, tienen cada uno sus propias cifras que difieren de las del resto y son de las que informan a la población interesada. La realidad es que calcular multitudes es una operación por lo general inexacta, que se traslada a los medios de comunicación en forma de intereses. Por ejemplo los organizadores de manifestaciones suelen exagerar las cifras de asistentes hasta límites inverosímiles buscando el beneficio particular. No es lo mismo decir que a una manifestación han ido diez mil personas, que decir que ha ido un millón, aunque en verdad, diferencias de este tipo sí se pueden detectar con el mero sentido común. Los cuerpos de seguridad dan las cifras oficiales de participación en manifestaciones y también tienen sus propias maneras de calcularla como podemos observar en Calle [2017]. Sin embargo, no son métodos perfectos por lo que convendría lograr un sistema para poder hacer estos cálculos con el menor número de recursos, tiempo empleado, dificultad, y el menor porcentaje de error.

Por supuesto, es importante conocer estos datos. La tecnología juega un papel importante en estos casos, ya que más allá de las manifestaciones, poder monitorizar el número total de asistentes en zonas determinadas tiene, como ya se ha comentado con anterioridad, connotaciones de seguridad. En Arabia Saudí, ante la incapacidad de mantener datos en tiempo real basándose en las cámaras, las autoridades empezaron a distribuir brazaletes electrónicos a los peregrinos para evitar estampidas. En grandes conciertos al aire libre, parques temáticos, eventos olímpicos, o centros comerciales, se puede saber con exactitud cuánta gente hay debido a que las zonas de entrada son fácilmente controladas. Una vez dentro de los recintos, o las zonas feriales, la situación se complica. La densidad de asistentes puede variar mucho entre las distintas partes del evento. Poder analizar en tiempo real los movimientos de las personas se hace clave para evitar problemas. Conseguir algoritmos que sean capaces de hacerlo con exactitud es esencial para poder reducir, o evitar, acumulaciones antes de que sucedan.

Según los estudios de comportamiento de masas en espacios públicos, densidades superiores a las seis personas por metro cuadrado entra en el terreno de lo alarmante. Para que nos hagamos una idea, implicaría meter a setecientas personas en un piso de cien metros cuadrados. El espacio quedaría muy limitado, el flujo circulatorio se atascaría y podrían originarse avalanchas y colapsos. En situaciones así, el riesgo de tragedia se dispara. Casos como los de Hillsborough, o Khodynka, ilustran lo que sucede cuando la densidad de una audiencia, o de una marcha popular, es tan alta que limita el movimiento de las personas allí presentes, derivando en avalanchas humanas o asfixias. De hecho, en Hillsborough, la densidad osciló entre las seis y las once personas por metro cuadrado.

La realidad es que no es nada fácil contar la gente que se concentra para un evento. Tal cantidad de gente, que además, en muchos casos, está en movimiento, resulta imposible de contabilizar. Nadie se imagina a veinte o treinta encargados de contar uno a uno, a pie de calle, a cada persona. Es entonces cuando se establecen diferentes herramientas para llevar a cabo el conteo de personas reunidas y es aquí donde entra la ciencia.

1.2. Métodos de conteo

Hasta mediados de los años noventa había un baremo que regía el cálculo, el cual decía que en un metro cuadrado cabían perfectamente cuatro personas. Aunque esto en teoría y físicamente es así, en realidad no lo es tanto debido a que las aglomeraciones generalmente aparecen en movimiento. En estos casos, cuatro personas caminando o moviendo los brazos no caben en un metro cuadrado, por lo que se tuvo que reajustar el método de conteo contabilizando una persona por metro cuadrado.

National Geographic [2017] explica que hay varios métodos para contar personas, generalmente en manifestaciones. A continuación se da un breve repaso a los principales métodos.

1.2.1. Conteo mediante mediciones físicas

Para calcular el número de personas se utilizan mediciones físicas, como la densidad de personas o su velocidad de desplazamiento.

Por medio del índice de densidad de manifestantes, en primer lugar, se toman fotografías desde puntos situados a gran altura, generalmente desde balcones altos en diversos puntos, o desde el propio helicóptero de la policía y después se trabaja sobre ellas. Se toman como puntos de referencia el principio y el final de la manifestación, se calcula el largo y el ancho de la calle, o lugar en el que se encuentre y finalmente se añade otro punto: el margen de error. Por ejemplo, se considera que debajo de cada árbol puede haber una persona, así como debajo de cada paraguas y se restan los metros cuadrados que pueden ocupar vehículos u otro tipo de obstáculos, además de las distancias entre personas. Posteriormente, simplemente se hacen las cuentas. Sin embargo, la estimación generalmente ignora el muestreo y se basa en una visual imprecisa.

El otro sistema, como se ha indicado anteriormente, es el de contabilizar a través de la velocidad de los manifestantes. Este método no ofrece datos tan concretos. Simplemente hay que anotar la hora a la que empezó el recorrido, la hora a la que finalizó y cuántas paradas se han hecho y cómo de largas fueron. Con todo eso se estima la velocidad media de cada persona. Si es lenta, la cantidad de gente es mayor. Por el contrario, si la velocidad es alta la cantidad de gente se presupone menor. No ofrece datos tan concretos como el de densidad de población. Además, a pesar de ser ambos modos los que casi siempre se utilizan de forma oficial, no llegan a otorgar un número fiable de personas por lo que hay que buscar otra metodología adicional.

1.2.2. Aplicar la inteligencia artificial

Se han propuesto varios métodos de inteligencia artificial para hacer conteo en el caso particular del tamaño de la multitudes en imágenes fijas (véase, por ejemplo, Botta et al. [2015]; Idrees et al. [2013]; Lempitsky y Zisserman [2010]). Funcionan bien en algunos casos particulares con patrones regulares sobre fondos homogéneos y partículas no superpuestas. Desafortunadamente, no pueden manejar el tamaño de una multitud por encima de unos pocos miles y todavía falta para llegar a los niveles de rendimiento que ofrecen los métodos antes comentados. Es bien sabido que los estimadores tienen dos fuentes de error, la varianza y el sesgo que se explicarán mas adelante. Los métodos automáticos computacionales están sesgados, ya que su error se debe a problemas sistemáticos (por ejemplo, falsos positivos y falsos negativos), que no tienen el valor esperado y provocan resultados impredecibles y generalmente pobres.

1.2.3. Otras alternativas

La sociología cuantitativa es la manera más eficaz de saber el número exacto de personas en una multitud. Se trata de una investigación que examina los datos de manera numérica. Eso es lo que hacía la empresa Lynce, que se dedicaba al conteo de personas en manifestaciones y que cerró en 2012 por falta de rentabilidad [P. Mohorte, 2017].

Durante cuatro años dicha empresa realizó conteos para los medios con precisión milimétrica. Se basaba en el método cuantitativo para establecer el número de personas en una manifestación. A través de fotografías de altísima resolución desde diversos puntos (pisos altos, avionetas, o incluso un zepelín, dependiendo del caso) que permitían ampliaciones hasta un detalle tal como para contar cabeza a cabeza; con la ayuda de un programa informático que ponía un número a cada persona que aparecía en ellas.

Los resultados eran efectivos. Su director, Juan Manuel Gutiérrez, así lo aseguraba cuando varios medios de comunicación lo entrevistaron al cierre de su empresa:

> «No nos importa en absoluto qué es lo que hay en la cabeza de una persona, sólo contamos su cabeza, y punto»

Nada más fácil ni nada más alejado de la realidad.

Tienen sentido estas palabras porque al final los datos en las manifestaciones parecen más un símbolo que una realidad. Al parecer, no había demasiada gente interesada en tener datos fiables del número de manifestantes. Posiblemente nunca los haya: la guerra de cifras siempre es política y nunca numérica [RTVE, 2012].

Todo esto no significa que la medición no sea posible. De aquellas mediciones de Lynce precisas y continuadas se pudo obtener una conclusión clara: casi ninguna concentración superó la media de 0,91 personas por metro cuadrado.

Por tanto, conocemos que la detección automática actual y los algoritmos de conteo fallan, el conteo manual exhaustivo es tedioso, lento, difícil de verificar e inviable para grandes poblaciones, los métodos en base a cálculos no son precisos y además, un sistema eficaz como este no interesa y no es rentable a la larga. Es en este punto donde surgen nuevas ideas.

1.2.4. El software Countem

Cruz et al. [2015] de la Universidad de Cantabria encontraron una solución bastante sencilla y con un margen de error aceptable en términos estadísticos. La solución era abordar el problema utilizando principios de muestreo geométrico. Estos principios son antiguos y sólidos, pero en gran parte desconocidos. Los adoptaron para estimar el tamaño de cualquier población de individuos que se encuentran en un área esencialmente plana. Por ejemplo: personas, animales, árboles en una sabana, etc.

Su diseño es imparcial independientemente del tamaño de la población, el patrón, los artefactos de perspectiva, etc. La implementación es muy simple y se basa en la superposición aleatoria de cuadrículas cuadradas. Además, propone una nueva fórmula de predicción de error teórico ya que a menudo falta una evaluación objetiva del error y se supone que los recuentos de cuadrantes son independientes. En lo que respecta a la eficiencia, los autores calcularon que contar entre cincuenta y cien individuos en veinte cuadrantes, puede producir errores estándar relativos de aproximadamente un ocho por ciento en casos típicos según los ejemplos utilizados en las pruebas. Este hecho rompe efectivamente la barrera impuesta hasta ahora por la actual falta de algoritmos automáticos de detección de rostros, ya que el muestreo semiautomático y el conteo manual se convierten en una alternativa atractiva.

El programa en sí consistía en dividir la fotografía en cuadrantes y a su vez en cada cuadrante, extraer de forma sistemática un recorte de menor tamaño (a la misma distancia y del mismo tamaño cada uno) del que se contará el número de partículas. Habría que hacer el cálculo en todos los recortes generados para obtener el resultado. Esto podía llevar unos minutos. Los únicos problemas que puede haber son: que hay que contar a mano, lo que puede generar errores humanos, y que puede que haya partículas indistinguibles por cualquier motivo (efectos de perspectiva, solapamientos, obstáculos, etc).

En un trabajo posterior Cruz y Gonzalez-Villa [2018] mejoraron el anterior sistema añadiendo unas nuevas variables que conseguían que el tamaño de los cuadrantes estuviera mejor seleccionados para un programa más eficiente de conteo. También en otra nueva mejora Cruz y González-Villa [2018] determinaron que las imágenes aéreas se formaran con un ángulo de visión perpendicular al suelo (gigapixel) beneficiando aún más este conteo. Ya sea disponiendo de estas imágenes o recalculando las imágenes en vista aérea.

Por nuestra parte, hemos probado este método y pensamos que es un buen sistema de conteo. Sin embargo, vamos a intentar ir más allá. Intentaremos que los cálculos sean más sencillos y más rápidos, pero incrementando de forma muy ligera el error medio. Posteriormente, se realizaran experimentos con el programa y se analizara cuánto más rápido se consigue una estimación precisa, es decir, cuantos recursos se pueden ahorrar antes de llegar a un error objetivo mínimo.

1.3. Motivación

Tal y como se ha comentado en el apartado anterior, han existido muchos métodos para el conteo de personas u otras partículas. Unos mejores y otros peores, unos con mayor uso y otros con menor. Sin embargo, ninguno con un grado de eficiencia suficiente para la importancia del problema que se intenta resolver.

Mis principales motivaciones personales para llevar a cabo este proyecto son la creación de una aplicación eficiente, sencilla y útil que evite la sofisticación o el uso de gran cantidad de recursos para resolver el problema que ofrece el conteo de una multitud, además dando la posibilidad de que cualquier usuario pueda disfrutar del trabajo al ser un proyecto real que no quedará guardado en un cajón, la utilización del lenguaje de programación Python, uno de los más flexibles, el cual he utilizado en estos últimos años y en el cual me gustaría profundizar, y el interés por la estadística y su relación con la informática, que es uno de los temas que más he disfrutado durante la carrera.

Además, la motivación se incrementaba sabiendo que dos de los profesores que he tenido durante el grado en Ingeniería Informática han estado involucrados en el desarrollo del primer proyecto Countem de la Universidad de Cantabria (ya explicado anteriormente) y que seguro que pueden contribuir con sus conocimientos e información, adaptando esa información para el desarrollo de la aplicación y del proyecto en sí.

Finalmente, para que conste, esta motivación personal no es solamente la de crear un sistema, sino la de ahondar en todos los conceptos que, a pesar de que ya han sido mencionados en el transcurso de la carrera, me gustaría haber podido investigar y practicar más en profundidad.

1.4. Objetivos

A nivel global, el objetivo principal es crear una aplicación eficiente que permita conseguir una estimación de las partículas que aparecen en una imagen en la que se entiende que habrá una gran cantidad de éstas, suficientemente grande para descartar el conteo manual de la totalidad de los elementos de interés y que será lo más precisa posible para el tiempo de conteo que necesitara por parte del usuario al que esté destinada.

En base al objetivo principal, se plantean los siguientes objetivos parciales:

- Estudiar el estado de las distintas tecnologías que intervienen en la creación de la aplicación.
- Diseñar una interfaz intuitiva para facilitar el uso de aplicación y la interacción de los usuarios con el entorno.
- Programar todas las funcionalidades de la aplicación.
- Analizar y optimizar la eficiencia del conteo respecto al error y el porcentaje de partículas contadas de forma manual.

2 Fundamentos Teóricos

En este capítulo se explicarán los conceptos necesarios para facilitar la comprensión de apartados posteriores del proyecto, incluidos la idea del mismo. Se dara por supuesto que se conocen las definiciones de sesgado e insesgado , y otras que aparecen en DeGroot y Schervish [2012].

2.1. Muestra

Una muestra es una parte o porción de la totalidad de un fenómeno, producto o actividad que se considera representativa del total. En estadística, la muestra es una porción extraída mediante métodos específicos que representan los resultados de una totalidad sobre la que se investiga o se hace el estudio llamada población usando la probabilidad como, por ejemplo, "la muestra estadística de 100 personas que se someten a una encuesta para conocer la satisfacción de un producto".

En nuestro caso, la población sería la totalidad de partículas que aparecen en la imagen, mientras que la muestra serían los que se seleccionan mediante la superposición de las rejillas, que sera como llamaremos a partir de ahora a los recortes de la imagen. Lo que buscamos de las muestras es que nos den un valor aproximado de cuantas personas hay en total sin tener que contar todas las partículas.

Normalmente se selecciona la muestra de una población para su estudio, debido a que estudiar a todos los elementos de una población resultaría muy extenso, poco práctico y a veces incluso imposible si la población es muy grande.

2.1.1. Técnicas de Muestreo

El muestreo es la herramienta que se encarga de seleccionar las muestras de la población. Si se emplea adecuadamente, permite obtener conclusiones específicas y por lo tanto evitar resultados sesgados. Estos estudios sirven para crear normas o directrices que permitirán tomar acciones o simplemente conocer más en detalle a la población estudiada. Existen diferentes tipos de técnicas de muestreo.

Muestreo Aleatorio

Es una técnica en la que todos los elementos tienen la misma probabilidad de ser seleccionados, por ser tomados al azar. Los tipos de muestreo aleatorio son:

- Muestreo aleatorio simple: los elementos se eligen de una lista al azar. Funciona más eficazmente cuando la población es reducida y homogénea. Sin embargo es costoso y converge peor que el resto de métodos aleatorios.
- Muestreo sistemático: el primer elemento se elige al azar y luego se escogen a intervalos constantes los elementos restantes. Este es el método que utilizaremos debido a que la primera rejilla se escoge aleatoriamente en cuanto a su posición, y el resto son réplicas en el resto de cuadrantes.
- Muestreo estratificado: se realiza dividiendo a la población en partes que respondan a características establecidas y luego se eligen aleatoriamente los individuos que se van a estudiar.

- Muestreo por conglomerado: la población se divide en grupos heterogéneos y éstos a su vez se subdividen en grupos homogéneos con características comunes para ser estudiados de acuerdo a lo requerido por el investigador.

Muestreo No Aleatorio

Se elige con base en el manejo de información de los elementos a estudiar, por lo que la representatividad de la muestra puede ser subjetiva. En este caso, se corre el riesgo de que los resultados sean sesgados.

Por poner un ejemplo, se quiere conocer de la población de una ciudad como Santander, cuantas personas son hombres y cuantas mujeres. La lógica nos sugiere que habrá aproximadamente un 50 % de hombres y un 50 % de mujeres. (Puede ser 55,45 o incluso 60,40). Sin embargo, si ese estudio se realiza por ejemplo cogiendo como muestra, las personas que asisten a un partido de fútbol, deporte generalmente de mayor interés para hombres que para mujeres, el resultado puede ser 70 % 30 % por ejemplo, lo que no es un reflejo de la realidad de la población.

Por tanto es preferible que los datos sean aleatorios para que sean útiles y representativos de la población. Dentro del muestreo aleatorio conviene no usar el muestreo aleatorio simple y usar cualquier otro que da mejores resultados a pesar de estar algo mas sesgados. Además cuando uno solo de los muestreos no es suficiente para llegar a un resultado aceptable, por ejemplo porque la población a estudiar es muy extensa, se pueden usar dos o más tipos de muestreo para el estudio.

2.2. Media

La media o promedio representa el reparto equitativo, el equilibrio, la equidad. Es el valor que tendrían los datos, si todos ellos fueran iguales. También es una medida de tendencia central. Nos sirve para resumir un conjunto de datos numéricos.

Aunque hay varios tipos de medias (ponderada, geométrica,...) nosotros nos centraremos en la media aritmética que es la que vamos a utilizar. Hay que tener en cuenta que la media aritmética es una medida sensible a la aparición de valores muy alejados de ella. Para calcularla tendremos que contar los datos que tenemos (n), sumar esos datos y por último dividir el resultado de la suma entre el número de datos.

La definición seria la siguiente :

$$\overline{X} = \frac{1}{n}\sum_{i=1}^{n} x_i = \frac{x_1 + x_2 + ... + x_n}{n} \tag{1}$$

Ejemplo 1 *Tenemos 15 alumnos en clase y tienen de menor a mayor estas notas:*
$x = (0, 2, 3, 4, 4, 5, 5, 5, 6, 7, 7, 7, 8, 9, 10)$

La suma de todas las notas daria: 82 Por tanto la media seria 82/15 que daria 5,46.

Siguiendo con el ejemplo si a esta clase le quitamos al alumno que saca el 0, la media seria de 5,85, mientras que si le quitamos al que tiene el 10, la media bajaría a 5,14 lo que significa que el cambio es importante cuando se introduce o elimina un dato en los extremos.

2.3. Mediana

Al igual que la media es un cálculo estadístico para analizar el comportamiento de un conjunto de datos. Si se ordenan todos los datos, de menor a mayor, la mediana es el valor que ocupa la posición central. Si el número de datos es par, la mediana es la media aritmética de los dos centrales.

La fórmula si el conjunto de datos es impar es:

$$M_e = x_{(n+1)/2} \tag{2}$$

mientras que si es par es:

$$M_e = (x_{\frac{n}{2}} + x_{\frac{n}{2}+1})/2 \tag{3}$$

Tiene como ventaja que solventa el problema de la media cuando llega un dato que se aleja mucho de los datos de nuestra muestra.

Ejemplo 2 *Usando el mismo ejemplo de notas empleado con la media, tenemos 15 datos, por lo tanto el número de datos es impar, y la mediana seria el número central (el de la posición 8).*
$x = (0, 2, 3, 4, 4, 5, 5, 5, 6, 7, 7, 7, 8, 9, 10)$
El resultado seria 5

Si hacemos como antes y quitamos el 0.
$x = (2, 3, 4, 4, 5, 5, 5, 6, 7, 7, 7, 8, 9, 10)$
Habría que hacer la media entre las 2 posiciones centrales, lo que nos dejaría un 5,5 de mediana.

Si le quitáramos el 10
$x = (2, 3, 4, 4, 5, 5, 5, 6, 7, 7, 7, 8, 9)$
La mediana seguiria siendo 5, por lo que no cambia absolutamente nada el valor.

2.4. Estimación

Es el procedimiento utilizado para conocer las características de un parámetro de la población, a partir de la muestra.

Existen varios tipos de estimación, pero nos vamos a centrar en la estimación puntual que consiste en la estimación del valor del parámetro mediante una sola muestra, obtenido de una fórmula determinada. Por ejemplo la media o la mediana son estimaciones puntuales.

En nuestro caso vamos a usar esta estimación para, a partir del resultado de personas que nos de la muestra (el conteo de personas en las rejillas) estimar cuantas personas habrá en la imagen completa, lo que en términos estadísticos se conoce como el método de los momentos (se igualan los momentos poblacionales a los estimadores puntuales de la muestra).

Para ello en el programa utilizaremos esta fórmula que nos da una estimación del número $\hat{N}$:

$$\hat{N} = Q \cdot \frac{T^2}{t^2} \cdot \frac{I}{i} \tag{4}$$

Donde:

- Q es el número total de personas en las rejillas analizadas.
- T es la longitud de cada cuadrante, por eso al ser el cuadrante un área, lo ponemos al cuadrado.
- t es la longitud de cada rejilla y por el mismo motivo que T está al cuadrado.
- i es el número de imágenes o rejillas ya analizadas.
- I es el número total de imágenes o rejillas a analizar.

Ejemplo 3 *Tenemos la Ilustración 2, que dividimos en cuadrantes (lineas negras), y a su vez en estos cuadrantes se colocan rejillas simétricas. Tenemos que estimar cuantos pingüinos hay en la imagen. En vez de contar todos los pingüinos, contamos los que aparecen dentro de las rejillas, y estimamos el total.*

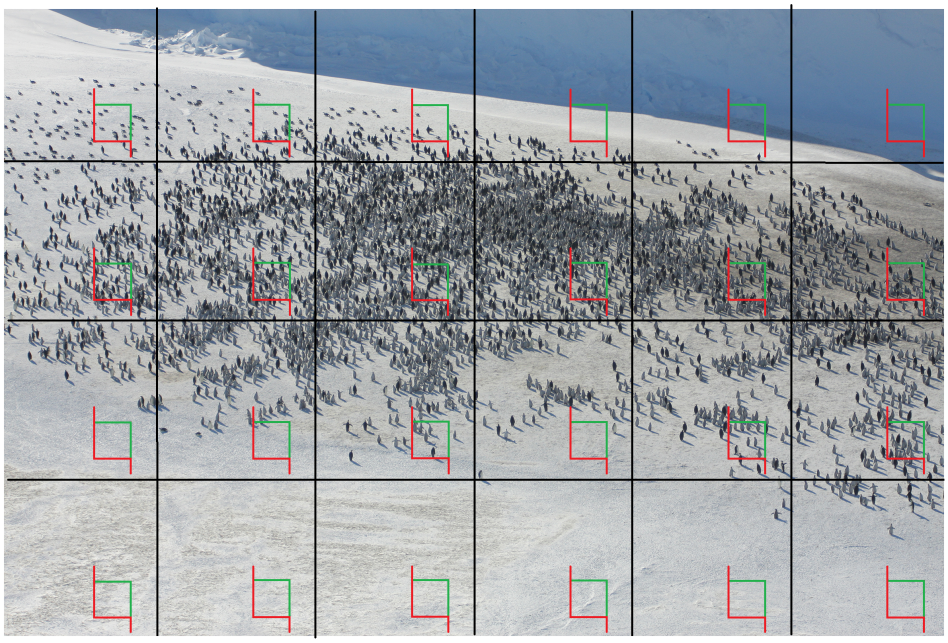

Ilustración 2: Estimación de Pingüinos. Imagen de R.Cristofari descargable desde Countem

Tras contar en todas las rejillas, los resultados son los siguientes:

$$\begin{vmatrix} 9 & 9 & 7 & 0 & 0 & 0 \\ 30 & 29 & 21 & 28 & 21 & 18 \\ 0 & 1 & 5 & 0 & 10 & 15 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{vmatrix}$$

Sabiendo que T es 400 y t es 93. La estimación tras pasar por la primera fila seria:

$$\hat{N} = 25 \cdot \frac{400^2}{93^2} \cdot \frac{24}{6} = 1849,92$$

Tras la segunda:

$$\hat{N} = 172 \cdot \frac{400^2}{93^2} \cdot \frac{24}{12} = 6363,74$$

Y finalmente:

$$\hat{N} = 203 \cdot \frac{400^2}{93^2} \cdot \frac{24}{24} = 3755,34$$

Lo más importante de un estimador, es que sea un estimador insesgado y estable en el muestreo es decir, que la varianza, que definiremos en el siguiente punto, sea lo mas pequeña posible.

2.5. Varianza

La varianza representa la variabilidad de una serie de datos respecto a su media. Formalmente se calcula como la suma de las residuos al cuadrado divididos entre el total de observaciones. Dicho sea de paso, entendemos como residuo a la diferencia entre el valor de una variable en un momento y el valor medio de toda la variable.

La unidad de medida de la varianza será siempre la unidad de medida correspondiente a los datos pero elevada al cuadrado. Por ejemplo: en los casos en que la variable mide una distancia en kilómetros, su varianza se expresa en km^2. La varianza siempre es mayor o igual que cero. A mayor valor de la varianza, mayor variabilidad. En cambio, a menor valor, más homogeneidad.

$$Var(X) = \frac{(x_1 - \overline{X})^2 + (x_2 - \overline{X})^2 + ... + (x_n - \overline{X})^2}{n} \tag{5}$$

La razón por la que los residuos se elevan al cuadrado es sencilla . Si no se hiciera, la suma de residuos sería cero debido a que los valores inferiores a la media y los superiores darían como resultado exactamente la media, y la distancia entre la suma de los valores y la media seria precisamente 0 porque seria el mismo dato [DeGroot y Schervish, 2012, Capítulo 1]. Es una propiedad de los residuos. Así que para evitarlo se elevan al cuadrado.

Se denomina varianza muestral al cálculo de la varianza de una comunidad, grupo o población en base a una muestra.

Respecto a la varianza, vamos a diferenciar 3 tipos en este proyecto, ya que a pesar de tener el mismo nombre, son distintas, se calculan de distinta forma y representan distintos datos.

2.5.1. Varianza Muestral

Es un estimador de la varianza total, que se obtiene a partir de la varianza de la muestra y tiene en cuenta su tamaño (en Python se calcula insertando el comando var). Por nuestra parte no se utilizará porque buscamos estimar la varianza total de todas las muestras y para ello utilizaremos otros estimadores. Por ejemplo, utilizar la varianza de una encuesta respecto a unas elecciones, cuando lo que se requiere es la varianza del resultado de las elecciones en si, no de una única encuesta.

2.5.2. Varianza Simple

Referido a lo último explicado, tanto la varianza simple como la compuesta representan la varianza de la población a partir de una muestra en aplicaciones de este tipo. La varianza simple asume independencia entre los cuadrantes [Cruz et al., 2015, Ecuación 2].

$$Var_{ind}(\hat{N}) = (\frac{a}{a'})^2 \cdot n \cdot \text{var}(q_1). \tag{6}$$

Donde a es el área de los cuadrantes (T^2), a' es el área de la rejilla (t^2), n es el número de cuadrantes y var(q_1) es la varianza de un cuadrado y se calcula como el sumatorio de los números, restándole la media y ese resultado poniéndolo al cuadrado y entre el número total.

Ejemplo 4 *Tenemos estos números* $x = (1, 0, 0, 0, 1, 1, 1)$
La media es 4/7
La varianza es:

$$\frac{(x_1 - \overline{X})^2 + (x_2 - \overline{X})^2 + ... + (x_7 - \overline{X})^2}{n}$$

Donde n es el número total de elementos, en este caso 7.
Entonces el cálculo seria:

$$= (1 - 4/7)^2 + (0 - 4/7)^2 + \ldots)/7 = (3/7^2 + (-4/7)^2 + \ldots)/7 =$$
$$= (9/49 * 4 + 16/49 * 3)/7 = (36/49 + 48/49)/7 = 84/49/7 = 84/343$$

2.5.3. Varianza Compuesta

Es un estimador propuesto en este proyecto. Representa lo mismo que la varianza simple, sin embargo es mucho más eficaz y exacto porque asume que hay dependencia entre cuadrantes. Las variables que debemos conocer son:

- t : Es la longitud del lado de la rejilla. Está asociado al tamaño de T, ya que $t < T$. La rejilla se toma en una posición al azar dentro del primer cuadrante que se replica sistemáticamente en el resto de cuadrantes.
- T: Corresponde a la longitud del lado de los cuadrantes en los que se toman las muestras. Las imágenes se dividen en estos cuadrantes.
- τ: Es la constante de la rejilla. Se calcula como la división de t entre T e indica que porción de la imagen estás cogiendo para el muestreo. Si τ se acerca a 1 el error sera menor, pero sera mas costoso el cálculo mientras que si τ se acerca a 0 las consecuencias serán las contrarias.
- banda: Las rejillas se identifican por una matriz con índices i,j. Las bandas denotan las columnas, identificadas como i. Por su parte las filas se identifican como j pero no son consideradas bandas.
- n: Es el número de columnas (bandas) en la imagen (cuadrantes que caben en una fila). Las que no contienen partículas se pueden omitir para agilizar el tiempo de cálculo.
- n_i : es el número de filas que se coge en la banda i que no son 0. Uno por cada fila que no esté vacía.
- q_{ij}: Es el número de elementos que están en la rejilla de la fila j de la banda i.
- Q_{oi}, Q_{ei}: Número de partículas capturadas en las filas pares e impares respectivamente de las bandas.
- Q_i: Número total de partículas en la banda i. Sumas las de Q_{oi} y las de Q_{ei}.
- Q: Número total de partículas contadas en total.

La fórmula de la varianza propuesta e implementada en el programa es [Cruz et al., 2015, Ecuación 3]:

$$Var_{Cav} = \frac{1}{6} \cdot \frac{(1-\tau)^2}{\tau^4(2-\tau)} \cdot [3(C_0 - \nu_n) - 4C_1 + C_2] + \frac{\nu_n}{\tau^4} \tag{7}$$

C_0 , C_1 y C_2 hay que calcularlos de la siguiente manera:

$$C_k = \sum_{j=1}^{n-k} Q_j Q_{j+k}, k = 0, 1, 2 \tag{8}$$

Donde:

- C_0 es Q_i^2
- C_1 es $Q_i \cdot Q_{i+1}$
- C_2 es $Q_i \cdot Q_{i+2}$

ν_n hay que obtenerlo de esta forma:

$$\nu_n = \frac{(1-\tau)^2}{3-2\tau} \cdot \sum_{i=1}^{n} (Q_{oi} - Q_{ei})^2 \tag{9}$$

2.6. Desviación Típica

Uno de los conceptos más importantes relacionados con la varianza es la desviación típica, que representa la magnitud de la dispersión de variables, y resulta muy útil en el campo de la estadística descriptiva. En realidad la desviación típica y la varianza vienen a medir lo mismo. La varianza es la desviación típica elevada al cuadrado y al revés, la desviación típica es la raíz cuadrada de la varianza.

La desviación típica se calcula para poder trabajar en las unidades de medida iniciales. Cabria preguntarse de qué sirve tener como concepto la varianza entonces. La razón es que aunque la interpretación del valor que arroja no nos da demasiada información, su cálculo es necesario para obtener el valor de otros parámetros como por ejemplo el coeficiente de error que se refiere a la desviación estándar de cada valor de la muestra y que se calcula como la desviación típica partida por la media y al contrario de la desviación típica no tiene unidades ya que es un porcentaje.

Ejemplo 5 *Si tenemos 5 personas con los siguientes pesos: 90kg, 77kg, 83kg, 66kg y 81 kg, podemos calcular su promedio sumándolos y dividiendo el resultado por la cantidad de elementos. Obtendríamos 79,4kg.*

Para conocer la varianza, deberíamos restar cada uno de los valores a la media recién evidenciada, elevar cada resultado al cuadrado, sumarlos entre sí y, finalmente, dividir todo por 5.
90-79,4=10,6 ->al cuadrado ->112,36
77-79,4=-2,4 ->al cuadrado ->5,76
83-79,4=3,6 ->al cuadrado ->12,96
66-79,4=-13,4 ->al cuadrado ->179,56
81-79,4=1,6 ->al cuadrado ->2,56

Varianza = (112,36 + 5,76 + 12,96 + 179,56 + 2,56) / 5
La varianza es 62,64 kilogramos cuadrados.

Por último, para dar con la desviación estándar, calculamos la raíz cuadrada, lo que nos deja con 7,91kg. Esto quiere decir que, en media, la diferencia entre los pesos de las distintas personas será de 7,91kg.

Estos datos resultan muy útiles y necesarios para analizar y describir información, dado que nos ofrecen distintos puntos de vista, así como diferentes tendencias de los datos que caracterizan el objeto en cuestión y permiten establecer parámetros de comparación más complejos y dinámicos que los meros valores aislados o simplemente sometidos a su promedio aritmético con lo que abren puertas a diferentes clasificaciones y a datos que pueden no haber sido considerados en un principio. Valiéndose tan sólo de la media entre un conjunto de valores, no es posible saber si alguno de ellos está excesivamente alejando de la «normalidad» existente en dicho contexto. La desviación típica permite establecer dos nuevos límites alrededor de dicha línea central, para saber cuándo un elemento es demasiado pequeño o grande.

En los siguientes apartados se van a introducir optimizaciones para implementar el método de conteo y predicción del error.

2.7. KdTrees

KdTrees es una estructura de datos que permite conocer varias características de determinados puntos en un espacio k dimensional, de manera eficiente. Como característica tenemos que todos los nodos de un árbol kd, desde el nodo raíz hasta los nodos hoja, almacenan puntos.

La idea básica es cortar dicho espacio k dimensional en mitades todo el tiempo y solo operar en uno de los espacios resultantes, para después operar en el resto (si hiciera falta). Estos cortes se harán según las

coordenadas de los puntos, lo que hace por ejemplo mas eficiente la introducción de datos en una posición concreta de un espacio , debido a que la estructura de KdTrees es en forma de árbol. Sin embargo se puede dar el caso de que 2 posiciones muy cercanas en el área, queden en distintas ramas del árbol.

Hay muchas maneras para construir un KdTree pero el sistema habitual es en el cual conforme se desciende en el árbol, se emplean ciclos a través de los ejes para seleccionar los planos. Por ejemplo, la raíz puede tener un plano alineado con el eje x, sus descendientes tendrían planos alineados con el y y los nietos de la raíz alineados de nuevo con el eje x o con el z si lo hubiera, y así sucesivamente. En cada paso, el punto seleccionado para crear el plano de corte será la mediana de los puntos que se encuentran en el árbol kd, lo que respeta sus coordenadas en el eje que está siendo usado. Este método consigue un árbol kd balanceado, donde cada nodo hoja está a la misma distancia de la raíz.

Ejemplo 6 *Tenemos una serie de puntos en unas coordenadas:*
$x = (1,9),(2,3),(4,1),(3,7),(5,4),(6,8),(7,2),(8,8),(7,9),(9,6)$

Como se puede ver en la Ilustración 3, el espacio se va dividiendo para hacer la búsqueda solo en una de las partes. Esta división se hace a partir de la mediana de la dimensión elegida. Si se coge la coordenada x, la mediana es 6 como se explico en la sección correspondiente y se divide el árbol en ese punto. Después, utilizando la otra dimensión (y) se hará el mismo proceso (hay que alternar) y se repetirá el proceso.

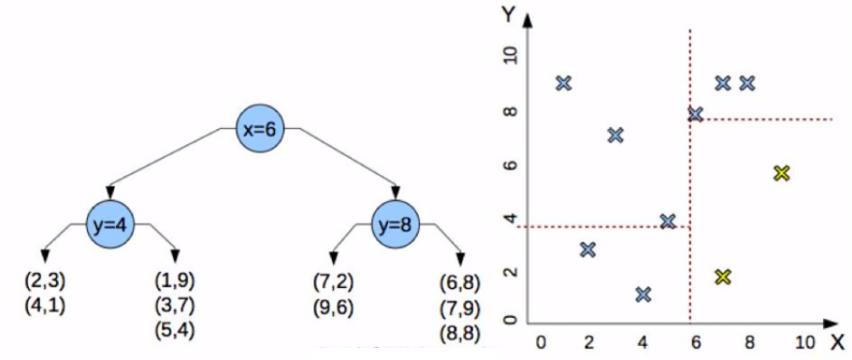

Ilustración 3: Ejemplo de KdTrees, sacado de Lavrenko [2014].

Si ahora se quiere buscar otro punto, se hará siguiendo el árbol y por tanto únicamente en una región del total del espacio. Por ejemplo si se quiere añadir el punto (7,4), se mirara el primer nodo del árbol (x=6). Como el punto tiene x=7, se irá hacia la rama derecha. Después se repetirá la operación con la coordenada y, en la cual la y del punto a añadir es menor que el nodo del nivel en el que estamos (y=8), por lo que el punto estará a la izquierda en el árbol (abajo en el eje de coordenadas) como se ve en la Ilustración 4.

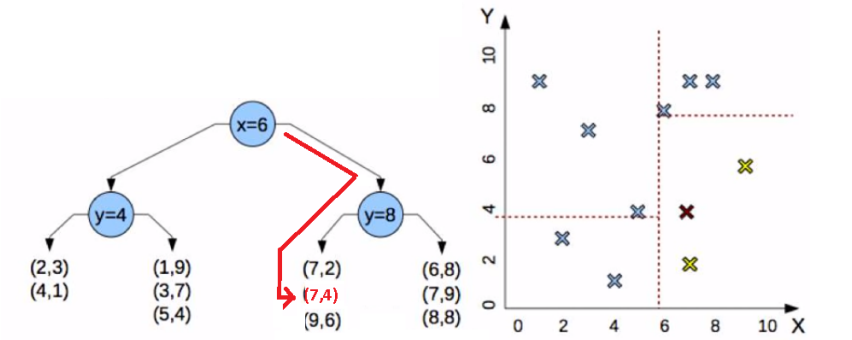

Ilustración 4: Se añade un nuevo punto

Para este proyecto implantaremos KdTrees a la hora de captar las coordenadas de un fichero o directamente de la imagen. La utilidad que nos proporciona es indicarnos si un punto está dentro de un rectángulo o no, ya que no se necesitan todos los puntos sino solo los que se sitúen dentro de la rejilla.

Además una de las formas que tiene de calcular la distancia es siguiendo el concepto de la Distancia Manhattan, que se refiere a la medición utilizando un solo eje de coordenadas a la vez (x o y), haciendo que por poner un ejemplo si un punto A es (1,2) y un punto B es (4,3), su distancia Manhattan sea igual a 4 (4-1 en el eje x) + (3-2 en el eje y). La usaremos en el programa debido a que no mide la distancia desde el centro de forma circular, sino de forma cuadrangular. Así podemos conseguir los puntos dentro del cuadrado (rejilla) rápidamente.

Ilustración 5: Distancia euclídea vs Distancia Manhattan. Imagen sacada de Grima [2017]

Empleamos Scipy que es una libreria ya implementada en Python que dispone de métodos para manejar KdTrees. Habrá que recorrer la imagen e ir agrupando las coordenadas en cuadrados para después crear la matriz real. Necesitamos 2 bucles for, uno para que recorra las filas y otro para las columnas, por lo que la complejidad del algoritmo sera de $O(n^2)$.

Mencionar que no se captaran los puntos sobre la linea inferior o izquierda de la rejilla porque cortan la imagen. Los trataremos como puntos en área restringida y se pueden observar en la Imagen 2, donde no se cuentan los pingüinos que cortan la linea roja. Esta es una regla de conteo empleada por Marcos Cruz en su proyecto ya mencionado en el Apartado 1.2.4 para evitar el conteo duplicado.

2.8. Broadcasting y Multiproccesing

No quería terminar esta sección sin mencionar dos herramientas investigadas y analizadas, pero que por distinto motivo no han sido empleadas en el proyecto. Ellas son Broadcasting y Multiprocessing que son diferentes técnicas que nos iban a permitir ser mas eficientes tanto en el desarrollo del software como en su ejecución.

Aun así se espera que puedan ser útiles en futuras etapas del proyecto. Su explicación y aplicaciones quedan recogidos en el Anexo 10.1.

3 Material Utilizado

En este capítulo se incluye una descripción de las tecnologías y herramientas empleadas en el desarrollo del proyecto.

3.1. Tecnologías

3.1.1. Python

Python es el lenguaje de programación elegido para el proyecto. Lo hemos escogido por su facilidad de uso y de entendimiento, debido a su sencilla sintaxis que cuenta con una vasta biblioteca de herramientas que van a permitir que todas las funcionalidades se implementen en el software que vamos a generar.

Una de las principales ventajas de Python es la posibilidad de crear un código con gran legibilidad, que ahorra tiempo y recursos, lo que facilita su comprensión e implementación. Para trabajar con Python hay varias interfaces gráficas (IDLE es la que empleamos en la mayoría de asignaturas en las que trabajamos con él). También se puede trabajar directamente desde la consola, ya sea de Unix o de Windows.

La elaboración del proyecto se llevará a cabo con la última versión estable del lenguaje de programación, Python 3, que sigue en continuo desarrollo y que dispone de una serie de características que no aparecen en Python 2.7 a la cual dejaron de dar soporte hace unos años.

Aparte este lenguaje lo hemos apoyado con distintas librerías que amplían sus capacidades y que pasamos a comentar.

Numpy

Numpy es una extensión de Python que agrega un mayor soporte para operar con vectores y matrices, que es para lo que se usara ya que a pesar de que Python tiene varios tipos de datos estructurados, en la práctica no son nada adecuados para cálculo numérico.

El módulo Numpy constituye una biblioteca de funciones matemáticas de alto nivel, que da acceso a una enorme cantidad de métodos y funciones aplicables a arrays, empezando por las funciones matemáticas básicas y siguiendo con muchas otras más elaboradas y que además incluyen utilidades de números aleatorios, ajuste lineal de funciones, etc ...

Kivy y el lenguaje Kv

Kivy es una biblioteca de código abierto de Python para el desarrollo de interfaces de usuario multiplataforma. Esto desde una herramienta intuitiva, orientada a generar prototipos de manera rápida y con diseños eficientes que ayudan a tener códigos reutilizables y de fácil programación. Las aplicaciones Kivy se pueden desarrollar para Linux, Windows, OS X, Android e iOS usando el mismo código base.

Se basa en OpenGL ES 2, lo que le permite que no haya variaciones dependiendo del dispositivo en el que sea ejecutada la interfaz, y soporta una gran cantidad de dispositivos de entradas, de igual manera, la he-

rramienta está equipada de una extensa biblioteca de widgets que ayudan a añadir múltiples funcionalidades.

Aunque puede funcionar directamente desde Python, Kivy tiene su propio lenguaje (Kv), para trabajar de una forma más sencilla, separando lo que es la interfaz, a lo que es el programa en sí. Sin embargo ambos lenguajes deben estar ligados y relacionan sus variables y datos.

Utilizaremos esta herramienta para crear nuestra interfaz de una manera rápida y sencilla, sabiendo que es compatible con nuestra versión de Python, y además ayudados de todo el soporte del que dispone (tutoriales, cursos, etc). Nos ayudaremos del lenguaje Kv tras analizar las posibles alternativas que incluían el lenguaje Python directamente, o editores gráficos que generaban ciertas dudas a la hora de disponer de todas las funcionalidades necesarias. Mencionar que otra librería probada que podría desempeñar el mismo papel sería Pyqt, sin embargo creemos que Kivy es más sencillo, y los resultados han sido bastante buenos.

El lenguaje Kv es un lenguaje con objetos y atributos en el que se pueden crear los botones necesarios como cargar imagen o pasar a la siguiente, los textos, por ejemplo de los resultados que deben ser mostrados al usuario, y el canvas, que servirá para pintar dentro de la aplicación, por encima de la imagen.

Pillow

Python Imaging Library (PIL) es una librería gratuita que permite la edición de imágenes directamente desde Python. Soporta una variedad de formatos, incluídos los más utilizados como GIF, JPEG y PNG. Debido a que la librería funciona solamente hasta la versión 2.7 de Python, se desarrollo Pillow, una librería de funcionalidades similares, estable y que funciona en Python 3.

Pillow, aparte de crear, abrir, y guardar imágenes, ofrece varios procedimientos estándar para su manipulación, editando bordes, modificando atributos como brillo o contraste y permitiendo agregar texto entre otras.

En este proyecto se utilizará Pillow para conseguir los recortes de la imagen principal y guardarlos en un directorio. También se empleara para cambiar la extensión de las imágenes si fuese necesario, además de para crear un collage con los recortes utilizados para el conteo.

Flask

Flask es un framework de Python que permite crear aplicaciones web de una forma muy sencilla. Vamos a emplearlo para crear el servidor web, en el que se podrán subir las imágenes, los datos y los resultados de la App.

Con Flask se pueden crear una web completa, mediante las funciones de las que dispone, incluso permite utilizar html para desarrollar esas paginas. También permite crear bases de datos, lo cual también emplearemos en este proyecto.

Matplotlib

Es una librería de Python especializada en la creación de gráficos en dos dimensiones. Los datos de entrada pueden aparecer en distintos formatos (arrays, listas). A partir de los mismos se crean los diagramas más convenientes para un buen análisis de los datos.

En nuestro caso utilizaremos este software para hacer gráficas y comparar resultados, ya que es la librería estándar para estas operaciones, simplificando el proceso como puede verse en las simulaciones de rendimiento en capítulos posteriores, diferenciando los datos en función de los valores de dichas simulaciones.

3.1.2. Latex

LaTeX es un sistema para la creación de documentos muy usado en el ámbito científico. Ha sido el utilizado en la realización de esta memoria. Su funcionamiento difiere mucho del editor de texto comúnmente utilizado, Microsoft Word. Esto es debido a que LaTeX funciona de manera similar a HTML, teniendo un conjunto de etiquetas para dar formato y un software capaz de renderizarlo.

Si bien es cierto que su curva de aprendizaje es bastante pronunciada, LaTeX cuenta con algunas ventajas importantes respecto a otras herramientas de este estilo:

- Consigue documentos de una gran calidad de imprenta, y que no la pierden por ejemplo al ser ampliados.
- Facilita la introducción de fórmulas y el uso de la notación matemática.
- Separa el estilo, del propio contenido.
- Permite dividir un proyecto en distintos ficheros, que son compilados para obtener el resultado final.
- Realiza de manera automática muchas tareas que de otro modo podrían resultar tediosas. Numerar capítulos y figuras, incluir y organizar la bibliografía adecuada, mantener índices y referencias cruzadas.
- Es un software libre.

LaTeX se ha convertido en el estándar de facto para las publicaciones científicas y las obras que contienen simbología matemática. Sin embargo tiene una serie de desventajas que lo alejan de los usuarios generales de editores de texto. No es fácil de usar y suele dar bastantes problemas para construir tablas, colocar imágenes, etc. Todo hay que hacerlo usando las etiquetas correspondientes, lo cual implica conocerlas y tener que esperar a generar el documento, para ver si se ha conseguido el resultado esperado. Para su aprendizaje he empleado el tutorial Oetiker et al. [2011], The not so short introduction to LaTex.

3.2. Herramientas de apoyo

3.2.1. Github

GitHub es una plataforma web de desarrollo de software de forma colaborativa que permite alojar proyectos utilizando el sistema de control de versiones Git. Es decir, un lugar donde guardar ficheros con código fuente en cualquier lenguaje y ponerlo a disposición de todo aquel que esté interesado.

El control de versiones (Git), gestióna los diversos cambios que se realizan sobre un repositorio. Es gratuito, open source, rápido y eficiente.

Aparte GitHub brinda herramientas muy útiles para el trabajo en equipo, dentro de un proyecto como la posibilidad de clonar repositorios ajenos (forks) o solicitar pulls.

Los forks son útiles ya sea para arreglar bugs o hacer alguna modificación. Después se puede enviar un pull al dueño del proyecto. Este podrá analizar los cambios que has realizado fácilmente, y si considera interesante la contribución, adjuntarlo al repositorio original.

En la practica Github permite añadir código o revertirlo con facilidad ya que permite crear ramas de prueba por si hay equivocaciones o para trabajar en funcionalidades nuevas sin necesidad de modificar la versión funcional. Precisamente por ello se ha usado para compartir y coordinar el código entre el director del proyecto, y el alumno, teniendo dicho código actualizado en todo momento para poder avanzar simultáneamente, y como consecuencia, con mayor rapidez en el proyecto. Tiene interfaz web, gráfica y en consola y hemos empleado las 3, quedándonos por comodidad con la gráfica en escritorio.

3.2.2. Eclipse

Eclipse es una de las mejores IDE (integrated development environment) del mercado. Su uso es muy sencillo e intuitivo e incluye herramientas de refactorización para hacer el código más legible. También incluye un comprobador de sintaxis en tiempo real que avisa si el programa falla a la hora de compilar, así como autocompletado para rellenar el código de manera automática. Además se mantiene siempre actualizado con un gran soporte y nos ofrece la posibilidad de exportar proyectos directamente a Github, donde solo habrá que sincronizar el repositorio.

Cabe destacar que este programa esta orientado a Java, pero permite programar en Python mediante un un plugin denominado Pydev. Es la IDE para Python que vamos a utilizar para realizar el programa. De hecho es uno de los mejores IDE de Python de código abierto, por lo que lo hemos escogido por delante de otras opciones testeadas como el propio IDLE que usamos en la carrera, Jupyter (otro entorno de trabajo), o la misma consola de Python.

3.2.3. Overleaf

Ya hemos explicado lo que es LaTeX y podemos instalarlo en nuestros ordenadores sin problema. Sin embargo, han comenzado a aparecer editores de texto estándar en línea para escribir informes con el fin de colaborar entre personas. Un ejemplo es Google Docs, que permite de una manera sencilla y práctica compartir y editar documentos de forma colaborativa a través de plataformas, zonas horarias, e incluso continentes. Pero, Google Docs no funciona con LaTeX. Por esta razón, en los últimos años, han aparecido nuevos editores colaborativos en línea como es el caso de Overleaf, que es un editor de Latex en linea que permite obviamente, escribir y editar documentos en grupo, pudiendo sincronizar los documentos en la nube. Funciona a la vez como editor de texto, distribución de LaTeX y visor de documentos asi que no haría falta nada más para lo que lo vamos a emplear, que es todo lo relacionado con la creación memoria, lo que implica la escritura en Latex.

3.2.4. Conda y Anaconda

Anaconda es una distribución de Python de código abierto que nos ayuda en la gestión de entornos virtuales y en la instalación de paquetes ya que contiene los paquetes más usados en temas de ciencia, matemáticas o ingeniería como pueden ser Numpy, Scipy o Matplotlib. Es imprescindible el uso de Anaconda en Windows (en Linux no tanto), por la integración de herramientas que ofrece.

Las diferentes versiones de los paquetes se administran mediante Conda, un gestor de paquetes que precisamente busca los paquetes en los repositorios. Conda fue originalmente desarrollado para solventar los problemas a la hora de gestionar los paquetes y aunque fue creado en Python, puede controlar programas creados en cualquier lenguaje. Puede facilmente instalar diferentes versiones de un paquete y sus librerias y precisamente para eso lo utilizaremos, siendo un software muy eficaz.

La mayor diferencia con Pip (el sistema clásico de instalación de paquetes en Python), es como se gestionan las dependencias, lo cual es muy importante. Pip instala todos los paquetes y dependencias requeridas, a pesar de que puedan dar problemas a otros paquetes instalados previamente. Conda analiza el estado del entorno, todo lo instalado, las versiones y sus limitaciones y se encarga de hayar la manera de instalar dependencias compatibles, negando el permiso si detecta que el software no es compatible.

3.2.5. Countem

Countem es el software de codigo abierto creado por el departamento de Matemáticas, Estadística y Computación de la Universidad de Cantabria que hemos tomado como referencia, para su mejora en este proyecto. Sirve para estimar el número de partículas (personas, cabezas, animales, objetos ...) en una imagen, de forma eficiente, insesgada y con un error pequeño. Además el programa proporciona una estimación de dicho error. El programa se puede encontrar en Unican [2016].

4 Metodología

El siguiente capítulo establece la metodología de trabajo aplicada durante el desarrollo del proyecto, estableciendo las distintas fases que lo han compuesto y las relaciones existentes entre ellas.

4.1. Metodología

La metodología explica el proceso utilizado para programar el software. En este trabajo se ha utilizado una metodología iterativa incremental. Este procedimiento se caracteriza por desarrollar el software en base de una serie de iteraciones, incluyendo funciones del sistema en cada una de ellas, lo que va a permitir dividir el proyecto en distintas etapas.

Tras cada iteración se pone en común el resultado con el cliente para comprobar que los resultados son los esperados sin la necesidad de tener finalizado el software, y así poder identificar errores o mejoras que se le puede hacer, en cada una de las etapas.

Cada iteración se compone además de las actividades reconocidas en el modelo en cascada (análisis, implementación y pruebas), realizando estas iteraciones hasta llegar a una versión final del software con la validación por ambas partes.

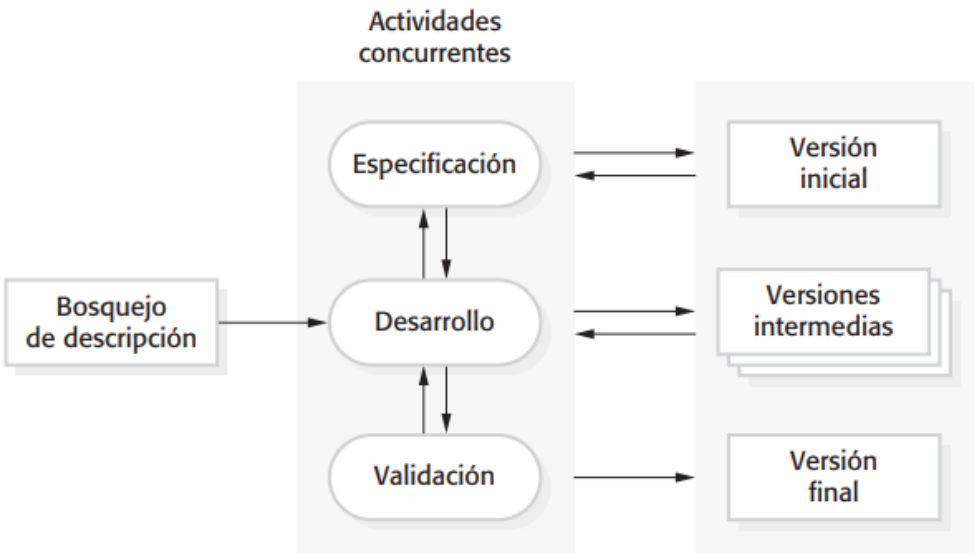

Ilustración 6: Metodología iterativa incremental. Imagen sacada de Sommerville y Campos Olguín [2011].

En el siguiente apartado se explicara más detenidamente cada una de las iteraciones empleadas en este proyecto.

4.2. Planificación

Como hemos explicado en el apartado anterior el proyecto se ha dividido en varios ciclos con cada uno de ellos teniendo varias fases como son:

- Análisis: Entender, discutir, analizar.
- Investigación: Estudiar (tutoriales), recoger información.
- Desarrollo: Programar las funcionalidades de la aplicación.
- Pruebas: Observar que el desarrollo es satisfactorio
- Cambios: Solución de errores o mejoras al sistema.
- Documentación: Centrado en documentar el proceso en la memoria.

Se muestra un diagrama en el que se observa el desarrollo del software, dividido por reuniones entre desarrolladores y cliente, identificando en que fases se concentraban antes (periodo entre reuniones) y durante la reunión.

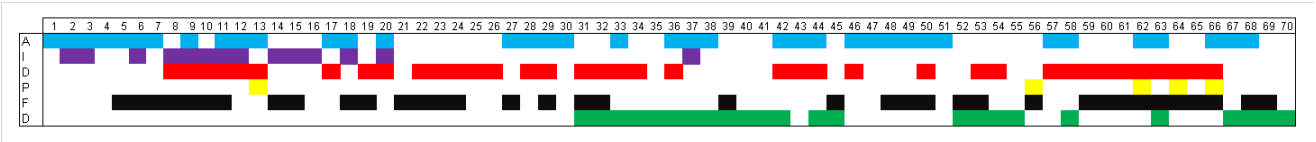

Ilustración 7: Diagrama de Gantt

El color azul es para la fase de análisis, el morado para la investigación, el rojo para el desarrollo, el amarillo para las pruebas, el negro para los cambios mientras que el verde es para la documentación.

Aclarar como se muestra en la Ilustración 7 que no en todos los ciclos se han realizado todas las fases. En las primeras iteraciones han tenido más importancia las fases de análisis e investigación (color azul y morado), centrándonos en los objetivos a cumplir, las funcionalidades requeridas, el software a utilizar, y la preparación (tutoriales e instalaciones). En las última parte del proyecto aparecen con mas frecuencia las fases de desarrollo del software, las pruebas y la documentación (colores rojo, amarillo y verde), con la fase de cambios apareciendo de forma intermitente durante el proceso (color negro).

Para un análisis mas profundo de cada una de las tareas realizadas en cada reunión, ver la Sección Tareas que se encuentra en el Anexo 10.4.

5 Análisis de Requisitos

Una vez establecidos los objetivos y descrito el contexto en que se ha desarrollado este proyecto, entramos en la fase de análisis, parte inicial de todo proyecto. Para ello se procede a establecer los requisitos, tanto funcionales como no funcionales que debería satisfacer el sistema.

5.1. Requisitos Funcionales

Tras estudiar el problema, en este apartado se acuerdan los requisitos que se deben satisfacer. Los siguientes requisitos funcionales describen las características que el sistema o algún componente del mismo han de ser capaces de cumplir. Se diferencian en requisitos de usuario que explican funciones que dependen de la iteración del usuario con el programa, y requisitos del sistema que se basan en tareas enfocadas en el sistema. En el Cuadro 1 se recogen los requisitos identificados.

Requisitos de usuario	
Identificador	Descripción
RFU01	El usuario podrá abrir fotografías, dibujos, etc
RFU02	El usuario indicara el área de las rejillas (ap) una vez se cargue la imagen.
RFU03	El usuario trabajará con la imagen de una rejilla a la vez.
RFU04	El usuario podrá salir de la aplicación en todo momento (Botón Salir).
RFU05	El usuario podrá deshacer si ha pulsado en puntos erróneos.
RFU06	El usuario podrá retroceder a la anterior rejilla y deberá volver a rellenar los recortes anteriores (Botón Volver).
RFU07	El usuario podrá guardar los resultados en un formato exportable.
Requisitos del sistema	
Identificador	Descripción
RFS01	Los recortes se ampliaran al tamaño de la pantalla para una mejor distinción de las cabezas en las imágenes.
RFS02	El programa restringirá el area de dibujo a los recortes.
RFS03	El programa permitirá pintar y contar puntos dentro del recorte.
RFS04	El programa restringirá los puntos sobre la linea inferior y sobre la linea izquierda de la rejilla que no computaran para el cálculo.
RFS05	El programa supervisara si la varianza y estimación son números positivos.
RFS06	El programa supervisara si el ag es mayor que el ap.
RFS07	El programa seleccionara aleatoriamente la posición de las rejillas, supervisando que la posición y tamaño de las rejilla permite ubicarla en un solo cuadrante.
RFS08	El sistema creara las imágenes de las rejillas en un directorio aparte.
RFS09	El sistema calculara el area de los cuadrantes (ag) en función del ancho de la imagen.
RFS10	El sistema recorrerá las rejillas de la imagen de la forma mas eficiente posible para llegar al resultado.
RFS11	El programa exportara las coordenadas que el usuario ha marcado de las fotos en un fichero txt.
RFS12	El coeficiente de error y la estimación del número de personas se actualizara y sera visible para el usuario a cada iteración de la imagen, en la interfaz.
RFS13	El programa borrara los puntos marcados de una rejilla, una vez pase a la siguiente.
RFS14	El programa finalizara al llegar a la última rejilla.

Cuadro 1: Requisitos funcionales

5.2. Requisitos No Funcionales

Establecidos los requisitos funcionales que ha de cumplir el sistema, se procede a describir los requisitos no funcionales que este ha de satisfacer. Los requisitos no funcionales (RNF) no establecen funcionalidades, sino que, describen atributos o restricciones que ha de poseer un sistema.

Los requisitos no funcionales pueden clasificarse, por su naturaleza, en varias categorías [Sommerville y Campos Olguín, 2011, figura 4.3]. Nos centraremos en los requerimientos del producto y por tanto las categorías a abarcar serán: usabilidad, rendimiento, espacio, seguridad y portabilidad. El Cuadro 2 recoge el listado completo de requisitos no funcionales identificados, categorizados según se ha comentado anteriormente.

Identificador	Descripción	Tipo	Importancia
RNF01	El usuario podrá abrir fotografías, almacenadas tanto en el sistema como disponibles en el servidor web.	Usabilidad	Alta
RNF02	El usuario podra cargar la imagen de cualquier parte del sistema.	Usabilidad	Alta
RNF03	El programa permitira cargar imagenes tipo jpg y png.	Usabilidad	Alta
RNF04	El sistema ofrecerá un buen rendimiento, cargando y respondiendo a las acciones del usuario en un tiempo mínimo.	Rendimiento	Media
RNF05	El programa implementara criterios de optimización para posibilitar estimar el mínimo posible de rejillas antes de finalizar satisfactoriamente.	Rendimiento	Alta
RNF06	La aplicación deberá ocupar el menor espacio en memoria posible.	Espacio	Media
RNF07	Los recortes se eliminaran al acabar la ejecución del programa para ahorrar espacio.	Espacio	Alta
RNF08	El sistema se podrá ejecutar en el sistema operativo Android.	Portabilidad	Media
RNF09	La aplicación podra ejecutarse sin conexión a internet para hacer los cálculos.	Portabilidad	Alta

Cuadro 2: Requisitos no funcionales

Matizar que los requisitos no funcionales de la categoría seguridad, no aparecen en este cuadro y no se implementaran en esta versión del programa, pero si se integraran en una futura versión. Serán considerados requisitos de despliegue y buscaran cumplir con el Esquema Nacional de Seguridad [CCN, 2015], en sus puntos 5.7 (Protección de la información), y 5.8.2 (Protección de servicios y aplicaciones web) con el objetivo de presentar política de seguridad a nuestra aplicación. Respecto a ello consideramos el sistema como uno con categoría de seguridad básica, ya que las cinco dimensiones de seguridad (disponibilidad, autenticidad, integridad, confidencialidad, trazabilidad) son bajas. Estas dimensiones vienen definidas en este documento de la JAE.

5.3. Especificación de Casos de Uso

Después del análisis de requisitos se procede a diseñar el diagrama de casos de uso (Ilustración 8), el cual representara el conjunto de acciones posibles por parte de los usuarios del sistema que lo utilicen. Además se añaden los casos de uso mas complicados (ilustraciones 9 y 10) para una mejor comprensión del funcionamiento de los mismos.

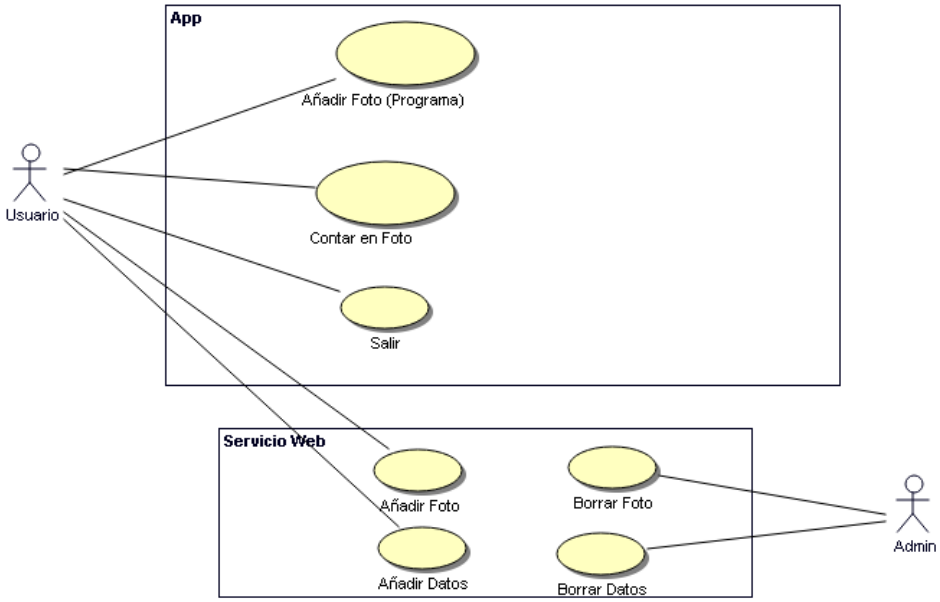

Ilustración 8: Diagrama de clases

Nombre	**Añadir Foto**
Actor Principal	Usuario
Actor Secundario	
Descripcion	El usuario escoge una foto para trabajar y se carga en el sistema
Evento de Activacion	Usuario pulsa boton Cargar
Precondición	Que no haya una foto ya cargada
Garantia Si Éxito	Se carga la foto con los atributos para las rejillas
Garantias Minimas	El usuario recibe un mensaje de éxito o de fracaso en la operación
Escenario Principal	1. El usuario pulsa el boton "Cargar" 2. El programa abre el sistema de ficheros 3. El usuario escoge una imagen 4. La imagen se carga en el programa y se muestra 5. El sistema le pide al usuario un valor de rejilla 6. El usuario indica un valor de rejilla
Extensiones	
Comentarios	

Ilustración 9: Caso de Uso - Añadir Foto

Nombre	**Contar en Foto**
Actor Principal	Usuario
Actor Secundario	
Descripcion	El usuario va contando las cabezas de las distintas imágenes
Evento de Activacion	Usuario pulsa boton Siguiente
Precondición	Que haya una imagen cargada en el sistema
Garantia Si Éxito	Obtener un resultado de estimación y varianza
Garantias Minimas	El usuario tiene los recortes
Escenario Principal	1. El usuario pulsa el boton "Siguiente" 2. El programa muestra el primer recorte de la imagen 3. El usuario cuenta en el recorte 4. El usuario le da al boton siguiente 5. El programa muestra el siguiente recorte 6. Se repiten los pasos 3,4 y 5 hasta finalizar con los recortes
Extensiones	El usuario decide salir del programa antes de recorrer todas las imágenes con una varianza cercana a la real
Comentarios	

Ilustración 10: Caso de Uso - Contar en Foto

6 Diseño e Implementación

Una vez presentado el dominio y establecidos los requisitos funcionales y no funcionales que ha de satisfacer el sistema, se procede a explicar el diseño detallado, asi como, la implementación del mismo y de cada una de las capas que lo compondrán.

6.1. Diseño Arquitectónico

Este apartado presenta el diseño arquitectónico del sistema, entrando en primer lugar en la arquitectura utilizada, para posteriormente establecer los componentes que lo conformaran, así como las interfaces que ofrecerá cada componente a los componentes de capas superiores.

La arquitectura o modelo de capas es una técnica software que separa las diferentes partes de la aplicación, con el objetivo de mejorar su rendimiento, mantenimiento y funciones. Esta separación de las diferentes partes hace que los cambios en cualquiera de las capas no afecte o afecte mínimamente al resto de capas en que está dividida la aplicación.

Es adecuada para las aplicaciones web debido a que permite establecer cada capa como un proceso separado y bien definido que puede correr en diferentes máquinas, por lo que se establece una separación funcional. Por esta razón permite además ofrecer implementaciones multiplataforma de forma más sencilla. Además según Sommerville y Campos Olguín [2011] la aproximación por capas soporta el desarrollo incremental de sistemas, ya que durante el desarrollo de la capa, algunos servicios proporcionados por esta capa puede hacerse disponible a los usuarios.

Este sistema sigue una arquitectura de tres capas, capa de presentación, capa de negocio y capa de datos. Cada capa se apoya por la capa anterior y todas se encuentran en el mismo nivel, o lo que es lo mismo, en el mismo dispositivo. A continuación, se detallan esas capas y se muestra un diagrama de componentes que describe la estructura de la aplicación. Se mostrará una descripción más detallada de los componentes en sus respectivos apartados.

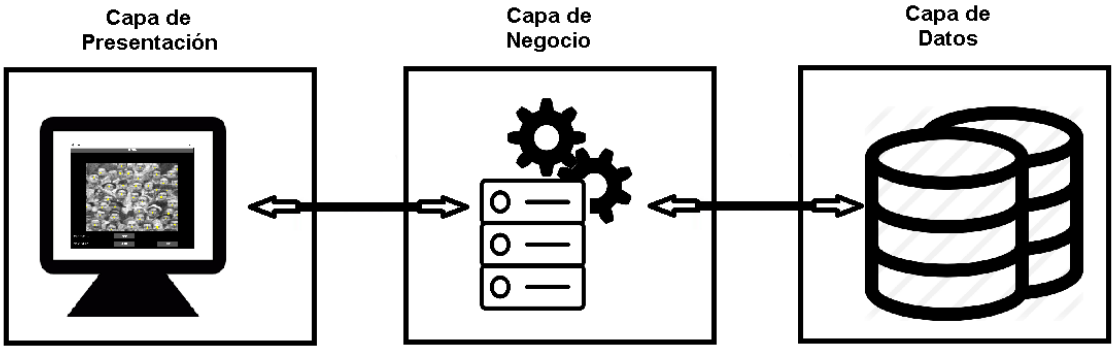

Ilustración 11: Diseño de la arquitectura de 3 capas utilizada

6.2. Capa de Presentación

La capa de presentación es la capa mas relacionada con el usuario de la aplicación, y la que utilizara directamente para realizar las distintas funciones que contiene el programa. Todo lo que el usuario puede ver, pulsar, etc, formara parte de esta capa. Así cabe destacar que la interfaz del programa, con sus valores por pantalla, sus botones, imágenes y cualquier otro objeto que aparezca en dicha interfaz.

En el caso de nuestro programa esta capa esta en su mayoría desarrollada en Kivy, que nos ha permitido encuadrar cada objeto de forma idónea en la pantalla siguiendo las exigencias del usuario.

Como es habitual, el programa tiene distintas escenas en función de la situación de ejecución en la que se encuentra. Es por eso que se ha realizado este diagrama de flujo del propio programa para explicar la manera en la que se pasa a las diferentes escenas desde el inicio hasta el final de la ejecución. Indicar que en este diagrama no aparece la parte del servidor, que comentaremos mas adelante.

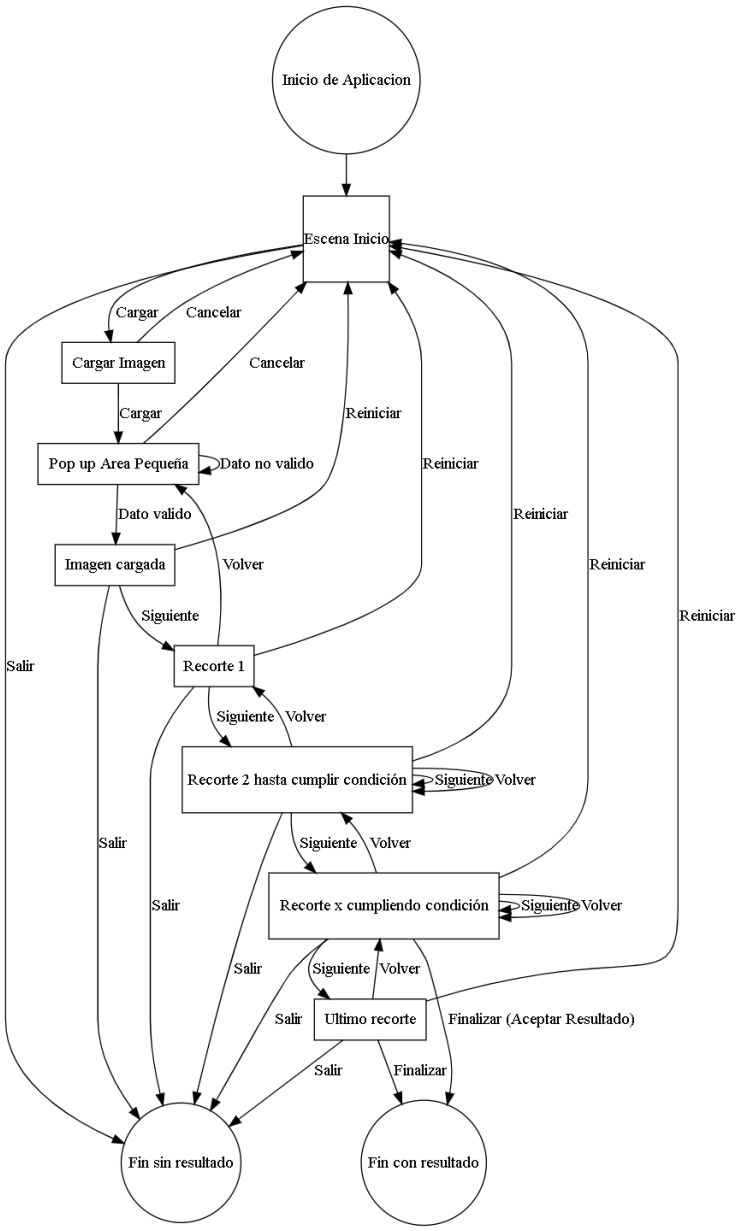

Ilustración 12: Diagrama de flujo de navegación entre escenas

Respecto al diagrama informar de que cada escena viene identificada como un rectángulo y que las flechas indican a partir de cada una de las escenas, a cual se puede pasar en función del botón que se pulse o de si los parámetros introducidos son válidos o no.

Se puede observar como se explicó en el RFU04 que todas las escenas tienen la función para salir del programa sin resultado. De la misma forma tras cargar la imagen, todas las escenas dan la opción tanto de reiniciar la aplicación, como de volver a la imagen anterior para volver a calcular los puntos como indica el RFU06. Se pasa a describir cada una de las escenas que contiene la aplicación principal en una ejecución básica.

Escena Inicio

La Escena Inicio es la que se carga en primera instancia al iniciar el programa. Sirve como introducción al mismo, para familiarizarte con la posición de los objetos aunque la mayoría estén ocultos en esta primera escena. Contiene el Botón Cargar en la parte superior que llevara a la Escena Cargar Imagen, y el Botón Salir, fijo en todas las escenas que permitirá salir de la aplicación directamente y en el que, precisamente al estar en todas las escenas y tener una función tan obvia, no es necesario profundizar. A esta escena se llega cada vez que durante la ejecución se quiere reiniciar el programa. Por otro lado se discutió añadir a esta escena un logotipo del programa en la zona central para hacerlo más ilustrativo y para darle valor comercial al producto desarrollado, pero por falta de tiempo se deja para trabajos futuros.

Escena Cargar Imagen

A esta escena se llega pulsando el Botón Cargar en la Escena Inicio, lo cual lleva a que se habrá el sistema de ficheros, donde el usuario puede buscar la imagen a utilizar en la aplicación, y darle al Botón Cargar en la parte inferior derecha para precisamente cargar la imagen a utilizar. Si se quiere retroceder a la Escena Inicio, se puede hacer también pulsando el Botón Cancelar en la parte inferior izquierda. Un ejemplo de esta escena es la Ilustración 16.

Escena Popup

Tras cargar la imagen de forma correcta, el programa automáticamente saltará a la siguiente escena que es la Escena Popup. En esta escena se abrirá un pop up de forma automática que servirá para pedir por pantalla el valor del lado del área pequeña para realizar los recortes y posteriormente los cálculos.

El usuario puede introducir un número y darle a aceptar si es el dato con el que quiere trabajar. Si el valor indicado es aceptado, el programa pasara a la Escena Imagen Cargada. Si por el contrario el valor indicado no es válido, aparecerá un mensaje de error indicando la causa del mismo y el programa volverá a iniciar esta escena para que el usuario pueda añadir un dato correcto.

En todo momento hay un Botón Cancelar en la zona inferior izquierda al igual que en la Escena Cargar Imagen por si se quiere regresar a la Escena Inicio.

Escena Imagen Cargada

Esta escena es una especie de previa a la realización del conteo en las siguientes iteraciones. Contiene la imagen elegida para el conteo en la parte central, la cual el usuario puede observar de forma detenida. A esta escena se llega tras tener seleccionada la imagen con la que trabajar, e introducir un valor válido del lado del recorte.

Las novedades de esta escena respecto a las anteriores son varias como se puede observar en el mockup de la Ilustración 13. Aparece en la posición del Botón Cargar, un Botón Reiniciar que permanecerá en la misma posición mientras el programa se encuentre en las escenas de los recortes por si se quiere volver a comenzar la ejecución del programa y cancelar todo el proceso. Se añade en la parte inferior izquierda dos

valores que son el coeficiente de error, y la estimación que aunque en esta escena no tengan un valor real, a partir de las escenas de los recortes si lo tendrán actualizado tras cada recorte. Por último se añade el Botón Siguiente que desde esta escena conduce al inicio del conteo de recortes.

Ilustración 13: Mockup de la Escena Imagen Cargada

Escena Recortes

Son las escenas en las que se realiza el conteo por parte del usuario. De hecho es la única parte del programa en la que se permite pintar, siempre y cuando se haga dentro del espacio central establecido para el recorte en cuestión como se señala en la Ilustración 21.

Lo más importante de estas escenas es la aparición en cada iteración del recorte esperado, en el mismo área en el que antes aparecía la imagen completa, y ampliado para que sea mas sencilla la visualización de las personas para el conteo.

Como he comentado en la escena anterior, en la parte inferior izquierda irá apareciendo el valor tras cada recorte del coeficiente de error esperado y de la estimación. Estos números se aproximaran con un margen de 2 decimales, los cuales consideramos suficientes para un cálculo preciso sin interferir demasiado en la posición del resto de objetos de la interfaz.

Ilustración 14: Conteo de recorte por el usuario señalizado con puntos amarillos.

Todos los recortes tienen el mismo formato, con la inclusión del Botón Volver debajo del Botón Siguiente, pero se diferencian por distintos motivos que se pasa a explicar:

- Recorte (1): El primer recorte. El Botón Volver no vuelve al anterior recorte principalmente porque no existe un recorte anterior. Por contra vuelve a la Escena Popup, por lo que vuelve a pedir el lado pequeño por si ha habido algún error por parte del usuario al pedirlo inicialmente.

- Recorte (2 - Hasta cumplir condición de salida): En caso de estos recortes todo es similar al recorte 1, pero el Botón Volver vuelve al recorte anterior.

- Recorte (1º que cumple la condición de salida - penúltimo recorte): Al cumplirse la condición de salida aparece el último botón oculto, el Botón Finalizar como se observa en la Ilustración 15 que permite salir del programa con un resultado aceptable, ahorrando el conteo en varios recortes y siendo por tanto más eficiente.

Ilustración 15: Mockup de una escena de recorte cuando cumple la condición de salida

- Último Recorte: En este caso lo que cambia es obviamente que no hay siguiente recorte, y por tanto se elimina la posibilidad de ir al siguiente recorte ocultando este botón, solo dando la opción de volver al anterior o de finalizar el conteo.

Señalar que es muy importante conocer en cada momento en que número de recorte se encuentra el programa, para evitar permitir al usuario realizar acciones que lleven a errores en la ejecución.

Una vez analizadas todas las escenas, explicar, como se ve en el diagrama de la Ilustración 12, que estas funciones pueden llevar a dos finales distintos de la aplicación, un final sin resultado en el que no tendremos datos para subir al servidor, o un final con resultado que aparecerá en un archivo txt y que si se podrá subir y relacionar en la parte del servidor, que explicaremos detalladamente mas adelante.

Indicar que en esta sección no entramos en detalle en esta parte del servidor, debido a que se encuentra en su fase inicial, realizando la función establecida, con dos botones, uno en la parte izquierda para seleccionar el archivo en tu equipo, y otro en la parte derecha para subir el archivo a la base de datos. Además en caso de la imagen se permite añadir una descripción de la misma en un recuadro de texto que aparece a la derecha del todo. Trabajaremos en una disposición estudiada más adelante.

6.3. Capa de Negocio

La segunda capa de este modelo es la capa de negocio, que es la que se utiliza para desarrollar las operaciones que realiza el programa y los métodos en los que se realizan esas operaciones, para que el programa responda a una petición del usuario en la interfaz (capa de presentación).

Así pues, pasamos a explicar las distintas operaciones que realiza el programa, entrando en profundidad en los métodos mas importantes que sirven para que una ejecución de las distintas funcionalidades se realice correctamente.

Interfaz

Al igual que en la capa de presentación, la primera operación que realiza el programa es cargar la interfaz, desde la cual se llama al resto de funciones dependiendo de las acciones que realice el usuario.

Esta interfaz esta programada en Kivy y esta formada por ”bloques de espacio”, divididos a lo largo de la pantalla, los cuales generan la capa de presentación ya explicada en el apartado anterior. Estos bloques se inician con un tamaño, y unas características iniciales. Por ejemplo el Botón Volver tiene un espacio dedicado en la interfaz. Sin embargo al ser un botón no utilizado al inicio del programa se oculta (añadiéndole opacidad), y se deshabilita, para que aunque el programa sepa que en ese espacio debe aparecer ese botón, y en la teoría esta, en la practica no lo hace para el usuario.

Otra solución sencilla desarrollada en la interfaz, fue el espacio para la imagen, ya que el programa espera una imagen en el espacio dedicado a la misma, pero al iniciar la aplicación no existe al no haber sido cargada aun por el usuario. La solución ha sido añadir al iniciar el programa un fondo negro en el espacio de la imagen, para así encuadrar bien esta zona reservada con el resto de la pantalla que también tiene este color. De todas formas la idea final como se indico en secciones anteriores es que en este espacio aparezca el logotipo de la aplicación, en el que habrá que trabajar en un futuro para darle visibilidad.

Este espacio reservado para la imagen (remarcado en rojo en la Ilustración 21) añade otro objeto dentro de si, que ocupa la misma posición en la interfaz, y que se utiliza para identificar la zona de la aplicación en la que el usuario podrá pintar. Además este objeto contiene un atributo booleano que permite en función de la escena en la que se encuentre el flujo pintar. Por ejemplo al inicializar el programa el booleano esta en False, sin embargo en las escenas de los recortes, el atributo cambia a True, y permite pintar en la zona establecida, salvo que se pulse el Botón Reiniciar, o el Botón Volver en el primer recorte, en cuyo caso el atributo se restablece a False.

Por último en este primer punto, explicar que a los botones además de reservarles un espacio en la pantalla, hay que darles una función, generalmente cuando se pulsan (on release), de esta manera se comunica esta parte diseñada en Kivy, con el programa realizado en Python, que contiene esas funciones a las que se llama al pulsar en los botones. Para mas detalle respecto a este código desarrollado en Kivy, se encuentra en el Bloque de Código 10.1.

El primero de esos botones que encontramos es el Botón Cargar, que ejecuta el Método Show_Load, el cual muestra un popup por pantalla que permite dirigirse al sistema de ficheros y escoger una imagen como se ve en la Ilustración 16 mediante un módulo llamado filechooser. Este popup también esta creado con Kivy, con el Botón Cancelar, en caso de querer salir del popup, y el Botón Cargar en caso de que el usuario encuentre la imagen con la que quiera trabajar. El Botón Cargar da paso al Método Load, que es el que inicializa todos los valores y objetos que se van a utilizar en la ejecución del programa.

Ilustración 16: Cargar Imagen

Método Load

El Método Load tiene como función principal realizar las inicializaciones previas para trabajar con una imagen, incluyendo la carga de dicha imagen y la obtención de los atributos necesarios. Es el único método que requiere entrada de datos por parte del usuario como son la ruta de la imagen y t o lado de rejilla con la que vamos a trabajar.

Esta función recoge la ruta completa de la imagen a utilizar, la cual divide para comprobar que su extensión es la que permite el programa. En caso de que la extensión sea del tipo imagen pero no concuerde con la que trabaja el programa (jpg), se transforma. En caso de no ser una imagen el programa volvería al inicio, esperando que el usuario cargue una imagen.

Una vez hecho esto, se empieza a trabajar en dejar todo preparado, creando el fichero coordenadas y el fichero resultados que serán dos archivos txt. El fichero coordenadas se encargara de guardar las coordenadas que el usuario pulse mientras que el fichero resultados es el que al acabar el programa se exporta por si el usuario quiere subirlo al servidor web. Este fichero contiene t y T, cpx y cpy (que son las coordenadas del recorte inicial), el valor del coeficiente de error, y la estimación de personas. Por último este fichero también contendrá el nombre de la imagen para poder al subirla al servidor web, comprobar si esta, y poder relacionarla como se observa a continuación.

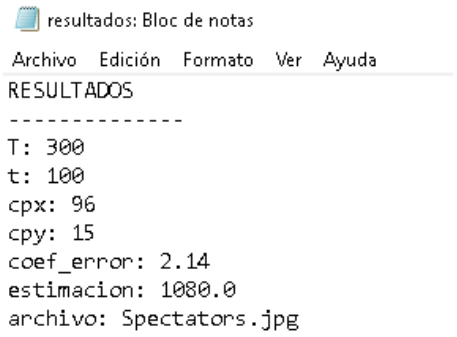

Ilustración 17: Archivo resultados.txt al terminar una ejecución

A partir de aquí se carga la imagen seleccionada en el espacio de la interfaz dedicado a ello para que se pueda observar (se le pone un identificador a este espacio en la interfaz, y modificamos la fuente utilizada en ese espacio, con la ruta de la imagen a utilizar), y se cogen los datos necesarios de la foto para procesos posteriores. Esto se hace en la función devolverancho, que se encuentra en el fichero pruebaguardarpillow5.py, al cual le pasas la ruta de la imagen y te devuelve las dimensiones x e y de la misma.

Precisamente la dimensión x de la imagen la necesitamos para calcular T, que como ya se ha explicado en anteriores apartados es el lado del cuadrante y que en el programa se calcula dividiendo la imagen en 6, siendo un valor probado y aceptable que además se redondea para que no haya decimales, los cuales pueden dar problemas. Con T ya conocida, el sistema pide el valor de t al usuario, para lo cual despliega un nuevo pop up (Escena Popup). El usuario debe escoger un valor válido para que el programa pueda avanzar desde aquí. Ese valor debe estar comprendido entre 1 y T -1 para evitar que la rejilla se salga del cuadrante.

Una vez T y t tengan valores correctos, puede terminar la inicialización del programa. Esto se hace en la función recortar_imagen localizada en pruebaguardarpillow5.py, y a la cual se le pasan t, T y la ruta de la imagen. Con ello la función realiza los recortes con los que trabajará el usuario que como se ha comentado anteriormente irán en función del ancho de la imagen.

La función recortar_imagen lo primero que hace es escoger aleatoriamente una posición de inicio de los recortes (*cpx* y *cpy*), valida (no se saldrán del cuadrante). Con la posición de inicio del recorte, y con el lado del mismo (t), utilizamos la librería Pillow para trabajar con la imagen, recortándola y almacenando los recortes resultantes en un directorio para mas tarde trabajar con ellos. Explicar que cuando se crean los recortes cada uno tiene un nombre distinto para poder después ser diferenciados, llamándose en función de la fila y la columna a la que representan de la imagen principal como se observa en la Ilustración 18, logrando una localización mas eficiente de los mismos. Por ejemplo el recorte3_5.jpg será el recorte perteneciente a la fila 3, y la columna 5 de la imagen principal, sabiendo que esas filas y columnas empiezan en 0. Además este método te devuelve el número de filas y columnas, así como el número de imágenes producidas, datos necesarios para la ejecución en el programa principal (main.py).

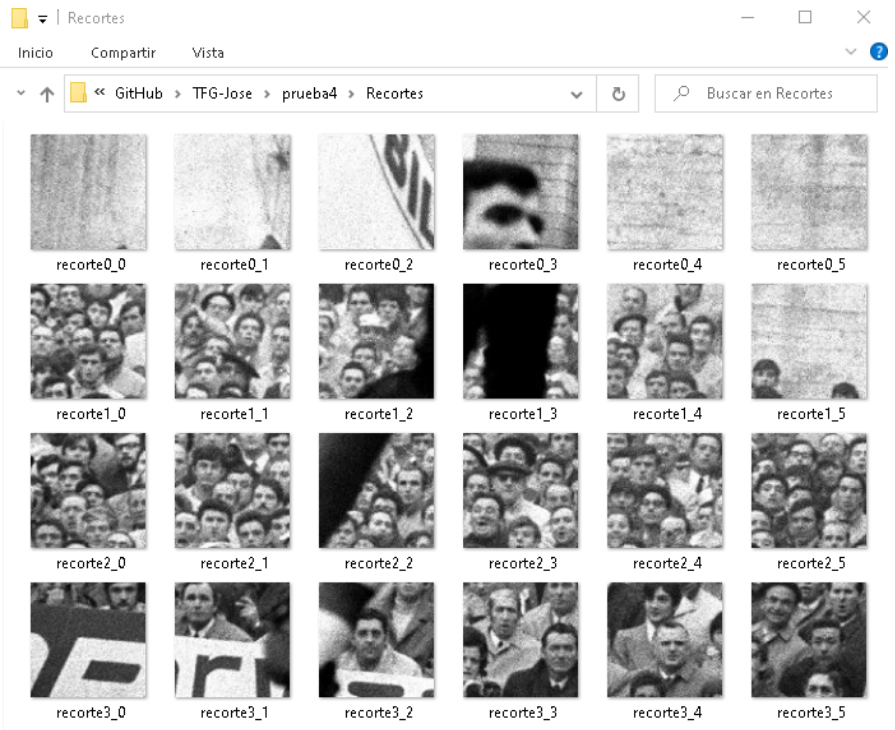

Ilustración 18: Ejemplo de almacenamiento y gestión de recortes en ejecución

Con estos datos se vuelve al Método Load, en el que se crean dos matrices de ceros con las dimensiones de la imagen y la separación de la misma. La creación de dos matrices en vez de una es debido a que la primera matriz será la que contendrá el conteo del usuario, y la segunda será creada de forma auxiliar para trabajar con ella, explicando su funcionamiento en siguientes secciones.

Por último en este método se aplicarán los cambios necesarios a los botones para que el usuario pueda continuar correctamente con el flujo del programa. Se deja referenciado el código de esta función para ver sus distintas inicializaciones en el Bloque de Código 10.2.

Método Siguiente

El principal de esos botones es el Botón Siguiente, que es el que continua con la ejecución principal del programa, y por el cual se procede a mostrar las escenas de los recortes.

Lo primero que hay que explicar sobre este método es que no muestra los recortes en un orden "lógico"(ni de izquierda a derecha, ni de arriba a abajo, etc ...), sino que hace una selección de prioridad como se muestra en el ejemplo de la Ilustración 19, de forma que permite trabajar primero con recortes alejados e independientes, por lo se obtendrán datos más relevantes de la imagen en los primeros conteos. Esto se hace así porque una de las características principales del software es permitir al usuario terminar el cálculo sin recorrer todos los recortes, en cuyo caso es preferible que queden sin comprobar recortes que en la práctica tengan mismos patrones que otros ya recorridos y por tanto sean menos relevantes para el resultado final. Una vez se llega a este método, el programa pasa a otro denominado generador, al cual se le pasan las filas y las columnas en las que esta dividida la imagen principal, y el método te genera una lista ordenada de prioridad que señala el orden en el que tienen que ser recorridos los recortes.

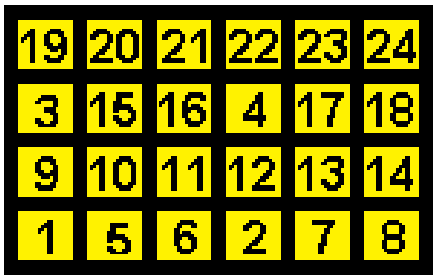

Ilustración 19: Ejemplo de recorrida de recortes. Los números indican el orden en el que se recorren, siendo el 1, el primero de todos y pudiendo quedarse los últimos sin recorrer.

Una vez con la lista y por tanto conociendo la manera en la que se recorren los recortes, existen tres posibles situaciones de la ejecución en la que se puede encontrar el programa como se señala en el apartado de la capa de presentación. Esto se comprueba con un if.

La primera opción es que el recorte sea el primero, lo cual se conoce llevando un conteo de las imágenes recorridas (si numimagenactual==1 es la primera). En este caso hay que activar las opciones de pintar en la imagen, así como activar el Botón Volver que estaba oculto. Además esta condición coincide con la siguiente que también se ejecuta en caso del primer recorte.

La segunda condición es que el recorte no sea el último (numimagenactual <numero_imagenes), en cuyo caso tiene que hacer las operaciones habituales del método. La primera es mostrar el recorte en el que estamos en pantalla. Este recorte aparece con sus dimensiones ampliadas ajustando la imagen al tamaño reservado en pantalla, lo cual podría ser un problema al desajustar esas dimensiones y por tanto las coordenadas, aunque para ello existe el Método Dimensionar que explicamos mas adelante.

Tras esto y con la matriz principal con los valores de conteo del usuario en cada zona, generamos la matriz auxiliar con posibles valores futuros de la matriz principal. Esto se hace en el Método Rellenamatriz cuyo código aparece en el Bloque de Código 10.3 y que pasamos a explicar.

A este método se le pasan como parámetros la matriz en ese momento exacto con sus valores, y las posiciones que faltan por recorrer, con las que gracias al generador que es una lista global, sabemos las posiciones de la matriz que faltan por completar. Estas posiciones se completaran con valores aleatorios en un rango desde cero, que entendemos que es el mínimo al ser imposible que haya negativos, hasta el máximo valor que contenga la matriz. Explicar que si la matriz de entrada no tiene valores o todas las posiciones tienen como valor 0, el máximo se igualara a 1 para tener un resultado.

Después como se ha indicado se rellenarán estas posiciones no recorridas y se devolverá la matriz resultante, de la cual se calculara la varianza al igual que se calculara de la matriz real para igualarlas. Este proceso se ilustra en la siguiente imagen.

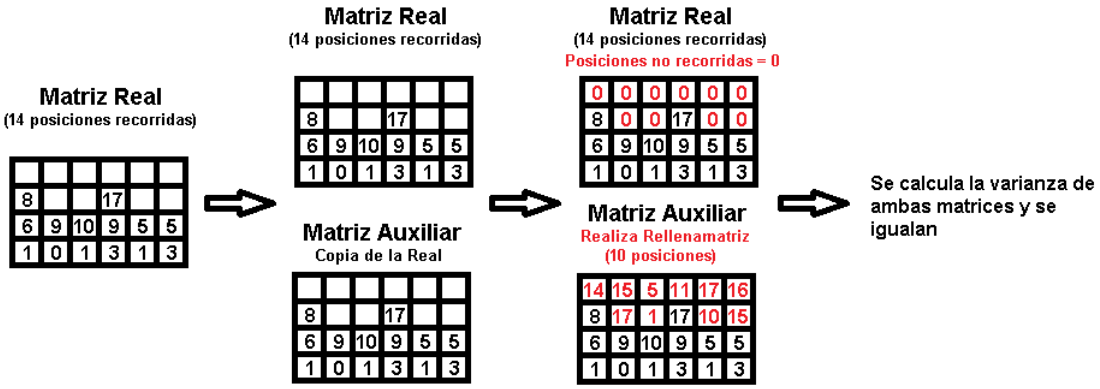

Ilustración 20: Generación y redimensión de la matriz auxiliar

La varianza se calcula en el Método Tras, que se encuentra en el fichero calculavarianzacompuesta.py, y que es llamado desde el main en cada iteración del Metodo Siguiente (cada vez que se pulsa el botón se recalcula y se muestra en pantalla). El método se denomina Tras, porque hace el cálculo de la varianza tanto de la matriz, como de su traspuesta, y con ambos resultados hace la media para una mejor aproximación de la varianza como se ve en la Ilustración 26.

Ambas matrices se envían como parámetro al Método Varc, que es el método que realmente calcula la varianza de la manera que se realiza en Cruz et al. [2015] y explicado en la Sección 2.5.3. A este método se le añade T y t, y tras el cálculo completo te devuelve el resultado. Para el cálculo de la varianza se ha utilizado la librería Numpy, que realiza las operaciones de forma mas eficiente. El código del cálculo de la varianza se encuentra en el Bloque de Código 10.4.

También se calcula la estimación de personas que debe haber en la imagen completa, en el mismo momento que se calcula la varianza. Para conocer este dato debemos coger el número total de personas contadas (la suma de los valores de la matriz), multiplicarlo por la diferencia entre T y t y por último multiplicarlo por la proporción de cuadrantes recorridos como se explica en la Fórmula 4.

Una vez con el valor de cada una de las varianzas, la de la matriz real y la de la matriz extrapolada, se comparan, y en caso de cumplir la condición, el programa da derecho a que el usuario termine, habilitando el Botón Finalizar que el usuario puede utilizar. La condición para que se llegue a esta escena es que se hayan recorrido al menos un 50 % de los recortes, y que la varianza se asemeje en un 90 % como mínimo.

En caso contrario o que el usuario decida no finalizar el programa, se mostrará en pantalla la estimación de personas y el coeficiente de error actualizados a este último recorte, y se permitirá al usuario pasar al siguiente recorte.

Si el usuario recorre todos los recortes llegara al último, momento en el que se cumplirá la condición (numimagenactual == numero_imagenes). En este momento no habrá mas imágenes, por lo que el Botón Siguiente se oculta en el programa que solo permitirá volver atrás o finalizar aunque no se cumpla la condición de salida rápida. Cuando se finaliza el programa se borran los recortes creados por el mismo para no acumular estos archivos temporales al sistema.

Otros Métodos

Guardarresultados

Este método se aplica para finalizar el programa cuando el usuario pulsa el Botón Finalizar ya sea habiendo recorrido todos los recortes, o antes si se cumple la condición. Se trata de la salida de los datos finales que se escriben en el fichero resultados.txt que se muestra en la Ilustración 17. Tras ello se procede a terminar el programa.

On_touch

Es el método que consigue que se pinte en la imagen y se guarde la coordenada en la que el usuario pulsa. Es un método vinculado al espacio reservado para la imagen, el cual captura la coordenada pulsada, y si esta dentro de ese espacio de la imagen y nos encontramos en una escena de recorte (pinta=True), al pulsar se crea una figura amarilla y circular identificando el punto exacto pulsado como se ve en la imagen.

Ilustración 21: Conteo de recorte por el usuario. Los puntos amarillos son las selecciones del usuario. El marco rojo es el área en el que el programa permite pintar.

Además, el método coge el generador y la matriz que son dos variables globales y calcula la posición real de la matriz en la que se encuentra el programa para añadir ese valor a dicha matriz como se puede observar en el Bloque de Código 10.5. Por último el método calcula la coordenada exacta seleccionada en el Método Dimensionar, explicado a continuación.

Dimensionar

El Método Dimensionar es el que realiza la transformación del punto marcado en el método anterior, con la coordenada en la que se pulsa en la aplicación, y no solo se ajusta en cuanto a tamaño real de la imagen (que se muestra agrandada), sino que también se resitua en la imagen completa. Para ello se le pasan al método las coordenadas x e y pulsadas, la posición de la imagen en pantalla, la esquina inferior escogida para el recorte (*cpx* y *cpy*), T y t y el recorte en el que estamos situados, lo que traerá también la fila y la columna de ese recorte que lo identifica en la imagen completa.

Con todos estos parámetros se calcula la coordenada real, primero de todo eliminando la parte del programa que no es el espacio dedicado a la imagen pero que si varia la posición de las coordenadas x e y. Después a la imagen se la transforma a sus proporciones reales, igualando t al largo y ancho del espacio para la imagen. En el siguiente paso se le añade el comienzo del recorte a la imagen para colocarlo en la posición exacta dentro del cuadrante. Por último hay que situarlo en el cuadrante correcto, por lo que conociendo la fila, la columna y T, lo que hay que hacer es sumar tantas veces T como filas (para x) y columnas (para y) haya desde el inicio.

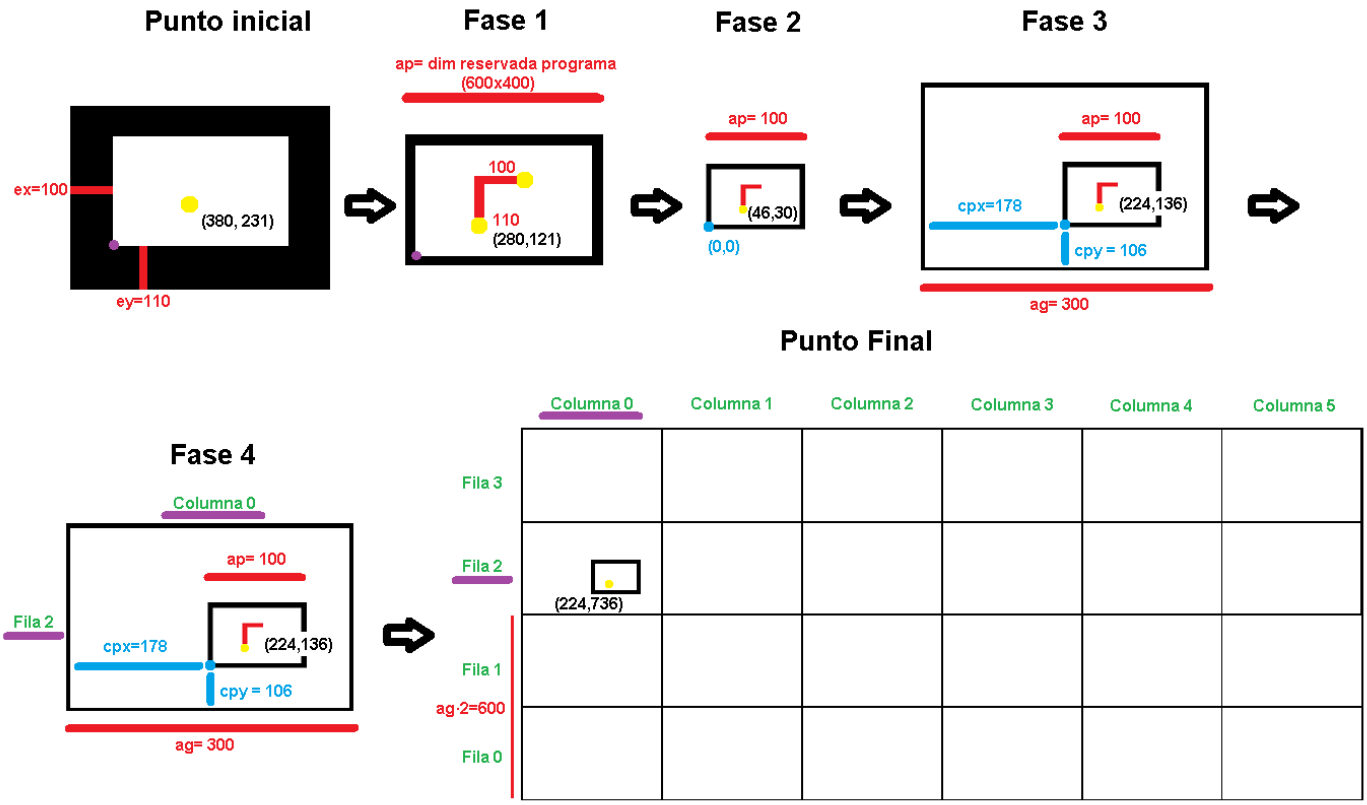

Ilustración 22: Fases de dimensionamiento de un punto marcado para llegar a su coordenada real. Punto inicial: Punto marcado en el programa. Fase 1: Eliminación del borde del programa (ex, ey). Fase 2: Dimensionar la imagen a tamaño real gracias a ap. Fase 3: Recolocar punto en el cuadrante mediante cpx y cpy. Fase 4: Ubicar el punto en la imagen completa gracias a la fila, la columna y el ag.

Como particularidad, este método no devuelve nada y lo único que hace es añadir la coordenada real al fichero coordenadas.txt. Esto es así porque es mas sencillo añadir el punto a la matriz con el contador cuando se marca (y pinta), ya que conoces el recorte en el que estás y para el conteo la posición exacta es irrelevante. El código de esta función se encuentra en el Bloque de Código 10.6.

Volver

El Método Volver es el método que consigue que se cumpla el RFU06, por si el usuario quiere volver atrás en el conteo si se ha equivocado en algún punto, o si quiere cambiar algún dato de forma sencilla. La función principal de este método es situándonos en un recorte, volver al anterior para que el usuario vuelva a contar tanto el recorte al que hemos ido, como el siguiente (en el que estábamos). Para ello, se igualan a cero ambas posiciones en la matriz, y se reduce en uno el numimagenactual, para que así el generador identifique que estamos en la posición anterior. Además gracias al generador también mostramos esa imagen anterior de igual forma que cuando avanzamos en la siguiente imagen en el Método Siguiente.

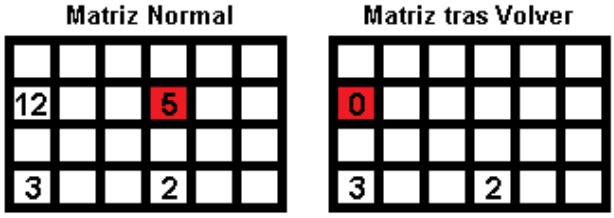

Ilustración 23: Ejemplo de funcionamiento del Método Volver con una matriz 6x4. Estando situado como indica el borde rojo, en el recorte (3,2), se eliminan los datos de ese punto y del anterior, en este caso el (0,2) según el ejemplo de la Ilustración 19 donde el usuario deberá contar de nuevo.

La complejidad de este método llega en caso de que queramos volver desde el primer recorte, en cuyo caso el método se dirige a la imagen completa y pide de nuevo t (Escena Popup), como se hizo en el Método Load. También hay que tener en cuenta el cambio de función de los botones en caso de ser necesario, por ejemplo si pasas de que se cumpla la condición a que no se cumpla (desactivar Botón Finalizar), o cuando llegas a la imagen completa, en cuyo caso tienes que desactivar el Botón Volver para no ocasionar fallo.

Reset

El último método a destacar es el Método Reset, el cual se estableció para que el programa trabajase de una forma mas fluida y dinámica, pudiendo reiniciar la aplicación desde el mismo programa si fuera necesario, en vez de salir del programa y tener que volver a entrar.

Cuando se pulsa el botón que tiene esta función, el programa lo primero que hace es reiniciar todas las variables a su estado inicial. Por tanto todas las variables globales pasan a valer cero, salvo las matrices que pasan a estar vacías. También se reinicia la ruta de la imagen y se configuran los botones para la escena inicial. Por último se borra la carpeta completa de recortes en caso de existir. Una vez hecho todo esto se detiene la interfaz y se vuelve a ejecutar para completar el reinicio de la aplicación.

Servidor Web

Por último, antes de introducir la siguiente sección, la capa de datos, quería mencionar el desarrollo del servidor Web, que tendrá mucha relación con ese próximo apartado. Este se utilizará para almacenar las distintas imágenes y datos obtenidos de las ejecuciones del programa principal. Su desarrollo ha sido realizado en Flask, un framework que permite precisamente la creación de aplicaciones web como se comento en el Capítulo 3.

En su versión inicial el servidor ha sido diseñado de la manera mas simplificada posible, sin necesidad de registro y con la opción de subir imágenes o datos. Para ello en el propio programa se restringen las extensiones permitidas a jpg, jpge, png y gif en el caso de imágenes, y a txt en el caso de los datos ya que los resultados salen del programa en ese formato. Cualquier otra extensión no es permitida. El programa comprueba en cada nueva subida la extensión del archivo, para con esa información recorrer un flujo u otro de operaciones a realizar.

Siendo más concretos los dos caminos que puede haber son subir una imagen, o subir un resultado. En ambos casos nos ayudaremos de la base de datos que incluirá el servidor y que explicaremos en la próxima sección. Si pasamos una imagen el programa comprobará si ya existe en la base de datos, lo que hace revisando si existe el mismo nombre de imagen. Si coincide el programa no permite añadir la imagen mientras que por el contrario si no existe si que se añade al servidor y pedirá una descripción de la misma. Si pasamos unos resultados la condición es la contraria. Es decir, tiene que existir en la base de datos una imagen con la extensión indicada en el txt para relacionarla. Si no existe no se permite subir ese resultado al servidor, ya que no se relaciona con ninguna imagen.

Aparte de esto el programa tiene condiciones para validar que el archivo de entrada es un archivo correcto y en caso contrario advertir con un mensaje de error, para que el usuario sepa el motivo por el cual ha fallado. Cómo hemos explicado anteriormente esta parte del sistema está directamente conectada con la base de datos utilizada y que pasamos a explicar en la siguiente sección.

6.4. Capa de Datos

La última capa de este modelo que falta por explicar es la capa de datos, que es la que explica como se almacenan estos datos que el programa recoge y después utiliza en el resto de capas.

Esta base de datos es necesaria en nuestro software porque el objetivo es almacenar las imágenes y los resultados de una forma ordenada, sin repeticiones y con relación entre las distintas instancias. Además, dado que los estimadores son insesgados merece la pena guardar los valores para mejorar la precisión y eficacia del programa.

A la hora de crear la base de datos, hemos buscado una opción tan lógica como simple, y es unificar nuestro servidor web creado en Flask, con la base de datos. Flask por defecto no tiene integrada ninguna base de datos, y es por ello por lo que mediante extensiones podemos utilizar la que mejor se adapte a nuestras necesidades. En nuestro caso hemos utilizado Flask-SQLAlchemy, que utiliza como su nombre indica lenguaje SQL, y que tiene como objetivo simplificar el uso de SQLAlchemy con Flask facilitando la realización de tareas comunes como es el caso en nuestro código, ya sea para añadir o comprobar un objeto de la base de datos.

Al ser esto así el propio programa Flask permite en el mismo código crear la instancia de la base de datos a utilizar. Esto nos da la ventaja de tener las validaciones del servidor y de la base de datos en el mismo archivo, lo que permite que la programación sea mas sencilla.

Es por ello por lo que las validaciones de los datos se realizan de una manera conjunta. En el propio servidor se comprueba que los datos tienen un formato permitido, comprobando que la extensión es válida, y una vez el resultado es satisfactorio se utiliza la base de datos para comprobar en caso de las imágenes que no hay repetición en los mismos para evitar añadir dos veces el mismo elemento, y en caso de los resultados que se pueden relacionar con una imagen existente.

Para ello la base de datos consta de dos clases como se puede observar en la Ilustración 24. La Clase Foto que guarda los datos de las imágenes, y la Clase Datos que es la que guarda los resultados del programa.

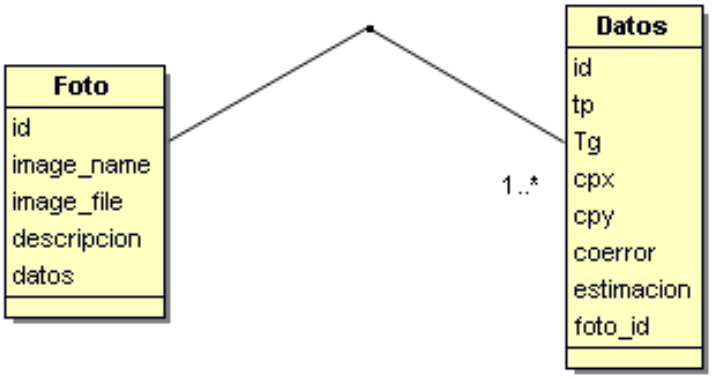

Ilustración 24: Diagrama de Base de Datos

La Clase Foto tiene los siguientes atributos:

- id: El identificador del objeto único para cualquier objeto de la base de datos, que sirve de primary key y para hacer la relación como tal entre distintas clases.
- image_name: El nombre de la imagen, que servirá para comprobar que no se guardan dos imágenes con el mismo nombre.
- image_file: El nombre del archivo que almacena la imagen, con su extensión. Este campo servirá tanto para relacionar los datos con la imagen (lo hacen mediante la ruta), como para identificar la imagen en el servidor. De hecho la imagen como tal no se almacena en la base de datos, sino que queda guardada en el servidor, y con este campo se puede acceder a ella. Obviamente este atributo y el de image_name aunque pueden ser similares, no tienen porque serlo. Por ejemplo puedes llamar a una imagen pingüinos, y que la imagen en si sea pingüinos_hielo05.jpg.
- descripcion: Una breve descripción de la imagen para que el usuario la reconozca con mayor facilidad.
- datos: El atributo que permite relacionar los datos que se vayan añadiendo a esa imagen.

Por su parte los objetos de la Clase Datos tendrán estos atributos:

- id: Al igual que en la Clase Foto, la primary key que servira como identificador en la propia base de datos.
- tp: t utilizado en la ejecución del programa.
- Tg: T utilizado en la ejecución del programa.
- cpx: La coordenada x izquierda que fija el comienzo del recorte dentro del cuadrante.
- cpy: La coordenada y inferior que fija el comienzo del recorte dentro del cuadrante.
- coerror: El resultado del coeficiente de error en la ejecución.
- estimacion: El resultado de la estimación en la ejecución.
- foto_id: El atributo que permite relacionar ese resultado con la id de una imagen (la primary key), que es como se realizan las relaciones en este tipo de base de datos.

Como se puede apreciar, los atributos de la Clase Datos son los que aparecen en los resultados de una ejecución de usuario como aparece en la Ilustración 17, salvo la ruta de la imagen, que se comprueba directamente desde el propio archivo resultados.txt en el servidor para validar el dato. El diseño de la base de datos se puede encontrar en el anexo, Bloque de Código 10.8.

7 Pruebas

En este capítulo se realizan las pruebas para validar y comprobar el funcionamiento de la aplicación, así como, su adecuación a los requisitos especificados previamente. En este proyecto se han realizado pruebas unitarias, de integración y de sistema, dejando las pruebas de aceptación para el futuro.

7.1. Pruebas Unitarias

Una prueba unitaria es una forma de comprobar el correcto funcionamiento de piezas aisladas de código, llamadas módulos.

En nuestro caso hemos realizado estas pruebas tanto en funcionalidades criticas como como el cálculo de la varianza, o el ajuste de la interfaz de la aplicación, como en otros módulos que también nos parece interesante comentar. Ahora entraremos mas detalladamente en ello.

7.1.1. Cálculo de la Varianza

El cálculo de la varianza se ha realizado en un fichero Python aparte (calculavarianzacompuesta.py), al cual hemos introducido los datos de entrada de un ejemplo [Cruz et al., 2015, Fig 1B], con sus respectivos valores de t, T, Q_{oi} y Q_{ei}.

A partir de estos datos y conociendo el resultado correcto del ejemplo, realizamos el resto de cálculos como viene estipulado en la publicación, calculando τ, C_0 , C_1 y C_2 para concluir realizando la operación final.

Una vez observado que el resultado con los datos de entrada es idéntico al indicado en el ejemplo como se puede ver en la Ilustración 25, deducimos que el cálculo de la varianza se realiza correctamente y que sus pruebas han concluido.

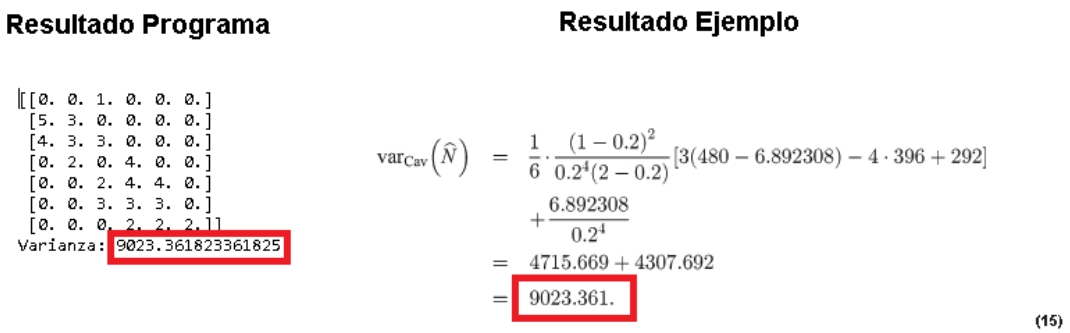

Ilustración 25: Comprobaciones de la varianza. En la parte derecha el ejemplo mencionado anteriormente [Cruz et al., 2015, Fig 1B]. En la parte izquierda el resultado del programa con los mismos datos.

Este código ha sido optimizado eliminando bucles que generaban un retardo en los cálculos y sustituyéndolos por funciones entre arrays las cuales tiene la librería Numpy explicada en los primeros capítulos de esta memoria (3.1.1 Python).

Por otro lado para mejorar el programa decidimos que además de las matrices de entrada, que llegarán para ser calculadas, calcularíamos su traspuesta (que tiene los mismos datos aunque en distinto orden) para hacer una media del resultado de forma de tener un resultado mas eficiente al coger como datos de entrada 2 muestras distintas que parten de los mismos datos.

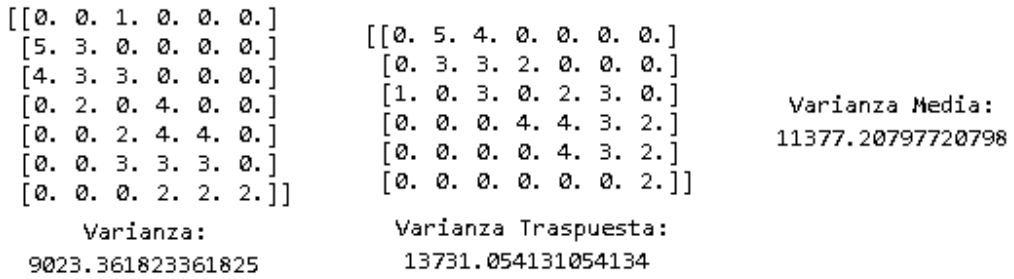

Ilustración 26: Cálculo de la varianza traspuesta siguiendo el ejemplo de [Cruz et al., 2015, Fig 1B]. A la izquierda la matriz normal con su varianza como en la Ilustración 25. En el medio la matriz traspuesta de la de la izquierda, con su valor de varianza. A la derecha la media de ambas que se entiende como un valor más exacto.

7.1.2. Diseño Interfaz (Kivy)

En este ámbito se han hecho todas las pruebas posibles, desde como pintar los puntos hasta pruebas para recolocar los distintos elementos de la interfaz como son los botones hasta la posición de las imágenes y los recortes.

En un primer momento se trabajó buscando una pantalla en negro en la que al pulsar se pintara el punto exacto donde se había pulsado. Lo siguiente que se hizo fue ajustar el tamaño de ese punto y guardar la coordenada exacta, la cual iba a ser importante para el registro de puntos en el futuro. Una vez hecho esto se añadió un espacio para las imágenes y se limito el rango para marcar coordenadas a ese espacio.

En una siguiente iteración se añadieron los botones correspondientes con los que trabajaría el usuario según figuraba en la capa de presentación de nuestro programa, sabiendo que esos botones podrían ocultarse o hacerse visibles dependiendo de en que punto se encontrase la ejecución del programa.

Todo esto se ha realizado buscando un tamaño que permitiera a su vez una buena visibilidad de todas las opciones del programa, así como no ocupar un espacio exagerado que obligara a botones o texto a salirse del rango de la aplicación.

Por último se diseñó el Método Dimensionar ya explicado en la Sección 6.3, Dimensionar. La manera de comprobar que el método funciona correctamente es yendo a los recortes que limitan la imagen principal (ya sea arriba, abajo, a la izquierda o a la derecha) y ver que coinciden con la coordenada que tienen que indicar. La coordenada de la izquierda debe ser x=0 y la coordenada de abajo debe ser y=0. Por su parte la de la derecha debe ser x=dimensión x de la imagen completa y la de arriba debe ser y= dimensión y de la imagen completa.

Este cálculo es más rápido si cogemos directamente los recortes de las esquinas opuestas para tener con 2 recortes las 4 coordenadas y poder comprobar que son correctas. Además para que la comprobación no dé problemas se necesita que el recorte contenga la esquina exacta de la imagen completa, por lo que seleccionando un t prácticamente igual que T (exactamente igual no se puede) se consigue.

7.1.3. Rellenar la matriz

La entrada de datos ha sido también un aspecto en el que hemos incidido bastante. Si que es verdad que en el programa final la entrada de datos la marca el usuario pulsando en los puntos, pero durante las pruebas y en las simulaciones la entrada de datos se hace por fichero en cuyo caso hemos tenido que almacenar cada coordenada en un formato general, para después transformar esas coordenadas en la matriz final, teniendo en cuenta el rango de los recortes. En primer lugar hicimos la prueba para una coordenada para después desarrollar el programa para que pudieras situar todas las coordenadas en sus posiciones. Esta segunda parte si que la hemos hecho en ambos programas mediante KdTrees, el cual es un sistema que nos permite conocer si un punto está en un recorte o no.

Esta primera versión de la matriz y sus puntos se fue retocando, por ejemplo eliminando los puntos encima de la coordenada x=0 e y=0 (corte izquierdo y corte inferior) para el cumplimineto del RFS04. Estos puntos no se tienen en cuenta, como se ha mencionado, siguiendo la regla de conteo determinada por la rejilla (puntos que intersecan con la línea roja) para evitar conteos dobles u otros efectos innecesarios. Esto se probó dejando el recorte con una posición y tamaño fijo sabiendo que en ese rango había un punto que se situaba en esta zona restringida.

7.1.4. Extrapolar la matriz

Tras tener varios valores en cada matriz, la idea era extrapolar los valores del resto de recortes para comprobar en que situación era viable terminar el conteo con un margen de error aceptable.

Para ello se extrapolaron los datos de las posiciones que no se habían recorrido como se explica y se muestra en la Ilustración 20. En un primer momento la idea para identificar las posiciones no recorridas era inicializar la matriz a -1, e ir cambiando el valor a la hora de entrar en cada recorte. Así, una vez había que reconocer las posiciones vacías, se hacia cuando se llegaba a una posición con valor -1. Sin embargo este mecanismo daba algún problema tanto en el conteo como en el resultado final y además era complejo, por lo que cambiamos el método, pasándole tanto el generador (y por tanto el orden en el que se debían recorrer las posiciones), como el número de recorte en el que estábamos.

De esta manera el programa es más sencillo y limpio ya que tenemos identificada la posición en la que estamos, las imágenes que quedan por recorrer y su orden, por tanto conocemos las posiciones de la matriz que están vacías y a las que hay que ponerle un valor para hacer el cálculo.

7.1.5. Recorte de imágenes

Para las pruebas de los recortes empezamos con una imagen, y utilizando un valor por defecto de T (o lado del cuadrante), para a partir de ahí realizar los recortes necesarios. La dificultad estaba en la creación del directorio en el que había que ubicar los recortes, y el renombre de los mismos ya que cada uno tenia distinto nombre. Además había que lograr también encuadrar los recortes para evitar problemas de precisión en el conteo. Una vez logrado este objetivo se repitió el mismo método pero con las rejillas, para lo que el mecanismo era el mismo pero con el uso la variable t.

7.1.6. Servidor Web

Se han hecho pruebas de los distintos flujos que el programa puede soportar hasta dejarlo en la versión inicial operativa actual. Esos flujos difieren en los datos de entrada ya que el programa actua de forma distinta si subes una imagen o unos datos de conteo, y dependiendo también de si la imagen esta ya en la base de datos o no, ya que en este caso se podían cruzar datos de estimaciones anteriores.

Los problemas encontrados han sido menores en este caso y más debidos a la inexperiencia con Flask y a la separación en capas de la aplicación como por ejemplo inconsistencia al nombrar las variables t y T, en las que el programa no diferenciaba mayúsculas de minúsculas, cambiando esos nombres por tp y Tg de la Clase Datos.

7.1.7. Simulaciones

Antes de nada explicar que al no ser este módulo parte del flujo del programa, sino un punto aparte con el mismo sistema de cálculo de varianza pero distinto modelo de entrada de datos, no necesita pruebas de integración como el resto de módulos sino que es único y solo requiere de pruebas unitarias. Su origen es el programa real, del cual hemos tenido que eliminar todas las funcionalidades aplicables al usuario (interfaz, conteo, etc…) y dejar solamente los cálculos a los cuales ha habido que añadir unos datasets de entrada explicados en el Capítulo 8.

7.2. Pruebas de Integración

El objetivo de las pruebas de integración es, una vez verificados los módulos unitarios por separado, ampliar el rango de alcance de las pruebas. De forma que se compruebe el comportamiento de dichos módulos, a la vez que se van integrando en el proyecto en cada iteración.

La mayoría de módulos se han importado en modo de función para que una vez con los datos en el programa solo hiciera falta comprobar su correcta integración.

Explicar que al estar la parte del servidor web, y la parte del programa en si diferenciadas, se han hecho pruebas mínimas de integración, debido a las pruebas más intensivas realizadas tanto en la parte del servidor web como en la lógica de negocio (programa).

7.2.1. Interfaz

Aunque aparezca en las pruebas unitarias, hay que explicar que la interfaz se ha ido integrando al programa iteración a iteración debido a que las funcionalidades del lenguaje Kivy y de los distintos métodos se complementaban. Así se empezó por la ejecución de la propia interfaz, construyendo los espacios con el objetivo de una óptima usabilidad de la interfaz, pasando a las distintas funciones como puede ser cargar las imágenes, contar los puntos en cada imagen con un contador, borrar dichos puntos al pasar a la siguiente imagen y por último realizar las funcionalidades de cada uno de los botones.

Un ejemplo de mejoras a las interfaz realizadas en las últimas iteraciones es la función de reiniciar, que ejecuta los comandos para reiniciar la aplicación y al mismo tiempo la interfaz también. Esto transponía en las primeras fases de prueba ciertos objetos, lo cual implicó trabajar con los directorios y con las variables para dejarlo en un estado inicial correcto como se puede observar en el Bloque de Código 10.7.

7.2.2. Generador

El generador es una parte fundamental de la eficiencia del programa como ya se ha hablado en el Apartado 6.3, Capa de Negocio. Explicar que antes de llegar a la manera final de recorrido de recortes se discutieron otras soluciones que según que imágenes podían ser menos eficientes o comportarse de una peor manera. Por ejemplo se barajó eliminar los recortes más extremos para transformar la imagen en

un cuadrado que se ejecutara siempre de igual forma. Esta opción era interesante pero podías descartar cuadrantes muy relevantes de la imagen en caso de que las partículas estuvieran en la zona exterior de la imagen. También se planteo un generador de puntos recurrente, el cual fue descartado por su complejidad.

Finalmente tras quedarnos con nuestro generador, y tras hacer las pruebas unitarias con un ejemplo de funcionamiento correcto se buscó añadir el módulo al programa, para el cual hubo que utilizar cada punto de la lista y buscar el recorte necesario para que se mostrara en pantalla y así el usuario pudiera trabajar con el.

No solo eso, al probar con nuevas imágenes de distintas dimensiones nos dimos cuenta de que ciertas pruebas no daban los resultados esperados. Esto era porque había un error a la hora de ajustar las filas y columnas que debía recorrer el generador tras observarlo detenidamente con el debug que ofrece Eclipse. El generador no actuaba bien si las filas y las columnas no seguían un cierto patrón. Había veces que no se recorrían porque acababan unas antes que otras y otras veces que intentaban recorrer posiciones que no existían y se salían de rango por lo que solucionamos este problema en esta fase.

7.2.3. Recorte de imágenes

A la hora de integrar esta fase al programa había que saber en que momento realizar los recortes de la imagen. Lógicamente no se puede hacer los recortes si no tienes ninguna imagen con la que trabajar, por lo que la mejor opción fue hacer estos recortes una vez tienes el ap (t), momento en el que ya has cargado la imagen completa, y ya tienes calculado el ag (T).

Como particularidad hubo que añadir al volver a pedir el ap, al reiniciar, y al finalizar el programa el borrado del directorio donde se almacenan los recortes, para que no hubiera problemas en la siguiente ejecución del programa al estar ya creados. De hecho se comprueba también al comienzo del programa si esta esa carpeta creada para eliminarla por el mismo motivo.

7.2.4. Rellenar la matriz

En el momento de integrar la matriz con la que iba a trabajar el programa había dos cosas a tener en cuenta. La primera era saber donde creabas la matriz, la cual estaba claro que seria una matriz de ceros (en el programa principal), ya que aun el usuario no ha marcado ningún individuo para el conteo. Tras analizarlo la solución mas lógica era crear esta matriz tras cargar la imagen ya que en ese momento el programa recoge los datos de esa imagen para saber su ancho, y por tanto calcular cuantas filas y columnas va a tener la matriz.

A partir de aquí el programa en cada recorte conoce en que zona de la imagen está como se comento en las secciones anteriores, para así saber en que posición está cada punto y a que espacio de la matriz le pertenece.

7.2.5. Extrapolar la matriz

Tras realizar las pruebas unitarias y tener una matriz con unos datos posibles en todas sus posiciones, hubo un problema a la hora de integrar este proceso al programa real. Y es que al hacer las comprobaciones la matriz auxiliar en la que teníamos estos posibles datos se igualaba con la matriz real que llevábamos contando. Esto hacía que la varianza fuera la misma en ambos casos y que el programa indicara que era recomendable salirse en todas las iteraciones.

El problema estaba en que antes de realizar la extrapolación, había que crear esta matriz auxiliar con los datos de la matriz que se estaba rellenando. Esto sin embargo no hacia que se creara otra matriz, sino que se creaba una variable que apuntaba a la misma posición, por lo tanto al estar cambiando la matriz auxiliar, se cambiaba también la real y por eso el valor era siempre similar en ambas matrices.

La solución estuvo en cambiar la manera de igualar las matrices, utilizando el comando .copy que conseguía lo que queríamos, crear una nueva matriz en una posición de memoria distinta que no estuviera vinculada a la otra matriz, pero si que tuviera los mismos valores. Una vez logrado esto se pudo trabajar con ambas matrices y comparar las diferentes varianzas resultantes.

7.2.6. Pop up

En este módulo utilizado para pedir el lado del recorte existía una particularidad. La realización de las pruebas unitarias se han hecho con un programa al que tras pulsar en un botón, saltaba el pop up. En el caso de integrarlo al programa completo el problema ha sido que había que meter un pop up dentro de otro pop up que es el que era llamado desde la interfaz. Esto ha variado el código ya que además de tener que realizar dos funciones distintas (una para cada pop up), había casos en los que tenían que cerrarse ambos a la vez como por ejemplo si el dato se introducía de forma correcta. Para ello se ha hecho una nueva función a la que le pides cerrar ambos pop up utilizando el Método Dismiss.

7.3. Pruebas de Sistema

Una vez completadas las fases de pruebas unitarias y de integración, se ha procedido a realizar las pruebas de sistema. Su objetivo es verificar el correcto funcionamiento del sistema en su conjunto con todos los módulos y funciones instaladas. Este tipo de pruebas se centran, principalmente, en verificar el cumplimiento de los requisitos no funcionales. Para este sistema se han establecido pruebas de rendimiento, espacio y portabilidad, dado que la mayoría de pruebas de usabilidad ya se han realizado en las anteriores secciones.

Las pruebas de rendimiento consisten en una serie de simulaciones con distintas imágenes y valores que se explican en el próximo apartado (Capítulo 8) donde se demuestra que los resultados son razonablemente similares a pesar de no recorrer todas las imágenes, y aplicar la condición de salida que permite el programa. En estos casos el tiempo de cálculo se ve disminuido de igual forma que en una ejecución normal el usuario puede finalizar el conteo antes de recorrer todas las imágenes consiguiendo resultados aceptables.

Respecto al espacio o almacenamiento se hicieron distintas pruebas ocupación de espacio de los recortes en ejecución, de las cuales se sacaron los datos que aparecen en el Cuadro 3. Los campos que aparecen en el mismo son de izquierda a derecha, el identificador de la prueba, la imagen utilizada (correspondientes al Anexo 10.3), su capacidad original, sus dimensiones, las filas, columnas y por tanto recortes que se originan con esas dimensiones, el ap o t, el rango de tamaño de recorte (el que mas ocupa y el que menos) ya que no todos los recortes pesan lo mismo, el tamaño de todos los recortes y el tamaño en disco de esos recortes.

P	Im	TamIc	Dimen	fil	col	rec	ap	TRI	TRC	TRCDisco
1	Sim1	219K	984x648	4	6	24	100	3,05K-4,50K	94K	132K
2	Sim2	276K	1024x752	5	6	30	100	3,70K*-4,88K	104K	182K
3	Sim3	64,8K	1024x680	4	6	24	100	994b-1,50K	30,7K	96K
4	Sim5	138K	1024x648	4	6	24	100	1,91K-2,75K	58,6K	96K
5	Sim4	47,6K	360x496	9	6	54	50**	704b*-1,59K	55,7K	148K
6	Sim1	219K	984x648	4	6	24	50	1,13K-1,43K	30,5K	96K
7	Sim1	219K	984x648	4	6	24	150	7,41K-9,55K	198K	256K

Cuadro 3: Espacio en ejecución

*En estos casos había recortes con información muy plana y sin datos que ocupaban mucho menos peso en el sistema. Generalmente se trataba de una fila de recortes aunque también podían ser recortes aislados.

**La simulación con la imagen 4 se tuvo que realizar con un t de 50 al no tener dimensiones suficientes para hacerla con un t de 100 como el resto. Aun así en vez de coger un t de 50 de todas, se hicieron las simulaciones generales con t de 100 porque considerábamos que mostraba resultados más clarificantes.

Sabiendo esto se han realizado dos tipos de simulaciones. El primer tipo comparando simulaciones con el mismo t. El segundo con la misma imagen (en este caso utilizamos la 1), con distinto t, mas grande y mas pequeño.

Tras realizar las pruebas lo primero que observamos como es lógico es que a imágenes similares en cuanto a dimensiones y t como son Sim1 y la Sim3, la imagen con mayor calidad ocupara mas espacio tanto en su imagen completa como tras realizar los recortes como se observa comparando las pruebas 1 y 3.

Entrando en análisis mas técnicos aunque ya explicados en anteriores secciones, podemos reconocer que el ancho de la imagen completa influye de forma significativa en el uso de espacio en disco. Esto es a causa de que en función de este ancho, el tamaño de los cuadrantes sera mayor o menor. Es una correlación debido a que cuanto mas ancha es la imagen, mayor es el valor T, y por tanto mayores dimensiones tienen esos cuadrantes. Sin embargo esto no afecta al tamaño de los recortes que vienen determinados por t y obviamente cuanto mayor sea este valor, mayores serán las dimensiones de los recortes y mayor espacio en disco ocuparan como se puede ver en la diferencia de TRI y TRC en las pruebas 1, 6 y 7, realizadas con la misma imagen.

Las dimensiones de la imagen completa influyen de otra manera, y es que cuanto mas alta es la imagen, o mejor dicho cuanto mayor es su dimensión y en proporción con su dimensión x, mayor sera el número filas necesarias para ocupar ese espacio, y por tanto mayor sera el número de cuadrantes, lo que lleva a un mayor número de recortes que ocupan espacio. Es por ello por lo que la Imagen Sim4 de la Prueba 5 tiene tantos recortes y a pesar de ocupar menos en su estado inicial, los recortes ocupan mas como puede verse en las pruebas 4 y 5 con mismo t. De hecho se puede llegar a la conclusión de que en función de las distintas proporciones de x/y hará falta mas o menos cuadrantes, lo cual resumimos en el siguiente cuadro para algunas de las pruebas realizadas.

Dimen	Proporción	T	uti	fil	col	cuadrantes
984x648	1,518	164	3,951	4	6	24
1024x752	1,361	170	4,42	5	6	30
360x496	0,725	60	8,266	9	6	54

Cuadro 4: Proporción de imagen (x/y), valor de T resultante y cuadrantes necesarios.

De todos modos esta ocupación en el sistema es temporal, ya que como se ha explicado los recortes se eliminarán a la finalización de la ejecución.

Por último, las pruebas de portabilidad no han podido llevarse a cabo al no tener completada la versión en Android, la cual desarrollaremos en próximas versiones de la app, como viene indicado en la Sección 9.2, Trabajos futuros.

7.4. Pruebas de Aceptación

Las pruebas de aceptación se conciben como la última parte del proceso de verificación, ya que se refieren a la validación del funcionamiento del sistema según los propios usuarios del mismo, así como, la adecuación a los objetivos y requisitos establecidos al principio del proyecto.

Estas pruebas por el momento no se han realizado al no tener una versión definitiva desarrollada y en producción. Por tanto queda como trabajo futuro comenzar la producción del software y valorar el feedback de los usuarios del mismo, para validar que el sistema cumple de forma satisfactoria los requerimientos establecidos.

8 Simulaciones

En esta sección se mostrarán las distintas simulaciones realizadas con el programa y los resultados obtenidos. Separaremos dicho apartado en dos secciones, una primera con la explicación y el objetivo de las mismas, y la segunda con los resultados obtenidos.

8.1. Funcionamiento de las simulaciones

Las simulaciones se realizan con un programa independiente al programa principal, que no requiere de interfaz ni de uso por parte del usuario. Si que contiene la mayoría de los cálculos del programa principal (variables, funciones) para ser lo mas aproximado posible al programa real.

El objetivo principal de estas simulaciones es entender como influye descartar el recorrer todas las imágenes (rejillas) y entender donde estaría el limite en función de los datos para que los datos sean relevantes, teniendo en cuenta que contra menos imágenes haga falta recorrer, menos tiempo se tardara en analizar la imagen y por tanto mayor eficiencia tendrá el programa.

La diferencia principal aparte de ser un programa automático como se citó anteriormente, es que este programa trabaja con datos reales de imágenes (coordenadas), las cuales transforma en matrices una vez se conoce el área de rejilla y la posición de la rejilla para cada una de ellas, de lo cual hablaremos mas adelante. De hecho para realizar las simulaciones en este programa auxiliar trabajamos siempre con la misma imagen y las mismas coordenadas de datos que coinciden con dicha imagen.

Una vez con la matriz real, se creará una matriz auxiliar que comenzará vacía, e ira transformando cada posición en la posición de la matriz real, para ir haciéndola similar. Esto se hará añadiendo en cada paso el valor real en la posición de la matriz que estamos recorriendo (la que indica el generador) a la matriz auxiliar. Entre cada paso además de que se recorre una rejilla, se comparan ambas varianzas (la de la matriz real y la de la matriz auxiliar), y si cumplen la condición que permite finalizar la simulación muestra el resultado y la acaba, pasando a una nueva simulación. Con esto queremos comprobar cuantos cuadrantes aproximadamente hay que recorrer para que ambas varianzas sean suficientemente parecidas y por tanto ambas matrices lo sean también. Como condición para acabar la simulación actual hemos indicado que se deben haber recorrido el 50 % de las posiciones, y la diferencia de varianza entre ambas matrices debe ser inferior a un porcentaje que se comenta mas adelante.

Respecto a las variables indicadas anteriormente, la idea de las simulaciones es hacer una gran cantidad de ellas con distintos valores de t (lado de rejilla) y distintas posiciones de dicha rejilla que serán valores aleatorios. Todo esto teniendo en cuenta las reglas del programa (t no puede ser mayor que T y en función de la posición de la rejilla será limitado para no salirse del cuadrante, etc.). Hemos decidido que ap (t) puede tener valores de entre $\frac{ag}{10}$ y $\frac{ag}{2}$ (siendo ag la variable T)y que el programa en total haga 10000 simulaciones y guarde los resultados en un fichero txt que se envía a otra función en la que se crea una gráfica para poder analizar las conclusiones, como hacemos en el siguiente capítulo.

Falta añadir que tras realizar las pruebas con una imagen para entender que el programa funciona, se ha utilizado un conjunto de datos con los puntos etiquetados, disponible en UCF [2011], de las cuales

conocíamos sus coordenadas de puntos para realizar las simulaciones. De aproximadamente 50 imágenes nos hemos quedado con 5 que nos han parecido las mas concluyentes. Estas imágenes son las de mayor y menor tamaño, las de más y menos datos (coordenadas) y una con un termino medio tanto de tamaño como de datos. Se encuentran en el Apéndice 10.3.

También se han dado distintas diferencias de varianza para poder salirse del programa, siendo un 30 % una diferencia determinada como grande, un 10 % una diferencia media, un 1 % una diferencia pequeña, y un 0.1 % una diferencia muy pequeña.

Todos estas diferencias en los valores se hacen para poder sacar más y mejores conclusiones de las simulaciones que se realizan pasando al programa estos datos de entrada en modo lista (los de los ficheros y los de las diferencias para terminar), y ejecutando todas las simulaciones de forma automática y continua para evitar tener que pasar de forma manual cada imagen y cada diferencia de varianza.

Antes de seguir, explicar que las gráficas tienen dos ejes, el eje x que indica el porcentaje de cuadrantes que se recorren en cada simulación antes de dar por válido el resultado, y el eje y que es el porcentaje de diferencia entre la varianza real y la generada.

Además hay que explicar que una de las características de estas gráficas es que no tienen en consideración una serie de valores de las simulaciones al ser los mas extremos. Para ser mas concretos, hay puntos fuera del dibujo de la gráfica, esto es porque son el 5 % de los valores mas superiores e inferiores, y pueden ser valores que den problemas para generar unas conclusiones claras, y por ello se desestiman dejándolos fuera de las lineas azul y naranja que delimitan el percentil de simulaciones a considerar.

8.2. Resultados y Análisis

Una vez ejecutado el programa tenemos todas las gráficas de todos los valores con sus 10000 simulaciones distintas, gráficas que añadimos a una misma imagen (Ilustración 27) para que se puedan observar todas a la vez y así explicar de una forma mas clara las conclusiones que se sacan con el programa.

Tras mostrar las 20 gráficas resultantes (5 imágenes x 4 diferencias de varianza) podemos entrar a valorar ciertas conclusiones. Por ejemplo, una de las primeras deducciones que se sacan es que al estar comparando dos matrices que tienen el mismo origen al provenir del mismo sistema de datos, es posible que se parezcan mucho y por tanto que al llegar al 50 % de imágenes recorridas el rango la diferencia de varianza se cumpla y se acabe esa simulación en concreto. Esto es obviamente menos frecuente cuanto más exigente sea la similitud de varianzas para finalizar la ejecución. No es lo mismo necesitar que la igualdad sea de un 70 % (30 % de diferencia), que sea de un 99 % (1 % de diferencia).

Otra conclusión que se observa es que contra más datos contiene la imagen, menor es el parecido de las gráficas, probablemente porque hay más muestras. Esto nos puede llevar a razonar que para imágenes con menos datos podemos ser más flexibles e incluso valorar modificar el valor de salida en el programa principal en función del número de datos, aunque sin conocer este número concreto ese dato sería una estimación mas que un dato sólido que nos permitiera sacar mayor rendimiento a estas imágenes. Una estimación a priori del número podría ser tomada de forma visual, dando un simple vistazo por el usuario.

Por otro lado vemos que las gráficas difieren considerablemente con los mismos datos, excepto las de diferencia pequeña y muy pequeña que son las más similares. Esto hay que tenerlo en cuenta a la hora de querer finalizar el programa principal, al que hemos impuesto una diferencia media (10 %) para permitir salir. Al final consideramos que es una diferencia posible ya que las gráficas con diferencia grande de salida no tienen nada que ver con el resto, y en el caso de las gráficas de diferencia media tienen mayor similitud a las gráficas reales.

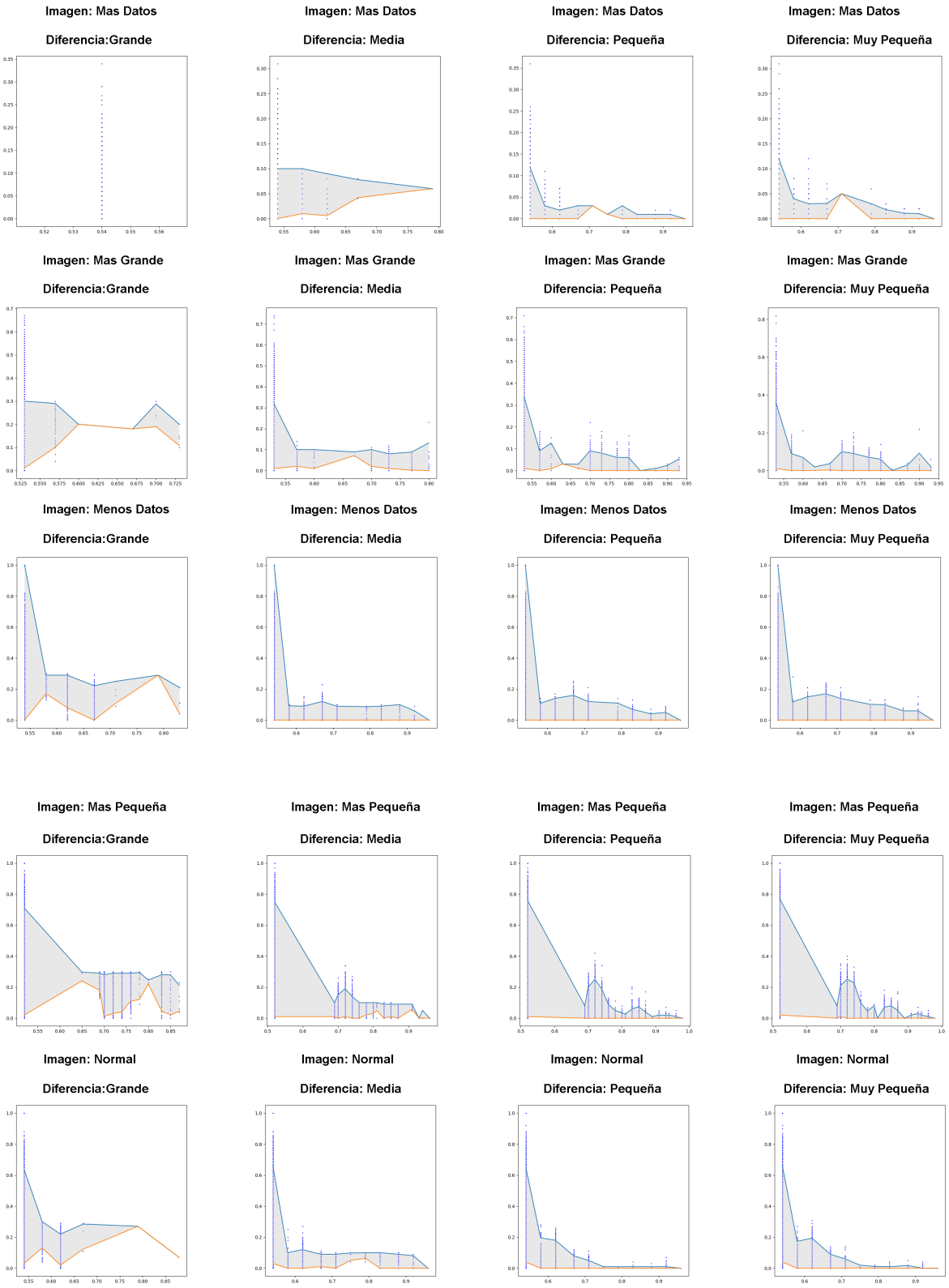

Ilustración 27: Gráficas resultantes con las simulaciones. Cada fila de gráficas representa simulaciones de la misma imagen con distintas diferencias de varianza y cada columna de gráficas representa simulaciones con la misma diferencia de varianza en distintas imágenes. La franja entre las lineas azules y naranja es la zona donde aparecen los resultados mas comunes (90 %) de cada simulación.

9 Conclusión y Trabajos Futuros

Para finalizar, el objetivo de este último capítulo de la memoria es recoger las conclusiones obtenidas en la realización del proyecto, así como un conjunto de posibles características y mejoras que pueden incorporarse en el futuro a este trabajo o a posteriores.

9.1. Conclusiones

Valorar que el desarrollo de esta aplicación ha sido muy gratificante, además realizándola de forma individual con la ayuda de mis tutores, se ha diseñado una aplicación útil aplicando conocimientos recogidos durante los cursos, y aprendiendo muchos otros por el camino. Una vez llegados al final de la memoria me quedo con los siguientes puntos:

- Se han dado los pasos necesarios para el desarrollo una aplicación web y una aplicación móvil que permitan contar el número de individuos en una foto mediante el uso de un método que mezcla una parte de conteo manual y otra de estimación estadística para dar un resultado con una estimación del error.
- Se han usado herramientas de desarrollo basadas en Python y se han realizado simulaciones de este método para ver cuales son los parámetros más adecuados para que se alcancen resultados fiables con el menor número de conteos.
- Pese a que la aplicación no se ha pasado a un servidor de producción, se ha obtenido una versión inicial que permite probar las principales funciones de esta aplicación, observándose en simulaciones que el rendimiento es adecuado.

Indicar que todo el código, y la documentación pertinente se almacena en el Repositorio de GitHub. La rama estable del producto es la Master.

9.2. Trabajos Futuros

En esta última sección se comentan los puntos que quedan en el desarrollo del software para dejarlo terminado, empezar a producirlo de forma comercial y mejorarlo posteriormente.

- Realizar la portabilidad del software a dispositivos móviles y en concreto dispositivos Android. Para ello la idea principal es utilizar python-for-android, un programa que genera paquetes para esa clase de sistemas, y así conseguir este objetivo. Además tras desarrollarlo habrá que realizar pruebas de portabilidad para comprobar que el funcionamiento del sistema es similar al funcionamiento en PC.
- Realizar el RFU05, que indica que el usuario podrá deshacer el último punto marcado directamente. Esto hará la aplicación mas flexible para el usuario ya que en caso de marcar un punto erróneo podrá borrarlo sin necesidad de volver a la imagen anterior y tener que volver a recorrer dos imágenes completas como se hace dando al Botón Volver. Para ello habrá que eliminar el punto tanto en la matriz, como en la interfaz, para lo que habrá que plantear la manera mas eficiente de hacerlo.
- Rediseñar el generador para que recoja los recortes de una manera mas eficiente mejorando el potencial del software.

- Mejorar servidor web con distintas funcionalidades como pueden ser un login para la creación de usuarios que relacione estos usuarios con las imágenes o datos que suben al mismo.
- Implementar requisitos básicos de seguridad cumpliendo con el Esquema Nacional de Seguridad [CCN, 2015], en sus puntos 5.7 (Protección de la información), y 5.8.2 (Protección de servicios y aplicaciones web) con el objetivo de presentar política de seguridad a nuestra aplicación.
- Mejorar el rendimiento de la aplicación mediante el uso de técnicas de Broadcasting, explicado en el Anexo 10.1.1.
- Crear y añadir un nombre y un logotipo del programa que será incluido en la Escena Inicio para convertirlo en un producto comercial y vistoso. Una de las posibles opciones de nombre para el programa sería PhotoCount.
- Hacer pruebas de usuario, que permitan adaptar el interfaz a las características del usuario objetivo (pruebas de aceptación de usuario).
- Recoger datos de usuarios reales y verificar sobre un dataset más grande los resultados obtenidos en simulación para un mejor análisis de los resultados finales. Ayudarse de Multiprocessing (explicado en el Anexo 10.1.2) para ello.
- Hacer mejoras al programa para modificar la condición de salida en función de la imagen y de su cantidad posible de datos ya que como hemos visto en el Capítulo 8, los datos de una imagen tienen una importancia máxima en los resultados de las varianzas y sus similitudes.

Comentar que en el Repositorio Git se ha incluido una Rama Experimental que se usará para depurar el código e incluir nuevas funcionalidades de cara a las nuevas versiones.

10 Anexos

10.1. Teoría Adicional

10.1.1. Broadcasting

El Broadcasting permite trabajar con arrays de distintas dimensiones, dimensionando el array mas pequeño respecto al mas grande para que tengan formas compatibles y a partir de ahí, realizar las operaciones oportunas.

El único requisito para el uso de Broadcasting es que haya una forma de alinear las dimensiones de ambos arrays para poder operar con ellos. En otras palabras, el array mas pequeño tiene que tener una dimensión 1 y la otra dimensión tiene que ser compatible con el otro array (similar en una de las dimensiones).

La Imagen 28 es un ejemplo de Broadcasting. Se consigue operar con los arrays a y b con los que en un primer momento no parece poder operarse.

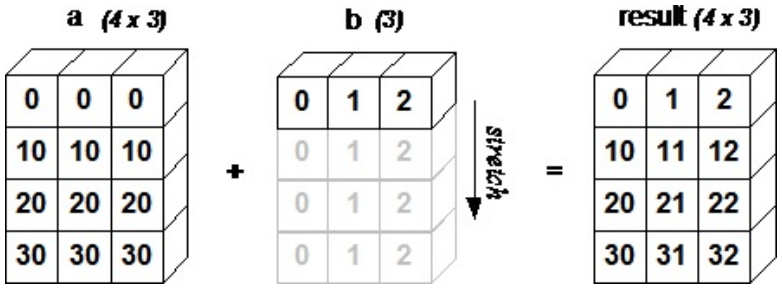

Ilustración 28: Ejemplo de Broadcasting. Imagen sacada de Sneja Shah

Es una funcionalidad de mucho uso que pensamos utilizar en las operaciones con matrices. Sin embargo finalmente no se empleó durante el proyecto, y queda pendiente para trabajos futuros relacionados como se indica en la Sección 9.2 dedicada a ellos.

10.1.2. Multiproccesing

El multiprocesamiento o "Multiprocessing" es el uso de dos o más procesadores (CPU) para la ejecución de procesos (puede ser uno o varios). No hay que confundirlo con multitasking (multitarea), que significa la ejecución de uno o varios procesos en un mismo sistema. El multiprocesamiento tampoco significa que un solo proceso o tarea utilice más de un procesador simultáneamente. Eso se llama procesamiento paralelo.

La idea en este proyecto era utilizar Multiprocessing a la hora de realizar las simulaciones, ya que son un gran número de ellas, separando las tareas y aprovechando al máximo los múltiples procesadores del sistema en el que se ejecuten para ahorrar tiempo en los cálculos y como consecuencia que la ejecución dure el menor tiempo posible. Sin embargo como en la sección anterior, no ha sido necesario su empleo durante el proyecto, y queda pendiente su uso en simulaciones mas complejas, que nos aporten una mayor cantidad de datos, o que utilicen imágenes de mayor calidad para sacar mejores conclusiones al respecto.

10.2. Código

Listing 10.1: Diseño de interfaz en Kivy

```
#:kivy 1.1.0
Root:
        entro: True
    BoxLayout:
        orientation: 'vertical'
        BoxLayout:
                size_hint_y: None
            height: 30
            Button:
                opacity: 100
                        disabled: False
                        id: Cargar
                text: 'Load'
                on_release: root.show_load()
        BoxLayout:
                    size_hint_y: None
                        size_hint_x: None
                        height: 20
                BoxLayout:
                        BoxLayout:
                                size_hint_y: None
                                size_hint_x: None
                    Image:
                                id: mario
                                size_hint_y: None
                                size_hint_x: None
                                size: 600, 400
                                pos: 200, 300
                        source: 'fondo.jpg'
                allow_stretch: True
                                keep_ratio: False
                                MyPaintWidget:
                                    id: lienzo
                                        size_hint_y: None
                                        size_hint_x: None
                                        size: 600, 400
                                    pos: 100, 110
                                        pinta: False
                                        nec: -30
                                        nec2: -30
                        BoxLayout:
                size_hint_y: None
                                size_hint_x: None
                BoxLayout:
                    size_hint_y: None
                        size_hint_x: None
                        height: 15
                BoxLayout:
                        size_hint_y: None
                        height: 30
                        BoxLayout:
                                orientation: 'vertical'
                                Label:
                                    id: var_text
                                        text: "Coeficiente"
```

```
            BoxLayout:
            Button:
                    opacity: 0
            disabled: True
                    id: Sig
                    text: "Siguiente"
                    on_release: root.siguiente()
            BoxLayout:
            Button:
                    opacity: 0
            disabled: True
                    id: Finalizar
                    text: "Finalizar"
                    on_release: root.guardarresultados()
            BoxLayout:
                    size_hint_x: None
    BoxLayout:
        size_hint_y: None
            size_hint_x: None
            height: 15
    BoxLayout:
            size_hint_y: None
            height: 30
            BoxLayout:
                    orientation: 'vertical'
                    Label:
                            id: var_est
                            text: "Estimacion"
            BoxLayout:
            Button:
                    opacity: 0
            disabled: True
                    id: Vol
                    text: "Volver"
                    on_release: root.volver()
            BoxLayout:
            Button:
                    id: Salir
                    text: "Salir"
                    on_release: exit()
            BoxLayout:
                    size_hint_x: None
    BoxLayout:
        size_hint_y: None
            size_hint_x: None
            height: 15
```

Listing 10.2: Metodo Load

```
def load(self, path, filename):
        print(filename[0])
        extension = os.path.splitext(filename[0])[1] #Para coger la extension (Con
            el import os)
        nombrefichero=os.path.splitext(filename[0])[0]

        #Si la extension no es jpg, pasarlo a jpg (el nombre y la imagen)
        if extension != ".jpg":
                    if extension == ".png" or extension ==".jpge" or extension ==".
                        gif":
                im = Image.open(filename[0])
                im.save(nombrefichero+".jpg", quality=95)
                filename[0]=nombrefichero+".jpg"
            else:   #Si el formato no es una imagen
                print ("Formato Incorrecto")
                self.dismiss_popup()
                return 0  #volver al inicio sin cargar nada

        #Para reescribir el fichero de coordenadas
        f=open('coordenadas.txt','w')
        f.write ("COORDENADAS \n")
        f.write ("---------- \n")
        f.close()

        #Para rescribir el fichero resultados
        f=open('resultados.txt','w')
        f.write ("RESULTADOS \n")
        f.write ("---------- \n")
        f.close()

        self.ids["mario"].source=filename[0]
        self.ids["var_text"].text= "20"

        self.dismiss_popup()

        #Hacer aqui el pop up del area pequena
        global rutafoto
        rutafoto = (filename[0])
        datosfoto= devolverancho(rutafoto) #retorno los datos de la imagen
        ancho=(datosfoto[0])

        global ag
        ag = math.ceil(ancho/6) #se puede cambiar el numero de columnas
        global ap
        ap = int(input("Lado Pequeño: ")) #No esta permitido que el cuadrado pequeño
             sea mayor o igual que el cuadrado grande
        while ap> ag:
            print("Area Pequena se sale de Area Grande")
            ap = int(input("Lado Pequeño: "))

        global filas
        global columnas
        global numero_imagenes
        numero_imagenes, filas, columnas=recortar_imagen(ag, ap, rutafoto)
        global matrix
        global mataux
        matrix=np.zeros((filas, columnas)) #Rellena
        mataux=np.zeros((filas, columnas)) #matriz con los valores reales en las
```

```
            posiciones recorridas y con ceros en el resto para calcular su varianza y
             calcular con la que estamos calculando nosotros. La inicializamos de la
            misma forma que la otra

        self.ids["Sig"].opacity=100
        self.ids["Sig"].disabled=False

        self.ids["Cargar"].text="Reiniciar"
        self.ids["Cargar"].on_release= self.reset()

        self.ids["Vol"].opacity=0
        self.ids["Vol"].disabled=True

        self.ids["Finalizar"].opacity=0
        self.ids["Finalizar"].disabled=True
```

Listing 10.3: Metodo Rellenamatriz

```
def rellenarmatriz(self, matriz, vacios):

                matextra=np.matrix.copy(matriz)

        #coger el maximo y el minimo de la matriz sin contar los -1

        maximo=np.max(matriz)
        if maximo < 1: #Para que no sea 0
            maximo=1
        minimo=0

        global listagenerador
        global numero_imagenes

        for i in range(numero_imagenes-vacios, numero_imagenes):
            matextra [listagenerador[i]]=randint (minimo, maximo)

        print (matextra, "Matriz extrapolada")
        print (matriz, "Matriz no extrapolada")
        return matextra
```

Listing 10.4: Codigo para calcular la varianza

```
#Programa que calcula la varianza compuesta

import pickle, sys
import numpy as np


def varc(ag,ap,a):

        tau = ap/ag

        j,n=a.shape
        #shape coge en una tupla la dimension del array

        #para coger qoi y qei, es decir la suma de los numeros
        qoi=a[[i for i in range(j) if i%2 ==0]]
        qoi=np.sum(qoi,axis=0)

        qei=a[[i for i in range(j) if not i%2 ==0]]
        qei=np.sum(qei,axis=0)

        array_qei= np.array(qei) #Para quitar las comas
        array_qoi= np.array(qoi)
        array_qop= array_qoi - array_qei
        array_qop= array_qop**2
        qs=np.sum(array_qop)

        array_Qi=array_qoi + array_qei
        Qi=np.sum(array_Qi)

        #C0 C1 y C2 , con index
        C0= array_Qi **2
        C0=np.sum(C0)

        #C1
        aux1C1= np.insert(array_Qi,0,0)
        aux2C1= np.insert(array_Qi,len(array_Qi),0)
        C1= aux1C1 * aux2C1
        C1=np.sum(C1)

        #C2
        aux1C2= np.insert(array_Qi,0,0)
        aux1C2= np.insert(aux1C2,0,0)
        aux2C2= np.insert(array_Qi,len(array_Qi),0)
        aux2C2= np.insert(aux2C2,len(aux2C2),0)
        C2= aux1C2 * aux2C2
        C2=np.sum(C2)

        #vn
        vn=(((1-tau)**2)/(3-2*tau))* qs

        #varianza
        varc= (1-tau)**2/(tau**4 * (2-tau))/6 * (3*(C0 - vn) -4*C1 + C2) + vn/tau**4
        return (varc)



def tras(matriz,ag,ap):
            ag=ag*ag
```

```
        ap=ap*ap
        matt= np.transpose(matriz) #hace la traspuesta
        vc1= varc(ag,ap,matriz) #calcula la varianza de la matriz normal
        vc2=varc(ag,ap,matt) #calcula la varianza de la matriz traspuesta
        vc= (vc1+vc2)/2 #hace la media de las 2
        return (vc)
```

Listing 10.5: Codigo para pintar y contar

```
def on_touch_down(self, touch):
    with self.canvas:
        Color(1, 1, 0)
        print("TOuch %.2f %.2f"%( touch.x,touch.y),self.contador)
        if touch.x > self.pos[0] and touch.x < self.pos[0] + self.size[0] and
            touch.y > self.pos[1] and touch.y < self.pos[1] + self.size[1] and
            self.pinta:
            self.contador +=1
            self.conjunto.add((touch.x,touch.y))
            d = 10.
            Ellipse(pos=(touch.x - d / 2, touch.y - d / 2), size=(d, d))
            global listagenerador
            fila=int(listagenerador[numimagenactual-1][0]) #Me quedo con la
                parte entera
            col= listagenerador[numimagenactual-1][1]

                            global matrix
            matrix[fila][col]+=1 #matrix[fila][col]+ 1
            cpx,cpy= devolveresquina ()
            print (self.pos[0], self.pos[1], cpx , cpy , ag , ap , fila,col)
            self.dimensionar(touch.x,touch.y, self.pos[0], self.pos[1], cpx, cpy
                , ag, ap, fila,col)
```

Listing 10.6: Codigo para dimensionar una coordenada

```
    def dimensionar (self, touchx, touchy, ex, ey, cpx, cpy, ag, ap, fila, columna):

        #sacar la coordenadada en la imagen dimensionada (no en el programa)
        coordx=touchx - ex
        coordy=touchy - ey

        #Dimensionar la imagen (area p con dimensiones por defecto 600,00)
        coordx=coordx*ap/self.size[0] #anchura del cuadrado en el que se presenta la
            imagen (x)
        coordy=coordy*ap/self.size[1] #altura del cuadrado en el que se presenta la
            imagen (y)

        #Poner la coordenada repsecto al ag
        coordx=coordx+cpx
        coordy=coordy+cpy

        #Sabiendo la fila y la columna poner la coordenada real en la imagen
        global filas
        filaaux=filas-1-fila #filaaux para coger la fila para calcular bien la
            coordenada empezando desde arriba
        coordx=coordx+ag*columna
        coordy=coordy+ag*filaaux

        f=open('coordenadas.txt','a')
        f.write ("%.2f␣\t␣%.2f\n"%(coordx,coordy)) #Para que te saque solo 2
            decimales (expresiones)
        f.close()

        pass
```

Listing 10.7: Metodo Reset

```
def reset(self ):

        global ruta
        global matrix
        global mataux
        global numimagenactual
        global columnas
        global filas
        global numero_imagenes
        global ag
        global ap
        global rutafoto
        global listagenerador

        #Vuelvo a inicializar las variables como al principio del main
        ruta= "."
        ruta= os.path.join(ruta,"Recortes")

        matrix=[] #La matriz que hay que pintar
        mataux=[]
        numimagenactual =0
        columnas=0
        filas=0
        numero_imagenes=0
        ag=0
        ap=0
        rutafoto= "" #para pasarlo despues al fichero la pongo como global
        listagenerador=[]

        if os.path.exists(ruta):
            shutil.rmtree(ruta)

                #Borrar datos de pantalla
        self.ids["var_text"].text= ""
        self.ids["var_est"].text= ""
        del self._popup

        #Quitar los botones
        self.ids["Sig"].opacity=0
        self.ids["Sig"].disabled=True

        self.ids["Finalizar"].opacity=0
        self.ids["Finalizar"].disabled=True

        self.ids["Vol"].opacity=0
        self.ids["Vol"].disabled=True

        Editor().stop()
        Editor().run()
        shutil.rmtree(ruta)
```

Listing 10.8: Diseño de base de datos

```
class Foto (db.Model):
        id = db.Column(db.Integer, primary_key =True)
        image_name = db.Column(db.String(20), unique=True, nullable=False)
        image_file = db.Column(db.String(20), unique=True, nullable=False)
        descripcion = db.Column(db.String(200), unique=False, nullable=False)
        datos = db.relationship('Datos', backref='nombrefoto', lazy=True)

        def __repr__(self):
                return f"Foto('{self.image_name}','{self.image_file},'{self.
                    descripcion}')"

class Datos (db.Model):
        id = db.Column(db.Integer, primary_key =True)
        tp = db.Column(db.String(20), unique=False, nullable=False)
        Tg = db.Column(db.String(20), unique=False, nullable=False)
        cpx = db.Column(db.String(20), unique=False, nullable=False)
        cpy = db.Column(db.String(20), unique=False, nullable=False)
        coerror = db.Column(db.String(20), unique=False, nullable=False)
        estimacion = db.Column(db.String(20), unique=False, nullable=False)
        foto_id = db.Column(db.Integer, db.ForeignKey('foto.id'), nullable=False)
```

10.3. Imágenes utilizadas en las simulaciones

Imágenes sacadas de UCF [2011].

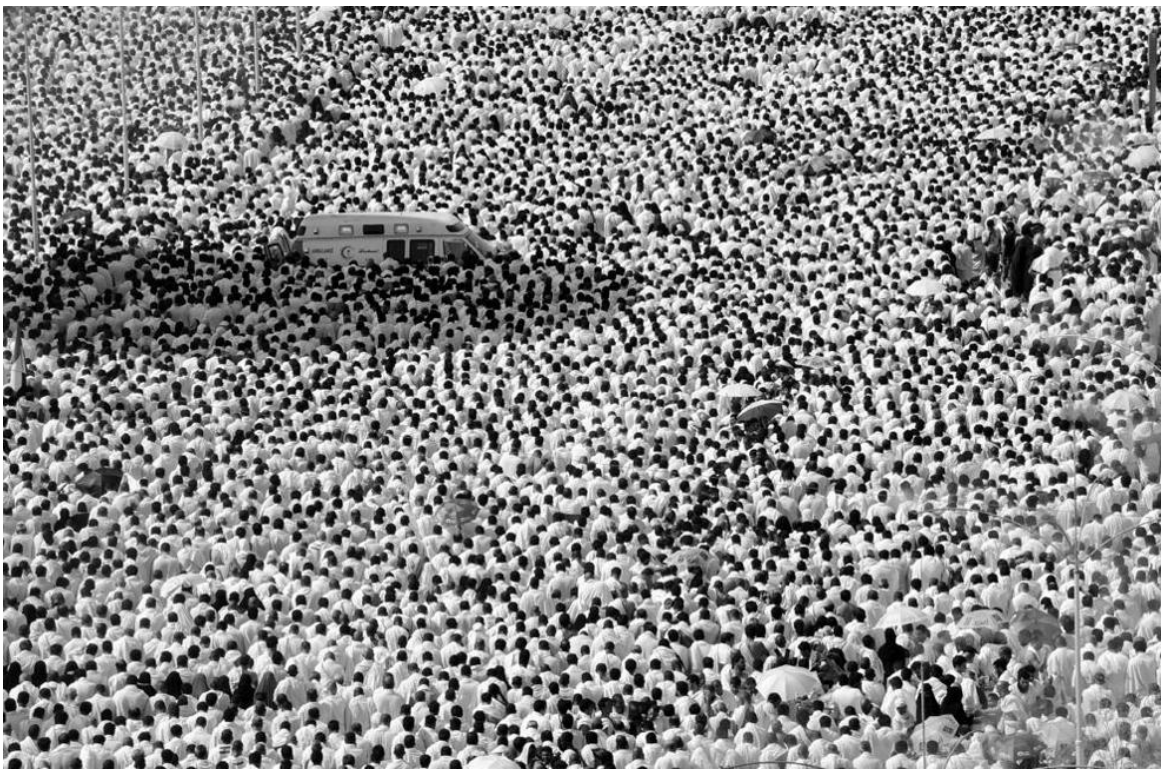

Ilustración 29: Imagen Simulaciones 1. Imagen con mas partículas de las disponibles en el dataset.

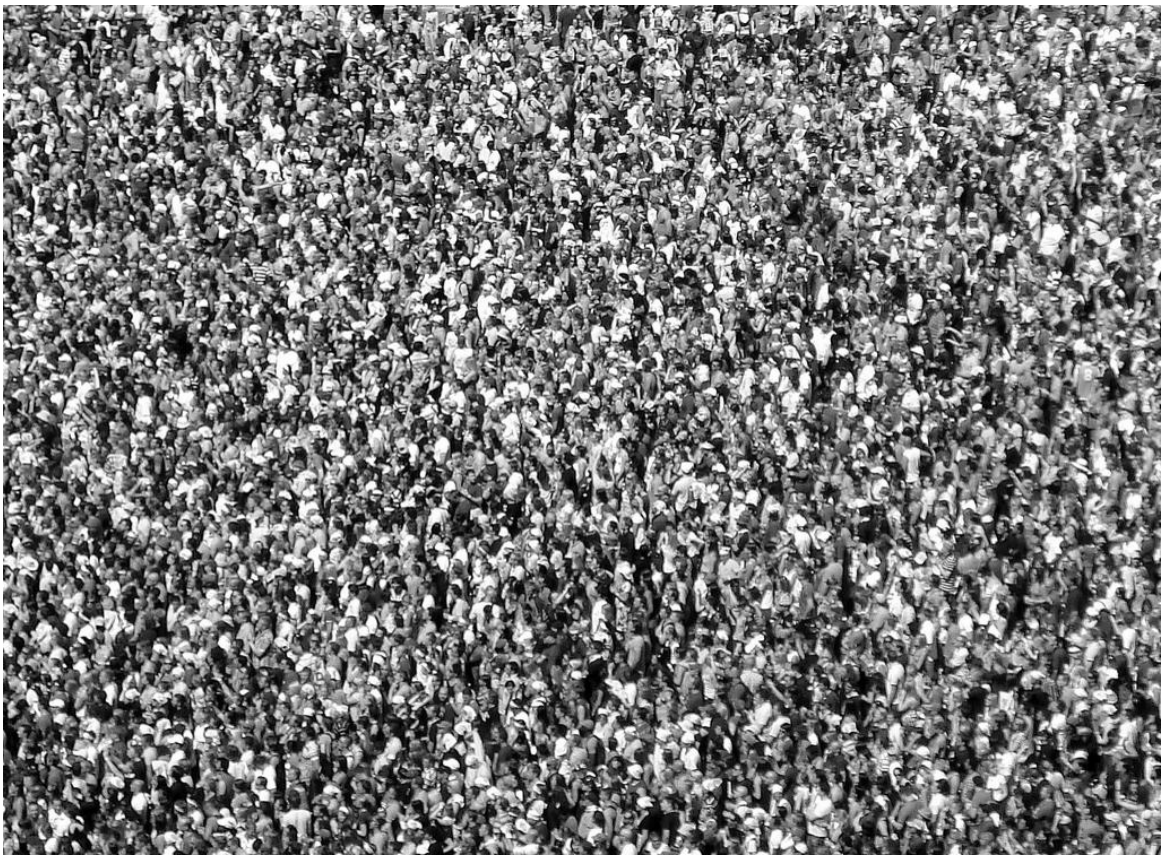

Ilustración 30: Imagen Simulaciones 2. Imagen más grande de las disponibles en el dataset.

Ilustración 31: Imagen Simulaciones 3. Imagen con menos partículas de las disponibles en el dataset.

Ilustración 32: Imagen Simulaciones 4. Imagen más pequeña de las disponibles en el dataset.

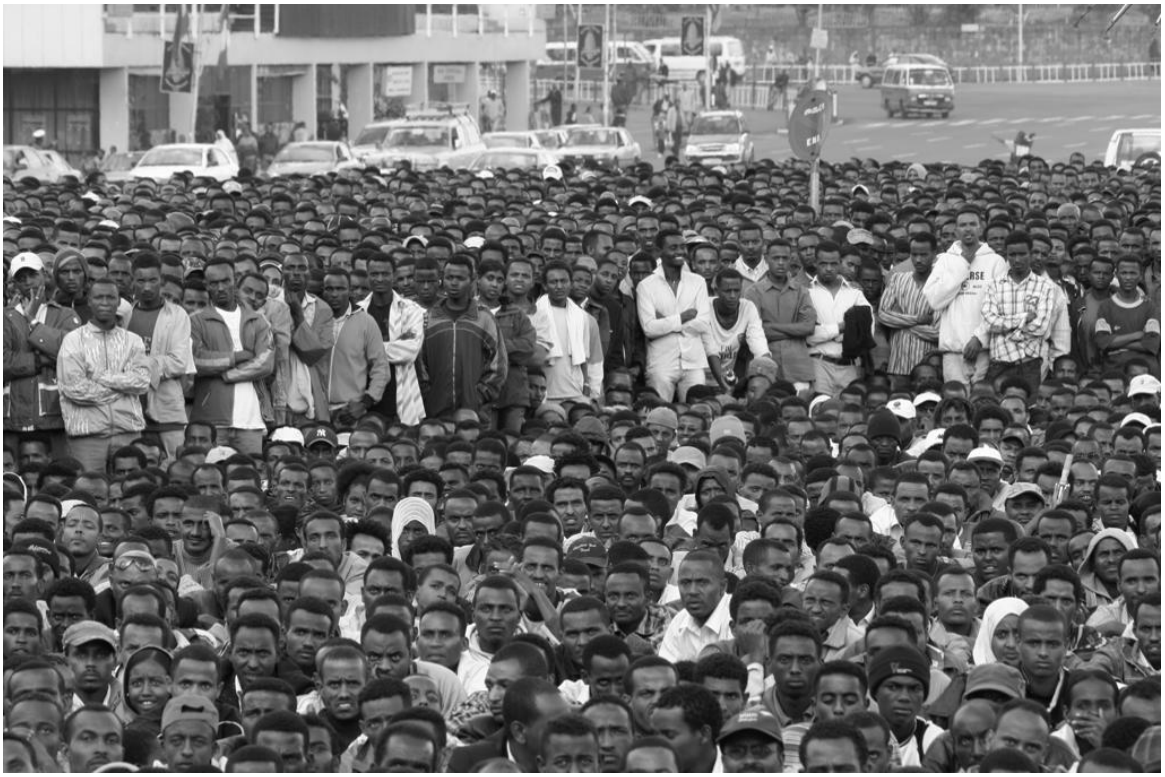

Ilustración 33: Imagen Simulaciones 5. Imagen estándar de las ofrecidas en el dataset.

10.4. Tareas

1	Entender articulo de Marcos Cruz y sus cálculos
	Estudiado e instalado librerias (Kivy, Python for android)
	Analisis Software Countem (Unican)
	Discutir librerias
	Discutir mejoras Software Countem (Reunion siguiente)
	Informarse acerca de librerias (Kivy)
2	Memoria (requisitos funcionales)
	Discutir acerca de la Memoria (Introduccion, Requisitos y metodologia)
3	Inicio programacion Kivy
4	Analisis de Software a utilizar
	Problemas instalacion Kivy
	Analisis de metodo de contar personas y cálculos (estimacion y varianza)
	Discusión de entrada de datos (coordenadas)
5	Analisis pruebas unitarias a realizar
	Analisis varianza compuesta (Articulo de Marcos Cruz)
	Analisis de la manera de generar los recortes
	Investigacion sobre KdTrees
	Iniciar la aplicación de KdTrees
6	Solucionar errores (fórmulas)
	Solucionados errores con librerias (Kivy y Numpy)
	Discutir acerca de mejoras para el software Countem (Traspuesta)
7	Discutir acerca de la varianza (Varianza muestral vs total)
	Programa Python calcular varianza simple con datos aleatorios
	Programa Python leer coordenadas de fichero
	Cambiar programa Python calcular varianza simple para que utilice de entrada matices y no filas
	Solucionar problema de entrada de datos (Coordenadas no diferenciadas)
	Discutir problemas Kivy
	Investigacion sobre Kdtrees y Scipy
8	Programa Python calcular varianza complicada con datos aleatorios
	Solucionar error en el cálculo de la varianza simple (valor n)
	Mejorar programa de la varianza compuesta (quitar bucles for y usar arrays y numpy)
	Analisis varianza compuesta (Articulo de Marcos Cruz) (C1,C2)
	Investigacion sobre KdTrees
	Finalizar varianza compuesta y comprobarla
9	Programa Python cuente los puntos sabiendo las coordenadas y el tamaño del cuadrado
	Creado programa de prueba que crea matriz según coordenadas
	Mejorar programa de la varianza compuesta (variables qoi y qoe, instrucciones mas eficientes)
	Investigacion sobre KdTrees (Distancia Manhattan)
	Investigacion sobre Pillow
10	Añadir matriz traspuesta a la varianza compuesta
	Solucionados errores con librerias (Pillow)
	Desarrollo RFS07 (KdTrees para todos los cuadrados)
	Investigacion sobre Broadcasting
	Discutir programacion (lineas que cortan)
	Discutir mejoras del programa (comparar varianza cambiando cuadrados de posicion) (1/2)
11	**Iniciando requisito RFS08 (Programa Pillow que haga y guarde recortes)**
	Programando RFS08 (Programa Pillow que haga y guarde recortes)
	Mejorar programa recortes (collage)
	Desarrollo RFS04 (KdTrees evitar lineas que cortan)
	Discutir mejoras del programa (comparar varianza cambiando cuadrados de posicion) (2/2)
	Investigar otras librerias (Python for android , Matplotlib)
	Investigar pruebas unitarias (Dive into Python3)
12	Instalar e investigar sobre latex (the not so short introduction to latex)

	Programa (varianza cambiando cuadrados de posicion)
	Pruebas RFS04 (KdTrees, evitar lineas que cortan)
	Discutir pruebas unitarias
	Analisis diseño de la app de Kivy (Capa de presentación)
13	**Analisis RNF05 (cálculo de la matriz, orden de los cuadrados)**
	Solucionados errores varianza compuesta (extrapolar)
14	Investigacion Programacion Kivy
	Solucionados errores varianza compuesta (cálculo en cuadrados grandes en vez de en pequeños)
15	Investigar Pydev
	Investigacion Programacion Kivy (lenguaje) (1/3)
16	Investigacion Multiprocessing
	Programar RFS03 (Kivy que pinta cuando marcas)
	Programar RFS03 (Programa Kivy, pinta y guardar coordenadas)
	Discutir RFU01 (imagen y como cargarla (filechooser) - capa de presentacion)
17	**Programacion del requisito RFU01 (Filechooser)**
	Solucionar problema con el diseño de la pagina inicial del programa (layouts y tamaño de la imagen)
	Analisis layouts app (Capa de presentacion)
	Solucionado error que no permitia cargar imágenes
18	Investigacion programacion Kivy (2/3)
	Solucionar errores canvas (se superponia en la app)
19	Añadido Boton Siguiente aun sin cumplir su funcion
	Discutir simulaciones
	Incluir el programa recortes en el main (por desarollar)
	Mejora canvas - añadido contador (Para formar la matriz)
	Programar el renombre de los recortes para localizarlos en la app
20	Programar etiquetas de estimacion y varianza en la pp
	Solucionado error por el que se recortaban imágenes instantaneamente
21	Investigacion programacion Kivy (variables) (3/3)
	Mejoras de rendimiento del canvas (fondo, errores)
	Programacion recortes de la app
	Solucion de errores (Recortes, rango del canvas)
	Programa extrapolar coordenadas reales de cada recorte
22	Programar calcular varianza en la app
	Programar RFS02 (Solucion rango del canvas)
23	Cambiar el renombre de recortes para distinguir fila y columna (1/2)
	Investigar RFS13 (eliminacion de puntos tras pasar de imagen)
	Cambiar el renombre de recortes para distinguir fila y columna (2/2)
24	Modificada ruta de trabajo a local para evitar problemas de portabilidad
25	**Añadido RFS11 (funcion que te imprime las coordenadas en fichero)**
	Programar RFS13 (eliminacion de puntos tras pasar de imagen)
	Programa coge como recortes cuadrado pequeño en vez de grande como hasta ahora
26	Programar creacion de matriz
	Solucionado error de la matriz (aparecia vacia)
	Discusion programacion (variables globales, tamaño area grande)
27	**Discusion sobre RFU02 (programacion pop up area pequeña)**
	Programar momento en el que se calcula varianza
	Programar RFS09 (calcular area grande)
	Programar eliminación de Boton Siguiente en el último recorte.
	Discutir solucion encuadrar recortes
28	**Programar RFS12 (cálculo de varianza en cada recorte)**
	Programar RNF07 (borrado de recortes cuando acaba el programa)
	Programar RFS12 (cálculo de varianza en cada recorte)
	Estimacion
	Poner en comun programas realizados
	Solucion de errores en los recortes
	Solucionado problema encuadrar recortes (math.ceil)
29	Analizar programar redimensionar puntos recortes

	Analisis captura de area pequeña
	Analizar programar redimensionar puntos recortes (cpx, cpy)
30	Analizar prioridad y forma de escoger (recorte) a la hora de calcular varianza
	Solucionados fallos con variables globales
	Solucionado fallo con matrices (se empezaban a rellenar por la segunda posicion)
	Programa redimensionar puntos recortes
	Programar RFS11 (salida de coordenadas por fichero)
	Modificado formato de salida de coordenadas por fichero
	Discutido problema con area pequeña y area grande que son lado y no area
31	Inicio de memoria
	Memoria: Documentacion tutoriales e iniciar agradecimientos, resumen e introduccion
	Corregido problema con area y lado
32	**Analisis RFS10 (forma de recorrer recortes - generador) (1/2)**
	Analisis recursos para introduccion
	Analisis de apartados de la memoria
	Analisis RFS10 (forma de recorrer recortes - generador) (2/2)
	Analisis forma de salir antes de finalizar los cálculos
33	Discutidas posibles pruebas para comprobar la eficiencia
	Introduccion realizada
	Creada matriz real de puntos para las simulaciones
	Corrección de citas en memoria
	Corregida introduccion
	Corregidos estilos de la memoria
34	Analisis capítulos de la memoria
	Cambios en la introduccion (Evitar opiniones)
	Añadidos párrafos y fórmulas a la memoria
35	Resolver dudas acerca de memoria
	Cambios en la introduccion (Titulos, orden)
	Analisis Servicio Web
	Analisis programacion cálculo de matrices (Pseudocodigo)
	Analisis otros capítulos de la memoria
36	Analisis requisitos (Libro)
	Corregida introducción memoria
	Tutoriales Flask
	Analisis estilo en la memoria (ecuaciones, ejemplos, referencias, …)
	Analisis de forma de hacer diagrama de Gantt (correo) (1/2)
37	Analisis del diseño del servicio web
	Mejoras memoria
	Analisis añadir imágenes (Memoria)
	Cambios en resto de secciones (ecuaciones, ejemplos, variables)
	Analisis de forma de hacer diagrama de Gantt (correo) (2/2)
38	Analisis tipo de requisitos (libro) para añadirlos
	Aprender a hacer cuadros en Overleaf
	Discutir acerca del modelo de 3 capas (Libro)
	Mejorar requisitos (Estan mal diferenciados)
39	Modulo para las librerias Numpy y Kivy
	Corregidos fundamentos teoricos (memoria) (Libro de estadistica)
	Analisis como hacer resumen, abstract y palabras clave
	Corregidos errores de requisitos
40	Resumen y palabras clave (Memoria)
	Correccion de memoria
41	Repaso de fundamentos (libro?)
	Primera version de Flask (subir ficheros)
	Analisis programa servicio web
	Discutir acerca de la memoria (definiciones, atributos, requisitos)
	Cambios en los ejemplos y en citas (Memoria)
	Analizar caso empresa Lynce para la memoria
42	Avanzar en Flask (Crear base de datos)

	Trabajar en RNF03 (Cambios en servicio web, Permite subir txt, jpg y png)
	Creada base de datos
	Analisis manera de diferenciar si subes que datos, y que hacer dependiendo 1/2
43	**Trabajo en RFU07 (Cambios en la salida de datos)**
	Avances en la base de datos (Crear Clase Datos)
	Memoria: Corregir fallos y añadir ejemplos
	Analisis manera de diferenciar si subes que datos, y que hacer dependiendo 2/2
	Repasar y modificar requisitos uno a uno
	Analisis de modulo para salir del conteo sin recorrer todos los recortes
44	Memoria: Repasar referencias, apartado de herramientas utilizadas.
	Añadidas imágenes a la memoria
	Solucionado error con base de datos (variable t)
	Analisis seguridad (requisitos + sistema nacional de seguridad)
	Añadidos requisitos de seguridad (Para despliegue) (libro Python cookook para implentar utilidades)
	Trabajar en servicio Web
	Referenciar y añadir ejemplos
45	Inicio Bouml para casos de uso
	Base de datos (Atributo descripcion)
	Analisis del programa tras añadir foto o datos
	Memoria: Ejemplos, requisitos de seguridad y casos de uso.
46	Programar servicio web (en funcion de entrada)
47	Analisis base de datos
	Solucionado problema por el que no podia acceder a la base de datos
	Analisis diferenciar varianza real con la nuestra
48	Analisis extrapolar últimos valores de las rejillas no recorridas
	Problemas con servicio web y con comparacion de varianzas
	Solucionado problema condicion base de datos
	Analisis flujo según datos en base de datos
49	Solucionado problema con matriz auxiliar
	Programada servicio web para añadir nuevos datos
	Programada manera en la que se rellena la matriz final
	Servicio web, añadido hueco para descripcion
	Programacion flujo según datos en base de datos
	Solucionado problema al rellenar matriz final
	Cambiar la variable que se muestra de varianza a coeficiente de error
50	Analisis simulaciones vs programa real
51	Problemas y dudas con simulaciones
	Problemas con los datos (Distinto formato)
	Requisitos
	Solucionar problemas de formato de imágenes
52	Correciones Memoria
	Casos de Uso, primer boceto
	Solucionar problemas librerias y software (Cambio de ordenador)
	Solucionar dudas correciones Memoria
	Poner en comun requisitos
	Discutir casos de uso
	Discutir metodologia
	Programa rellena matriz (Domingo)
	Cambio de formato cuadros requisitos
	Casos de uso
	Metodologia
53	Gantt
	Entender y avanzar en el rellena matriz
	Repasar dudas en casos de uso
54	Discuitir diagrama de Gantt

55	Repasar dudas en casos de uso (Garantia minima)
	Pruebas de integracion (rellena matriz)
56	Simulaciones (Eliminar variables no utilizadas)
	Desarrollo Simulaciones
57	Analizar utilidad de Boton Finalizar
	Añadido el funcionamiento de las simulaciones
	Simulaciones. Eliminacion de codigo en Kivy (No utilizado)
58	Explicacion simulaciones
	Pasar resultados de las simulaciones a fichero
59	Simulaciones: Comenzamos a trabajar con el error relativo.
	Obtenidos nuevos datasets para trabajar en las simulaciones
	Simulaciones: Incluida nueva forma de coger coordenadas para datasets con distinta estructura
	Corregido fallo con rutas locales
	Corregido error con ap que a veces era decimal
60	Metodo extrapolar matriz en el programa real
	Solucionados problemas con el rellenamatriz al inicializar a -1 cosa que no hace falta con el generador
	Corregido error matriz extrapolada que se igualaba a la real
	Crear Boton Finalizar (Kivy) y añadida la publicacion del resultado en fichero en caso de usarlo
61	Corregido error con el generador tras probarlo con otra imagen
	RFU04 Completado (Añadido Boton Salir)
	Desarrollado RFU06 (Añadido Boton Volver)
	Selección de datasets mas relevantes para las simulaciones
	Realizar simulaciones
	Discutir posibles errores de concepto de ag y ap
	Pruebas interfaz
	Visto error RFU06 (Boton Volver)
	Discutir los resultados de las simulaciones y mejorarlos
62	Automatizar simulaciones
	Corregir error RFU06 (Ocultar Boton Volver)
	Trabajar en visibilidad de botones (Boton Cargar, Boton Siguiente)
	Añadir pruebas unitarias (Varianza, canvas)
	Corregido error de fichero de salida de pruebas
	Mejora en la automatizacion de simulaciones.
	Añadida Condición para cumplimiento de RFS06
63	Discutir diseño arquitectonico de la aplicación (Capas)
	Iniciar la creacion del metodo de reinicio
	Simulaciones: Ejecutar y analizar
	Correcciones cálculo Varianza
	Problemas con la ejecucion del reset
64	Corregido fallo por el que no se borraban las graficas al automatizar las simulaciones
	Simulaciones ejercutar y agrupar
	Crear Pop up area pequeña y ajustar
65	Solucionado problema con el Boton Cargar y su cambio de funcion a reinicio
	Funcion reiniciar (Fallos)
	Simulaciones, conclusiones
	Solucionar problema de recurrencia con el reset
	Solucionado un problema de implantación de textos con el reset.
	Intentar solucionar un problema con el Filechooser cuando haces el reset (Sin éxito)
66	Problemas de integracion con el pop up del ap
	Añadir Simulaciones a la memoria
	Discutir graficos para la memoria en la parte de diseño y pruebas
67	Discuir realización de pruebas (Unitarias, integración, Sitema y Aceptación)

	Añadir pruebas a la memoria
	Correcciones Memoria
	Discutir errores del programa para avanzar
68	Discuir pruebas del sistema (Requisitos no funcionales)
	Diagrama de flujo (GraphvizOnline)
	Capa de negocio (Diseño)
	Capa de datos
	Correcciones
	Solucionar problema pasar imagen que no fuera jpg
	Discutir dudas correcciones
69	Repaso de memoria (Simulaciones, conclusiones)
70	Memoria: Correcciones y repaso general

Referencias

F. Botta, H. S. Moat y T. Preis. Quantifying crowd size with mobile phone and twitter data. *Royal Society open science*, 2(5):150162, 2015.

N. Calle. Los datos oficiales de manifestantes: así cuentan los cuerpos de seguridad. Enlace, 2017. Última vez visitado: 2019-05-31.

CCN. Esquema nacional de seguridad. https://tinyurl.com/yxanpj4n, 2015. Última vez visitado: 2020-10-19.

M. Cruz y J. González-Villa. Correction: Simplified procedure for efficient and unbiased population size estimation. *PloS one*, 13(11):e0208359, 2018.

M. Cruz y J. Gonzalez-Villa. Simplified procedure for efficient and unbiased population size estimation. *PloS one*, 13(10):e0206091, 2018.

M. Cruz y J. González-Villa. Unbiased population size estimation on still gigapixel images. *Sociological Methods & Research*, page 0049124118799373, 2018.

M. Cruz, D. Gómez y L. M. Cruz-Orive. Efficient and unbiased estimation of population size. *PloS one*, 10(11):e0141868, 2015.

M. H. DeGroot y M. J. Schervish. *Probability and statistics*. Pearson Education, 2012.

C. Grima. Las distancias en manhattan. *metode*, 94, 2017.

H. Idrees, I. Saleemi, C. Seibert, y M. Shah. Multi-source multi-scale counting in extremely dense crowd images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2547–2554, 2013.

JAE. Seguridad de servicios. https://tinyurl.com/yytaz8u8. Última vez visitado: 2020-10-19.

V. Lavrenko. Kd tree algorithm: how it works, 2014. https://www.youtube.com/watch?v=TLxWtXEbtFE

V. Lempitsky y A. Zisserman. *Learning to count objects in images*. Springer, 2010.

R. National Geographic. ¿es posible saber cuántas personas hay en una manifestación? Enlace, 2017. Última vez visitado: 2019-05-31.

T. Oetiker, H. Partl, I. Hyna, y E. Schlegl. The not so short introduction to latex 2. *Or LATEX 2d in*, 69, 2011.

A. P. Mohorte. El mejor modo de calcular cuánta gente hay en una manifestación (y pasar de las cifras oficiales). Enlace, 2017. Última vez visitado: 2019-05-31.

RTVE. La empresa lynce patentó un método para contar a todas las personas de una manifestación, 2012. http://www.rtve.es/alacarta/videos/telediario/empresa-lynce-patento-metodo-para-contar-todas 1332586/

I. Sommerville y V. Campos Olguín. *Ingeniería de Software (Novena edición ed.)*. México: Pearson Educación, 2011.

UCF. A large crowd counting data set, 2011.
https://www.crcv.ucf.edu/data/ucf-qnrf/

Unican. Countem | efficient and unbiased estimation, 2016.
https://countem.unican.es/