

Procesamiento de imágenes digitales

Filtro bokeh basado en la segmentación fondo/figura

Santiago Juan Tarrío Gete , Ángel Franco Álvarez y Francisco Miguel Pérez Canales

Resumen

En este trabajo, se realiza un proceso de detección de múltiples figuras en una fotografía para clasificar el fondo y las distintas figuras, basados en el trabajo previo de otro artículo. Una vez clasificados, se aplica una simulación del efecto bokeh con el que se obtiene la misma fotografía pero con un desenfoque parecido al que hacen las cámaras fotográficas o el ojo humano, en el que la figura principal mantiene su nitidez y el borronado esta basado en la distancia.

Key words: Bokeh, Saliency-detection, Trabajo, Imagen digital

1. Introducción

En este proyecto se pretende abarcar, mediante un objetivo concreto, un amplio abanico de campos de forma transversal, enfocados en el procesamiento de imágenes digitales.

El tema a tratar en este trabajo es el concepto japonés del *bokeh* [1], que se refiere al tipo de desenfoque producido por un objetivo en un sentido estético. En líneas generales, se trata de un tipo de desenfoque en el que la figura principal aparece enfocada, y el resto de elementos de la imagen aparecen desenfocados, bien completamente o bien de una forma escalada, de acuerdo con algún criterio (como por ejemplo la distancia). Para llegar a realizar este desenfoque, en los objetos que estamos fotografiando necesitamos realizar un complejo proceso que detallaremos a lo largo del trabajo para diferenciar lo que es el fondo de las figuras.

El objetivo de la aplicación desarrollada es que, dada una imagen plana y sin conocimiento sobre la profundidad, se puedan estimar los píxeles de la misma como parte del fondo o la figura, para posteriormente poder aplicar un difuminado estético al fondo, simulando así el efecto del bokeh.

Mediante el desarrollo del trabajo se pretende profundizar en diversos conceptos relacionados con el procesamiento de imágenes digitales, como son la segmentación, los filtros en espacio, el tratamiento de diversos espacios de color (RGB, LAB, HSV) o la morfología, así como en otros campos de interés para los integrantes: la algorítmica, los métodos estadísticos y el uso de Python.

2. Planteamiento teórico

El trabajo consta de dos partes, correspondientes a trabajos basados en dos artículos distintos.

La primera parte del proyecto se va a centrar sobre el artículo científico [4]. Este artículo busca desarrollar un sistema que permita segmentar las figuras principales de una imagen diferenciándola del fondo de la

misma, mediante el encadenamiento de una serie de algoritmos conocidos de segmentación y clasificación, que se detallan a continuación.

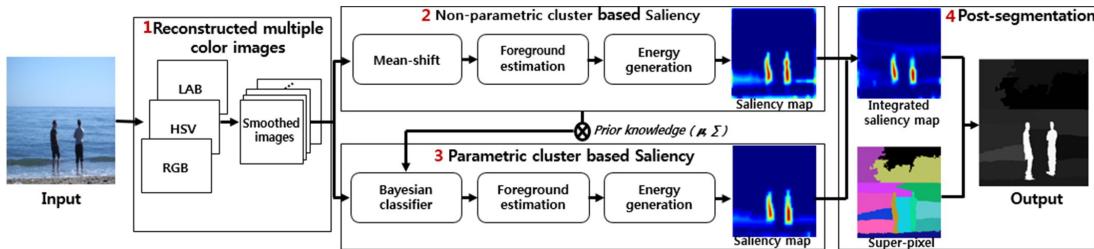


Figura 1. Resumen del algoritmo expuesto en el artículo [4]

La segunda parte en cambio, esta inspirada en el artículo científico [5] y en los resultados propiciados por el artículo mencionado anteriormente [4]. El artículo [5] busca desarrollar un sistema que permite simular el efecto bokeh en una imagen. Decimos que es inspirado, ya que se aproxima el problema desde otra perspectiva pero con el mismo espíritu que en el artículo.



(a) Imagen original sin aplicarle una simulación de bokeh.



(b) Imagen tras aplicarle simulación de bokeh.

Figura 2. Comparativa de las imágenes antes y después. Sacado del artículo [5]

2.1. Preprocesamiento

El algoritmo parte de una figura, que se convierte a 3 formatos: LAB, HSV y RGB. A cada uno de los tres formatos se le aplica un filtro gaussiano con 3 parámetros σ distintos ($\sigma = 1, \sigma = 3, \sigma = 5$). De manera que al final se tienen 9 imágenes con las que trabajar.

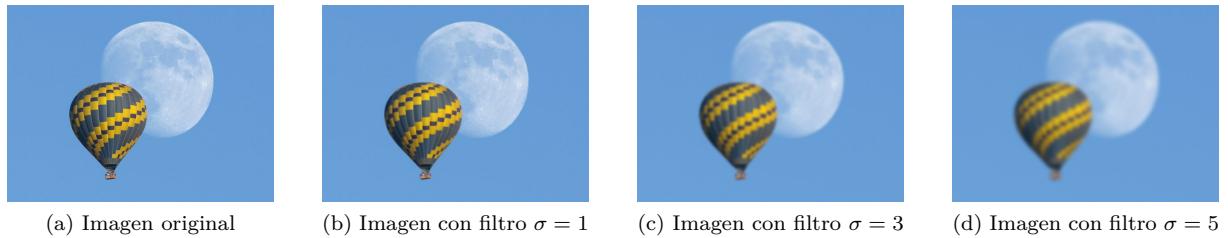


Figura 3. Comparativa de una imagen de prueba con distintos gaussianos.

2.2. Clasificación no paramétrica

2.2.1. MeanShift

A cada una de estas imágenes se le aplica un algoritmo de clasificación llamado MeanShift. [2]. MeanShift es una técnica de análisis de las características del espacio para localizar los máximos de una función de densidad. Con este algoritmo, se pueden encontrar los centroides de colores de una imagen y clasificar los distintos pixeles en un grupo de color segun su cercanía con cada centroide.

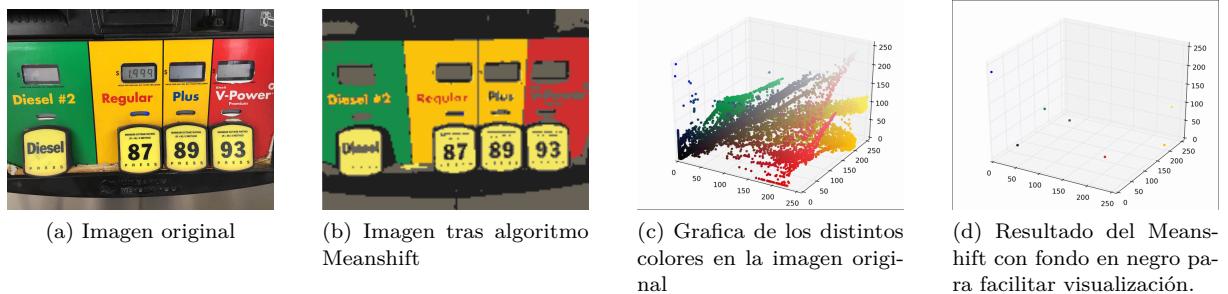


Figura 4. Ejemplo de MeanShift con la imagen completa. Sacado de [6]

MeanShift es computacionalmente $O(Tn^2)$ costoso, donde T es el número de iteraciones y n el número de puntos. Por lo tanto, el tiempo que tarda en procesar todos los puntos se dispara en imágenes de gran resolución, por lo que hemos de sacrificar precisión y trabajar con imágenes reescaladas a aproximadamente 200x200. Este método de clasificación es no pámátrico. Hay dos parámetros libres correspondientes al ancho de banda del kernel que se va a usar en el método. En el artículo principal [4] nos indica que debemos usar el kernel de Epanechnikov. Dicho kernel se define así:

$$k_E = \begin{cases} 1 - x, & \|x\| < 1 \\ 0, & \text{en otro caso} \end{cases} \quad (1)$$

Estos dos parámetros se corresponden a una proporción con respecto a la topología de la imagen en espacio métrico (x, y) y en espacio cromático (la información de color de ese pixel segun si es RGB, HSV o LAB). Por ejemplo para RGB: el primer párametro se corresponde a la diagonal de la imagen, y el segundo depende del formato de la imagen: Para RGB y LAB utilizamos 255 ya que es el rango de color a utilizar, para HSV elegimos un valor entre 0 y 1.

Además de reescalar, también se utiliza solo una pequeña parte de la imagen que coincide con los intersecciones de una máscara en forma de rejilla. Así, se consigue trabajar con algunos puntos de esa imagen escalada. El resultado de Meanshift nos proporciona una clasificación de los píxeles de la imagen, atendiendo a las características espaciales y de color de cada uno de los píxeles agrupandolos en conjuntos parecidos.

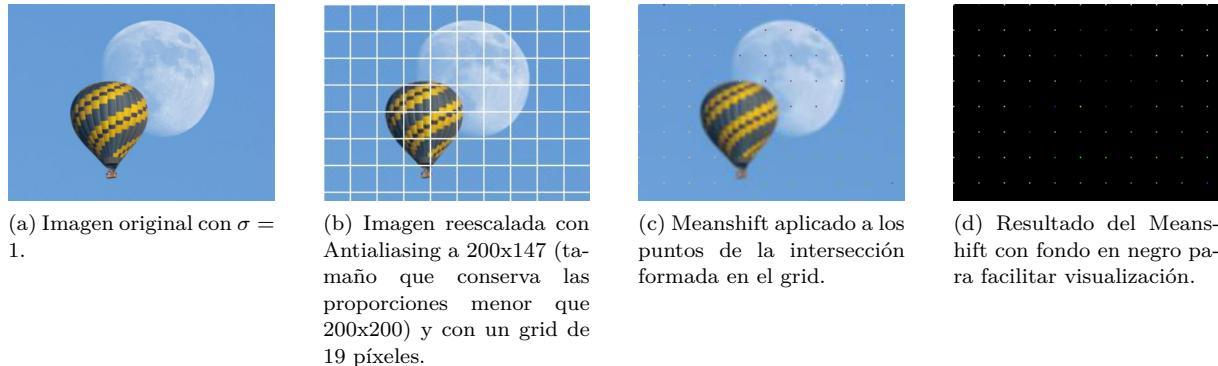
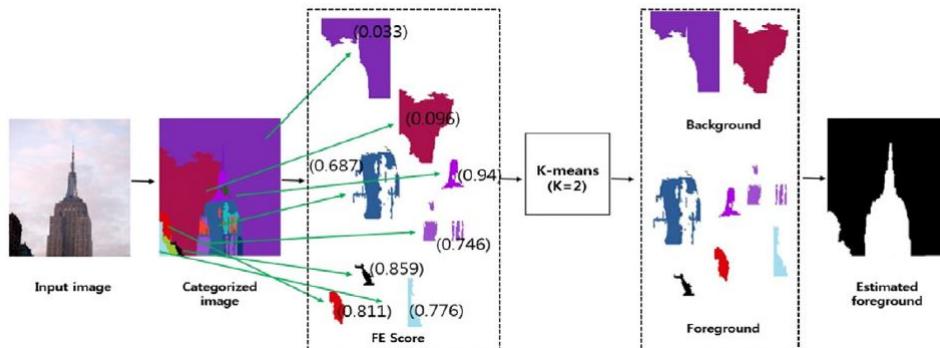


Figura 5. Ejemplo de Meanshift en RGB con $\sigma = 1$.

2.2.2. Foreground Estimation

A las clasificaciones obtenidas con Meanshift posteriormente se le calculará una estimación de pertenencia como fondo, o como figura mediante la interpretación de una serie de características espaciales. Estas características se pueden resumir en que:

- Los objetos figura están por lo general cerca del centro de la imagen y con un menor contacto con los bordes.
- Las variaciones espaciales de los objetos figura son relativamente pequeñas comparadas con los objetos fondo.



Esta clasificación se hace a partir de la siguiente modelización que calcula la pertenencia al fondo o a la figura:

$$FE_i(v_i, b_i) = \exp(-(v_i + b_i)), i \in \{i, \dots, k\} \quad (2)$$

$$v_i(c_i) = \frac{1}{h_{c_i}^d} \sum_{(x,y) \in c_i} (x - \mu x)(y - \mu y), i \in \{i, \dots, k\} \quad (3)$$

$$b_i(c_i, B_r) = \frac{h_{c_i}^d \in B_r}{h_{B_r}^d}, i \in \{i, \dots, k\} \quad (4)$$

La primera ecuación (2) es la que dado un pixel de Meanshift, clasifica si es fondo o si es figura. Valores altos indican que es figura, mientras que valores bajos indican que es fondo. v_i hace referencia a las variaciones espaciales de un cluster, mientras que b_i hace referencia a una puntuación de como esta cerca del centro de la imagen. \exp es la función exponencial y el indice i indica los k clusteres que hay.

La segunda ecuación (3) indica la variación de las coordenadas de un cluster c_i , donde $h_{c_i}^d$ es el área del cluster observado, x e y las coordenadas de ese cluster, μ hace referencia a la media de cada coordenada: si aparece multiplicando a x , es la media de la coordenadas x que pertenecen a ese cluster, y si aparece multiplicando a y , es la media de las coordeandas y que pertenecen a ese cluster. Nuevamente, el indice i indica los k clusteres que hay. Estos valores v_i una vez calculados para todos los k clusters, se han de normalizar entre 0 y 1.

La tercera ecuación (4) indica como esta solapado un determinado cluster c_i a una región B_r . Donde B_r es el borde una imagen en donde el rango r es considerado. $h_{B_r}^d$ es el área conformada por la región B_r y $h_{c_i}^d \in B_r$ hace referencia al area de superposición entre el cluster segmentado c_i y el borde B_r . Este B_r también es normalizado entre 0 y 1 a efectos de calculo.

Cuando este calculo finaliza para cada uno de los clusteres, el algoritmo finaliza y entonces se aplica el algoritmo de K-medias con $k = 2$, para obtener por una parte las figuras, y por otra parte el fondo. K-medias es un algoritmo de clustering en donde se cogen sendos datos y se forman k centroides (que corresponden a k clusteres). Una vez hecho esto, se clasifica cada dato segun el grado de cercanía que se obtiene al aplicarse una función de distancia del dato con el centroide. Un dato pertenecera a un cluster si esta más cerca de su centroide asociado que los demás centroides.

2.2.3. Energy Generation

Una vez hecho el Foreground estimation, se tienen ahora puntos aislados en la imagen y separados por fondo y por figura. Con la familia de puntos que son figura, se construye un grid a a partir de estos puntos aislados. Una vez construido dicho grid, se le aplica una morfología que permite llenar los cuadros cerrados que se forman si estos tienen el tamaño de la celda.

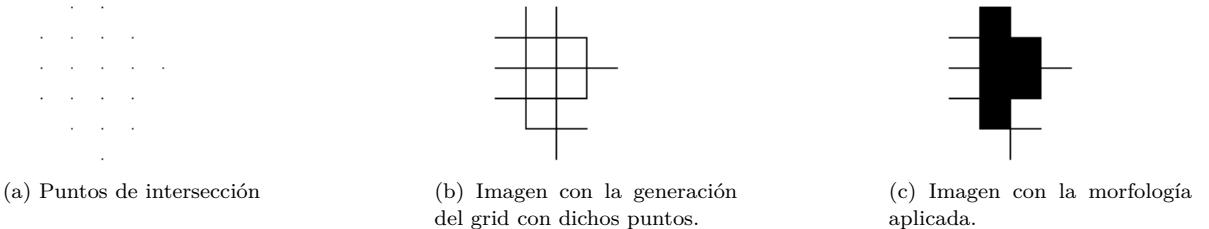
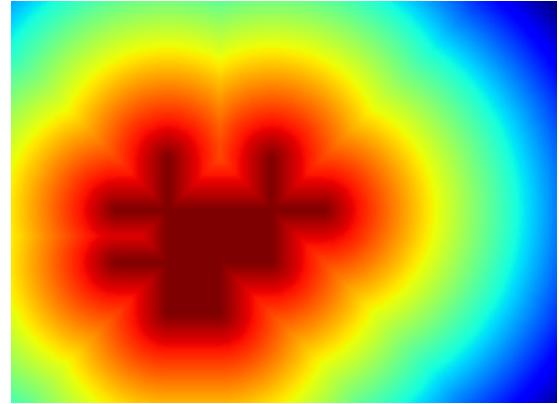


Figura 6. Ejemplo de procedimiento morfológico con los puntos de la figura.

Cuando la morfología esta hecha, se le aplica la transformada de la distancia a la imagen en blanco y negro obtenida del paso anterior, que corresponde al *foreground* estimado. La Distance Transforme (DT) genera un mapa de distancias en el que el valor de cada píxel es sustituido por el valor de la distancia que tiene hasta la frontera más cercana.



(a) Imagen con $\sigma = 1$ pequeña.



(b) Imagen con su transformación de distancia correspondiente.

Figura 7. Ejemplo de Transformación de Distancia en RGB con $\sigma = 1$.

2.3. Clasificador bayesiano

El clasificador bayesiano es una técnica de clusterización paramétrica, así como un método estadístico que se utiliza para la asignación de una etiqueta de clase a un vector de entrada dando la probabilidad correspondiente más alta para la que se utilizan modelos de distribución gaussiana. Por lo tanto, se necesita de un conocimiento previo: un número de clases y vectores de color etiquetados, ambos han sido obtenidos a través del proceso no paramétrico meanshift, y éstos son utilizadas para los cálculos de la matriz de covarianza (Σ) y la media (μ) ; Ahora, los modelos de distribución gaussianos pueden ser definidos usando ambos Σ y μ que corresponden a los vectores etiquetados. Para reducir la variación espacial de resultados de la clasificación, tanto los espacios de color así como las coordenadas espaciales son tomadas en cuenta ya que los pixeles que sobresalen en la imagen deben de ser agrupados alrededor de su centroide tanto como sea posible.



(a) Imagen original.



(b) Clasificador bayesiano

Figura 8. Ejemplo de Clasificador Bayesiano en RGB.

2.4. Foreground y Energy generation al resultado del clasificador bayesiano

Al igual que con el proceso de meanshift, necesitamos un mapa de visibilidad para el cual volvemos a utilizar el proceso de foreground estimation y energy generation.

2.5. Post-Segmentación

Una vez se generan todos los mapas de energía para cada una de las nueve variantes de la imagen (tres *blurs* gaussianos con distintos valores de σ para cada espacio de color), se pueden fusionar todos en uno solo, que representa el mapa de energía estimado para la imagen original. Ésta unión se realiza mediante la media aritmética de los nueve mapas en cada pixel.

Tras obtener el mapa de energía unificado, se crea una segmentación superpixel asociada a la imagen reescalada con la que se ha construido el mapa de energía. Tomando las regiones de cada uno de los superpixeles, se reconstruye el mapa de energía dando un valor en escala de grises similar al mayor valor del mapa de energía en esa región.

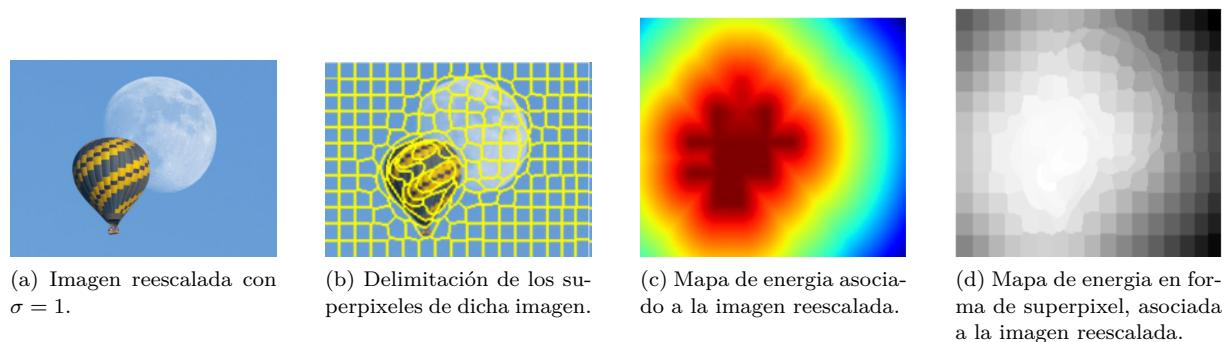


Figura 9. Ejemplo de generación de mapa de energía de superpixel.

2.6. Filtro

En esta sección, se coge el resultado anterior y se realiza el filtro en la imagen original sin reescalar.

Para ello primeramente, se reescala el mapa de energía de superpixel a las mismas dimensiones que la imagen original.

Una vez obtenido el mapa reescalado, se hace el filtro gaussiano adaptado a este. Este filtro se trata de coger el mapa de profundidad y aplicarle 5 thresholdings:

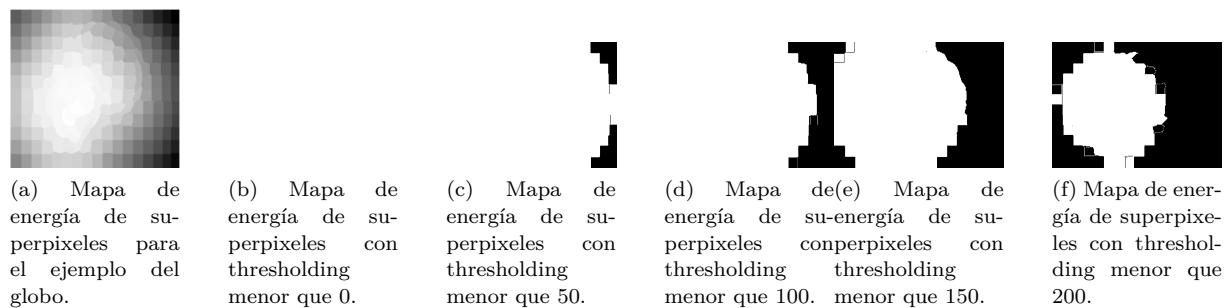


Figura 10. Ejemplo de generación de mapa de energía de superpixel.

Una vez aplicado, se le pega a cada thresholding un gaussiano segun el thresholding que sea. A menor thresholding (es decir, al estar más atrás) se pone un filtro más alto de gaussiano.



Figura 11. Ejemplo de generación de mapa de energía de superpixel.

Una vez se tienen todas estas imágenes, se van pegando una por una encima hasta obtener la imagen con el filtro. Primeramente el gaussiano del fondo, luego la del fondo medio anterior encima, luego la del fondo medio medio encima y así sucesivamente hasta pegar la de la figura. Con el filtro se consigue que estéticamente la figura quede definida, mientras que el fondo progresivamente no.



Figura 12. Imagen final con el filtro aplicado.

Esta imagen, es un ejemplo didáctico, no se nota mucho el difuminado, pero esta presente. En el apartado de experimentación, hay ejemplos mejores del resultado que da este algoritmo.

3. Resolución Práctica

3.1. Organización previa

Para la resolución práctica del trabajo se ha recurrido al lenguaje de programación Python acompañado de toda una familia de librerías para resolver las diferentes fases del desarrollo. Éstas fases pueden dividirse en:

- Desarrollo de la aplicación web
- Desarrollo de los métodos de procesamiento de imágenes
- Desarrollo de los algoritmos con tratamiento matricial de los datos

Para la primera fase, que se abarcará primero por ser parte estructural del trabajo finalmente entregado, se utiliza Django [7], ya que un proyecto Django puede contener las demás librerías de Python utilizadas. Existen muchas librerías para tratamiento de imágenes, de las cuales en el proyecto se usan tres que, conjuntas, cubren todas las funciones necesarias para el desarrollo de la segunda fase. Estas librerías son

Python Imaging Library(PIL), Scikit Image, y OpenCV-Python. Para la tercera fase, donde se trabaja con arrays, matrices y álgebra lineal, se utiliza Numpy, librería de arrays de Python por antonomasia, además de Scipy. ésta tiene un paquete llamado *linalg* que contiene funciones de álgebra lineal, como cálculo de determinantes, matrices de covarianza, normas, etc. También se utiliza la librería Scipy para algunos cálculos concretos. De forma anexa, se usa Matplotlib para tratar con plots.

La razón para definir el desarrollo de la aplicación web como fase 1 es para poder disponer de una suite de pruebas en la que encadenar los distintos pasos del algoritmo a medida que se vayan implementando, sin perder de vista el objetivo final que es el desarrollo de la aplicación. Por tanto, se desarrollará una versión esqueleto de la misma al principio del trabajo, a medida que se vayan estudiando y analizando los conceptos teóricos, para poder integrarlos inmediatamente. Al tener uno de los integrantes experiencia en el desarrollo con Django, se le asigna el desarrollo completo de la fase 1, dejando a los otros integrantes la tarea de estudiar y analizar los fundamentos teóricos y comenzar a trabajar en los algoritmos.

La aplicación desarrollada en Python se desplegará sobre un servidor web de Digital Ocean para que esté disponible el día de la presentación.

El trabajo de desarrollo de los algoritmos se comenzará por el primer artículo [4], por ser el más extenso y el que contiene la funcionalidad principal que se busca. Una vez desarrollada dicha funcionalidad se pasará a implementar el filtro bokeh explicado en el artículo [5]. Ya que este paso va al final del proyecto, es previsible que no se consiga, teniendo que reestimar los objetivos del proyecto. Por tanto, los hitos a cumplir serán:

Hitos:

Santiago	Puesta en marcha de la página web
Ángel y Francisco	Estudio y desarrollo de los algoritmos

3.2. Desarrollo y progreso

Para la organización y gestión del proyecto nos hemos servido de una serie de herramientas y facilidades, como son Telegram para intercambiar mensajes, Discord para mantener reuniones no presenciales, las salas de estudio en grupo de la biblioteca de la ETSII para trabajar de forma presencial y organizar el reparto de tareas, y Trello para gestionar dicho reparto y que sea accesible para todos los integrantes.

La implementación para el algoritmo MeanShift se adaptará de la realizada por Matt Niedrich en su repositorio público [6], sustituyendo el kernel gaussiano por el de Epanechnikov, retocando la condición de parada y agregando el concepto de vecindad, ya que su implementación tiene en cuenta a *todos* los puntos en cada movimiento, lo cual es ineficiente y no más eficaz que considerar un entorno *d* para el cual un punto afecta al movimiento de otro si su distancia es menor o igual a *d*.

Para la completa compresión del algoritmo se ha recurrido a [3], que explica en detalle los conceptos de kernel y función de densidad.

Las implementaciones de los algoritmos de FE, EG, clasificador bayesiano (a pesar de no ser aplicado) y filtro gaussiano adaptativo son todas hechas por el grupo de trabajo.

Se han utilizado librerías de algoritmos para casos en los que no resultaba eficiente desarrollar una solución propia, como son los conocidísimos algoritmos de K-medias o segmentación por superpixel.

La aplicación con Django estará alojada de forma temporal en un VPN de Digital Ocean [8] dónde hemos contratado 2 núcleos, 2GB de ram y 40GB de disco duro. Se podrá acceder a la aplicación desde la dirección IP : <https://165.227.209.234>

Cómo no estabamos seguros de poder ser capaces de desplegar la aplicación, ya que no lo habíamos hecho nunca, también tenemos el trabajo montado en un notebook de Jupyter que contiene todo el proceso listo para ejecutar. No obstante, al final si conseguimos desplegar la aplicación en Digital Ocean [9].

La implementación del algoritmo ha sido parcial debido a una dificultad de computación alta. En vez de hacer los cálculos con 3 filtros de Gauss en RGB, HSV y en LAB; se han hecho simplemente los cálculos con 3 filtros de Gauss en RGB. Además, no se ha aplicado el proceso del clasificador bayesiano debido al

mismo motivo. No obstante, se han obtenido resultados satisfactorios con los calculos de los 3 filtros en RGB (puesto que se ha podido unificar su mapa de energía y obtener un resultado más preciso).

En la aplicación con Django, se describe el proceso aplicando una sola imagen RGB con un nivel de Gauss. En el fichero Jupyter, se describe el proceso aplicado a 3 imágenes RGB con diferentes niveles de Gauss ($\sigma = 1$, $\sigma = 3$, y $\sigma = 5$) y unificado en un solo mapa de energía.

4. Experimentación

En este apartado vamos a proceder a mostrar las pruebas realizadas y a explicar los resultados.



(a) Imagen original



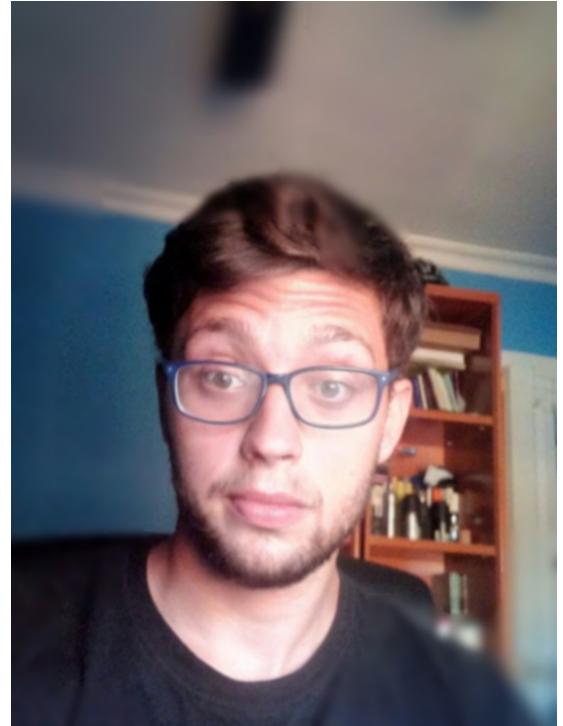
(b) Imagen tras aplicarle el filtro.

Figura 13. Comparativa de las imágenes antes y después del proceso

Este es un ejemplo de un buen resultado. Los parámetros usados fueron: un grid de tamaño de celda 15, un vector de entrada en el Meanshift de [150,255] (el primero, corresponde a la distancia espacial, el segundo a la distancia cromática) y una distancia de entrada en el Meanshift de 50.



(a) Imagen original



(b) Imagen tras aplicarle el filtro.

Figura 14. Comparativa de las imágenes antes y después del proceso

Este es otro ejemplo de un buen resultado. Los paraméetros usados fueron: un grid de tamaño de celda 15, un vector de entrada en el Meanshift de [150,255] (el primero, corresponde a la distancia espacial, el segundo a la distancia cromática) y una distancia de entrada en el Meanshift de 50.



(a) Imagen original



(b) Imagen tras aplicarle el filtro.

Figura 15. Comparativa de las imágenes antes y después del proceso

Este es otro ejemplo de un buen resultado. Los paraméetros usados fueron: un grid de tamaño de celda 20, un vector de entrada en el Meanshift de [100,255] (el primero, corresponde a la distancia espacial, el segundo a la distancia cromática) y una distancia de entrada en el Meanshift de 60.

Procesamiento de Imágenes Digitales



(a) Imagen original



(b) Imagen tras aplicarle el filtro.

Figura 16. Comparativa de las imágenes antes y después del proceso

En este resultado podemos ver como debido a las propiedades topológicas de la imagen, el algoritmo no detecta correctamente las figuras debido a las tonalidades parecidas al fondo. Los parámetros usados fueron: un grid de tamaño de celda 20, un vector de entrada en el Meanshift de [100,255] (el primero, corresponde a la distancia espacial, el segundo a la distancia cromática) y una distancia de entrada en el Meanshift de 40.



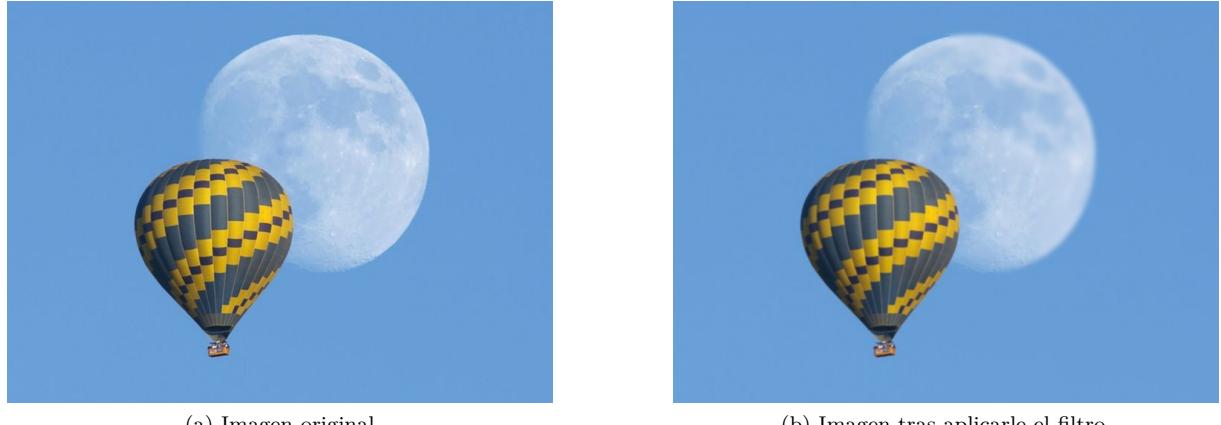
(a) Imagen original



(b) Imagen tras aplicarle el filtro.

Figura 17. Comparativa de las imágenes antes y después del proceso

En este resultado, se puede apreciar que la imagen detecta más como figura lo que es la mesa y la ropa por ejemplo. Esto es porque considera más cerca la mesa que la cabeza, lo cual también es correcto. Sin embargo, no tiene la estética que se busca en el filtro en esta fotografía en concreto. Los parámetros usados fueron: un grid de tamaño de celda 20, un vector de entrada en el Meanshift de [100,255] (el primero, corresponde a la distancia espacial, el segundo a la distancia cromática) y una distancia de entrada en el Meanshift de 50.



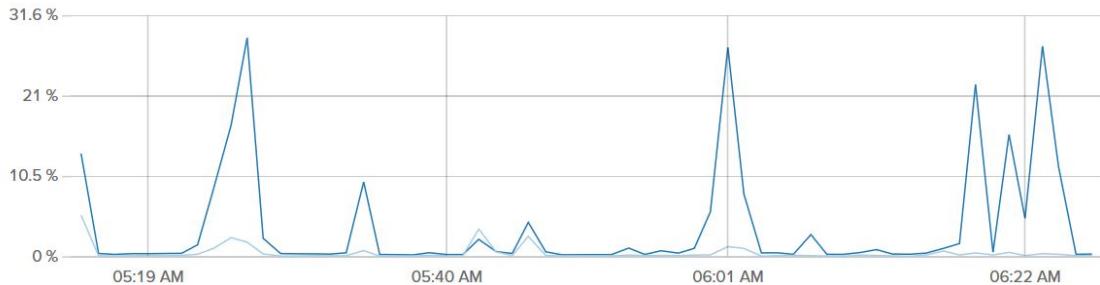
(a) Imagen original

(b) Imagen tras aplicarle el filtro.

Figura 18. Comparativa de las imágenes antes y después del proceso

En este resultado, no se puede apreciar del todo que se haya hecho una difuminación de la distancia debido a que el fondo es bastante homogéneo. Solo se nota que la luna esta algo difuminada. Los parámetros usados fueron: un grid de tamaño de celda 20, un vector de entrada en el Meanshift de [100,255] (el primero, corresponde a la distancia espacial, el segundo a la distancia cromática) y una distancia de entrada en el Meanshift de 40.

CPU



(a) Rendimiento del servidor

Rendimiento del procesador del servidor, el último pico corresponde a la realización de 2 pruebas de manera simultanea. Podemos observar que no hay problemas en la ejecución en paralelo de la aplicación, hasta cierto límite, ya que no se estima que pueda ejecutar con soltura más de 6 pruebas simultáneas (es decir, ejecutando el mismo paso cada vez). Particularmente costosos son los algoritmos de MeanShift y de filtro gaussiano final.

5. Manual de usuario

5.1. Pagina Web

Para facilitar el uso y acceso de la aplicación, se optó por desarrollarla en un entorno web como aplicación de navegador, por lo que los requisitos de uso son mínimos: acceso a internet y un navegador

instalado(pudiendo la página ser accesible y utilizable sin problemas desde plataformas móvil). Existe la posibilidad de que el servidor escogido para el despliegue de la aplicación no funcione o haya caducado, en cuyo caso se puede desplegar como servidor local siguiendo las instrucciones de instalación y despliegue de Django.

Este manual se centrará en el uso y desarrollo de pruebas en la aplicación suponiéndola ya completamente operativa.

La página principal, a la que se accede mediante el enlace <http://165.227.209.234/>, es un portal de bienvenida, que explica el mismo resumen de esta misma memoria y provee dos enlaces principales: el listado de pruebas realizadas y la descarga de este documento.

Más abajo encontramos lo realmente importante: la sección de experimentación. Aquí podemos elegir entre una de las imágenes ya cargadas en el sistema para generar una nueva prueba paso a paso, o subir una de nuestro ordenador para realizar la prueba con ella. No hay un límite en el número de imágenes que puede cargar una persona en el sistema, salvo el que dicta el sentido común (es un servidor gratuito al fin y al cabo).

Una vez determinadas la imagen y los parámetros con los que queremos realizar la prueba, podemos pulsar *ejecutar algoritmo* para comenzar el proceso paso a paso. Éste nos mostrará una a una todas las fases en relación a las imágenes que se obtienen a raíz de ejecutar cada paso, así como un cuadro de información desplegable que explica cada uno de ellos. Al final de la prueba se permite añadir un comentario a la misma, para que conste cualquier observación que se quiera hacer al respecto. La imagen final obtenida o cualquiera de las intermedias se puede descargar pulsando *Clickderecho -> Guardar imagen como...*

Desde el menú de Imágenes el funcionamiento es análogo: se escoge una imagen de las presentes, se definen los parámetros y se pulsa Ejecutar.

En el menú de Pruebas se pueden consultar todas las pruebas realizadas hasta la fecha, comparando la imagen original con el resultado, y mostrando los comentarios. Nótese que cualquier prueba que se inicia genera una nueva instancia de prueba en este menú, se llegue al paso final o no. Futuras implementaciones podrían incluir la opción de reanudar una prueba donde se dejó, pero esta funcionalidad no se incorporó al no ser parte fundamental del proyecto.

Toda esta información se plasmará en un documento README que estará disponible individualmente para futuras consultas.

5.2. Fichero Jupyter

En cuanto al Notebook de Jupyter, este se puede acceder en una carpeta llamada Jupyter mediante el programa Jupyter [10] usando Windows.

Para acceder a la libreta Jupyter llamada Jupyter_executable.ipynb en primer lugar se ha de instalar Jupyter y Python en Windows. Recomendamos instalar Python y Jupyter usando la distribución Anaconda, que incluye Python, el notebook Jupyter y otros paquetes de uso común para computación científica y ciencias de la información.

Primero, descargue Anaconda através del siguiente enlace: https://repo.continuum.io/archive/Anaconda3-5.0.1-Windows-x86_64.exe

En segundo lugar, instale la versión de Anaconda que descargó, siguiendo las siguientes instrucciones:

Localice el instalador y ejecútelo.

Presione continuar (Next) repetidamente para que se instale python y Anaconda con los ajustes por defecto.

Una vez instalado Anaconda, se habrá instalado también Jupyter. Para inicializar el fichero, se ha de ejecutar la linea de comandos en la carpeta donde se haya descomprimido el fichero. Para hacer esto en Windows de una forma más comoda, ha de acceder a la carpeta donde tiene guardada la carpeta 'Jupyter' con el explorador de archivos. Una vez allí, en la barra superior del explorador de archivo (donde sale su ubicación, si está dentro de la carpeta Jupyter, en esa barra saldrá como que la ultima carpeta en esa barra), pulse en una región en blanco dentro de esa barra para modificar el texto. Le saldrá un texto del

estilo: C:\Carpeta1\Carpeta2\... \Jupyter. Borre todo ese texto y escriba cmd. Si todo ha salido bien, entonces tendrá la linea de comandos ejecutada dentro de la carpeta. Dentro de esa linea de comandos, verá que esta dentro de la carpeta. Introduzca el siguiente comando: jupyter notebook

Una vez introducido, pulse Enter y se le ejecutará el proceso que abre jupyter notebook y se le abrirá jupyter en su navegador web. Dentro del navegador web, podrá ver los archivos y en concreto uno llamado Jupyter_executable.ipynb

Pulse en él y se le abrirá una nueva ventana del navegador que le llevará al notebook y ya podrá acceder a este. Si todo sale bien, se encontrara en el notebook y tendrá el siguiente texto informativo:

¡Hola, si has llegado a este archivo es que entonces has instalado Jupyter! ¡Felicidades!

Este fichero .ipynb (formato Notebook de jupyter) está pensando para ejecutar el algoritmo propuesto en la memoria con los cálculos con las 3 imágenes en RGB en diferentes filtros gaussianos y sin realizar la parte de la segmentación Bayesiana.

ADVERTENCIA: Para que este Notebook funcione, deben estar todas las dependencias .py en la misma carpeta que este notebook. Aquí un listado de todos los ficheros que deben estar:

- bayesian.py
- bokeh.py
- energy_generation.py
- foreground_estimation.py
- grid_utils.py
- mean_shift_epanechnikov.py
- mean_shift_utils_epanechnikov.py
- point_grouper.py
- superpixel.py

Además, la imagen con la que se quiere trabajar debe estar en la misma carpeta que este notebook. Por último, se debe usar Windows.

Instrucciones:

1. Datos de entrada

En la celda comentada como Datos, están los datos de entrada del algoritmo, con el siguiente formato de asignación:

parámetro = dato_a_asignar

'parámetro' es la variable interna que usará el algoritmo para ejecutar el código.

'=' es un operador que relaciona el 'parámetro' con el 'dato_a_asignar'

'dato_a_asignar' es el dato en concreto que será asociado a dicho parámetro.

Datos tenidos en cuenta en el algoritmo:

1. nombre_imagen

Este dato hace referencia al nombre de la imagen con su extensión que se le quiere pasar como entrada al algoritmo. Si por ejemplo se desea hacer la prueba de ejecución del algoritmo con el fichero imagen.png se debe cambiar la linea de la asignación de este parámetro por:

nombre_imagen = 'imagen.png'

Los dos hacen referencia a que se está asignando un valor String en Python a la variable nombre_imagen, que será usado por la aplicación.

NOTA: Solo se acepta formato .png y .jpg. Aunque se acepta formato .jpg, no se acepta formato .jpeg, tenga cuidado.

Para pasar una imagen de un formato .jpeg a formato .jpg abra esa imagen con paint y posteriormente guardela como .jpg (esto suponiendo que está en Windows y posee la herramienta paint, y de hecho, como

Procesamiento de Imágenes Digitales

recordatorio, debe de estarlo para que el programa funcione)

2. tamano_celda

Este dato hace referencia al tamaño de celda que se le quiere pasar como entrada al algoritmo. Si por ejemplo se desea hacer la prueba de ejecución del algoritmo con el tamaño de celda 20 se debe cambiar la linea de la asignación de este parámetro por:

```
tamano_celda = 20
```

Esta asignación es la de un numero entero a la variable tamano_celda, que será usado por la aplicación. Es MUY IMPORTANTE que este valor no posea decimales, debe ser entero. Por ejemplo esto NO sería valido:

```
tamano_celda = 20.3
```

3. kernel_bandwidth

Este dato hace referencia al vector de banda ancha que se le desea pasar para ejecutar el algoritmo meanshift. Posee dos componentes: la componente de la izquierda, hace referencia a la distancia espacial; y la componente de la derecha, hace referencia a la distancia en el espacio de colores: [*distancia_espacial, distancia_cromatica*]. Si por ejemplo se desea hacer la prueba de ejecución del algoritmo con el kernel_bandwidth de vector [100, 255] (100 distancia espacial, 255 distancia cromática) se debe cambiar la linea de la asignación de este parámetro por:

```
kernel_bandwidth = [100, 255]
```

Los corchetes indican que el dato a asignar es un array, y la coma, es su delimitador de elementos.

4. distancia

Este dato hace referencia a la distancia que se le desea pasar para ejecutar el algoritmo meanshift. Si por ejemplo se desea hacer la prueba de ejecución del algoritmo con la distancia a 40 se debe cambiar la linea de la asignación de este parámetro por:

```
distancia = 40
```

Esta asignación es la de un numero entero a la variable distancia, que será usado por la aplicación. Es MUY IMPORTANTE que este valor no posea decimales, debe ser entero. Por ejemplo esto NO sería valido:

```
distancia = 40.3
```

2. Sobre como editar la celda de datos para introducir los datos

Para editar la celda de datos de entrada, pinche en la celda donde pone #DATOS con click izquierdo. Posteriormente, debe editar los datos segun como se ha explicado antes en la sección de arriba.

3. Sobre ejecutar el algoritmo

Antes de ejecutar el algoritmo asegurese de que los datos que desea introducir en la celda marcada en la primera fila como #DATOS estan correctos y que los requisitos que se cumplen en la ADVERTENCIA están cumplidos. Si es asi, para ejecutar este algoritmo, debe darle click izquierdo a la celda marcada en la primera fila como #DATOS para seleccionarla, y posteriormente, mantener shift y pulsar enter una vez. Si todo esta correcto, la celda se evaluará, los datos de la celda se guardaran y le aparecerá un mensaje que le dirá lo siguiente:

¡Se han insertado los datos correctamente!

Una vez evaluada esa celda, debe hacer click en la celda marcada como #ALGORITMO en la primera fila y mantener shift y pulsar enter de nuevo para evaluarla; con esto se ejecutara el algoritmo. Este terminará cuando de como respuesta el mensaje de:

¡TERMINADO!

Cuando el algoritmo finaliza, se almacenan metadatos del algoritmo en la carpeta metadatos, y en la carpeta resultados, se almacena el resultado con un nombre parecido a este:

nombre_imagen_gaussiano_segun_superpixel.png

Asi por ejemplo, si el nombre_imagen es 'imagen.png', el resultado se almacenara como:

imagen_gaussiano_segun_superpixel.png

NOTA: El programa esta diseñado para crear la carpeta metadatos y la carpeta resultados en Windows si no están presentes. No obstante, es probable que pueda fallar. Si el programa por el motivo que sea no crea dos carpetas en el mismo sitio donde esta este fichero Notebook, creelas a mano pinchando en la ubicación de la carpeta de este archivo con click derecho y seleccione el desplegable ‘Nuevo’ y acto seguido ‘Carpeta’. Así le aparecerá una carpeta que podrá renombrar como metadatos. El mismo procedimiento puede hacer de nuevo para crear la otra carpeta de resultados si está no ha aparecido.

6. Conclusiones

De la realización de este proyecto se han extraído multitud de conclusiones sobre una serie de elementos, factores y aspectos del mismo, que se detallan a continuación:

6.1. Sobre la experimentación

De la experimentación se puede concluir que los algoritmos, contra todo pronóstico, funcionan como se supone que lo hacen en el artículo en mayor o menor medida. Si bien las pruebas parciales, cuando aún no se disponía de toda la cadena de algoritmos eran muy poco esperanzadoras, paso a paso se ha ido confirmando que el sistema funciona de la manera esperada y arroja resultados prometedores. Otra conclusión tremadamente positiva es el conocimiento adquirido sobre algoritmos y tratamiento de datos, ya que se ha extraído de todas las pruebas el equilibrio necesario entre representación y eficiencia: una mayor cantidad de datos representativos del problema a atacar supone un mayor coste en tiempo y recursos para resolverlo. En nuestro caso, el parámetro de tamaño de grid ha resultado decisivo para aproximar soluciones aceptables.

6.2. Sobre el artículo escogido

En este caso las conclusiones no son tan positivas, ya que todo el equipo se arrepiente de haber elegido tan poco meditada el artículo [4]. El artículo en particular no revisa su propia documentación ni referencias, copia al pie de la letra algunos pasajes y fórmulas, que encima copia mal, induciendo a error y propiciando retrasos de días en algunas implementaciones. En general parece poco revisado y excesivamente enrevesado. Había disponibles otros artículos que trataban el tema de *Saliency – detection*, con menos pasos y vueltas, pero al descubrir la gargantuesca tarea que se avecinaba, ya habíamos empezado a trabajar. La impresión general de este artículo es agridulce: Parece bien fundamentado y eficaz, pero la ejecución deja que desear.

6.3. Sobre las técnicas algorítmicas aplicadas a procesamiento de imágenes digitales

No se recomienda abordar el procesamiento de imágenes digitales con un procesador Intel i5 asmático que, si bien hace lo que puede, es completamente ineficiente en estos casos. El campo, sin embargo, es absolutamente fascinante y abre un sinfín de incógnitas y curiosidades, sobre todo en la proverbial Edad de Oro de las redes neuronales artificiales, donde cada vez se desarrollan aplicaciones del procesamiento de imágenes más avanzadas y eficientes. Tristemente, todo este progreso está lejos de los pequeños ordenadores domésticos, y se lleva a cabo en enormes CPDs por parte de las grandes empresas del sector.

6.4. Sobre las tecnologías utilizadas

Ninguno de los integrantes del equipo era ajeno a Python antes de este trabajo, y por eso se eligió como pila tecnológica. Sin embargo, sí que se puede apreciar la relativa lentitud del lenguaje en comparación

Procesamiento de Imágenes Digitales

con otros más cercanos a la máquina como pueden ser C++ o incluso Java. Sin embargo, la facilidad de programación, la claridad de código y la extensa documentación suman puntos a favor de Python, sobre todo para personas ya habituadas a usarlo. La adición de Django a la pila como framework web ha resultado ser un acierto, a pesar de que, como siempre en el desarrollo software, se pasa más tiempo depurando errores de dependencias que trabajando en el proyecto.

6.5. Sobre la gestión del tiempo y el trabajo

A pesar de lo prolongado en el tiempo del proyecto, se han encontrado serias dificultades para mantenerlo ajustado en tiempo. Las interferencias con otras materias, las dificultades presentadas por el artículo principal (ya expuestas tres subsecciones atrás), y los problemas de coordinación, han mantenido al equipo luchando por realizar un entregable de calidad acorde a lo esperado. Como conclusión se extrae lo esperado: la planificación no ha sido buena, la coordinación ha sido nula hasta el momento en que se acercaba la entrega.

6.6. Sobre mejoras y extensiones en la dirección definida en este proyecto

Las mejoras disponibles son evidentes: de nueve ramas en las que parte el algoritmo del artículo original, nuestro trabajo ejecuta tres en el caso de la libreta de Jupyter y una en el caso de la página web, a modo de demostración. Una mejor organización del código, puramente enfocada a la demostración del algoritmo, facilita enormemente la tarea de ramificar el problema como se describe. Otra mejora importante sería la implementación, tras la generación del mapa estimado de profundidad, un verdadero filtro de emborronado basado en la distancia, como el que era objetivo inicialmente.^[5] También se recomienda la optimización del código para evitar repetir cálculos, sobre todo en lo que a la aplicación web se refiere.

6.7. Sobre la experiencia de realización del trabajo

Las conclusiones generales son positivas, el trabajo ha sido tremadamente interesante y el desarrollo, aunque accidentado, ha sido cuanto menos divertido e inspirador.

7. Referencias

Referencias

- [1] Bokeh, <https://es.wikipedia.org/wiki/Bokeh>
- [2] Mean shift: a robust approach toward feature space, artículo científico sobre meanshift <http://ieeexplore.ieee.org/document/1000236/>
- [3] Introduction To Mean Shift Algorithm, post de blog sobre el algoritmo MeanShift <https://saravananthirumuruganathan.wordpress.com/2010/04/01/introduction-to-mean-shift-algorithm/>
- [4] Artículo científico Detection of multiple salient objects through the integration of estimated foreground clues <http://www.sciencedirect.com/science/article/pii/S0262885616301238>
- [5] Artículo científico Photographic Depth of Field Blur Rendering <https://otik.uk.zcu.cz/bitstream/11025/11225/1/Cyril.pdf>
- [6] Repositorio Simple implementation of mean shift clustering in python https://github.com/mattnedrich/MeanShift_py
- [7] Django website, Documentación del framework Django para desarrollo web en Python. <https://www.djangoproject.com>
- [8] Servidor DigitalOcean, Proveedor de servicio de alojamientos. <https://www.DigitalOcean.com>
- [9] How To Deploy a Local Django App to a VPS, Tutorial despliegue de aplicación. <https://www.digitalocean.com/community/tutorials/how-to-deploy-a-local-django-app-to-a-vps>

- [10] Jupyter, Página web de Jupyter. <http://jupyter.org>
<https://www.digitalocean.com/community/tutorials/how-to-deploy-a-local-django-app-to-a-vps>