Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Selección de procedimientos de tratamiento digital de imagen en aplicaciones de detección de defectos en carriles-bici y una propuesta de clasificación mediante Big Data

Autor: Francisco Javier Salvado de la Llave

Tutor: Francisco Valderrama Gual

Dpto. Ingeniería Gráfica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2018

Departamento de Ingeniería Gráfica ETSI / ETSIA / ETSIE /

Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Selección de procedimientos de tratamiento digital de imagen en aplicaciones de detección de defectos en carriles bici y una propuesta de clasificación mediante big data

Autor:
Francisco Javier Salvado de la Llave

Tutor:
Francisco Valderrama Gual
Profesor Titular de Universidad

Dpto. Ingeniería Gráfica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2018

Trabajo Fin de Grado: Selección de procedimientos de tratamiento digital de imagen en aplicaciones de detección de defectos en carriles bici y una propuesta de clasificación mediante big data

Autor: Francisco Javier Salvado de la Llave

Tutor: Francisco Andrés Valderrama Gual

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2018

El Secretario del Tribunal

A mis padres

Resumen

En este documento se describen, en primer lugar, una serie de procedimientos de tratamiento digital de imagen, a llevar a cabo con el fin de conseguir automatizar la detección de defectos en señales horizontales de carriles bici. Para ello, haremos uso de imágenes procedentes del carril bici de la ciudad de Sevilla, y realizaremos el tratamiento digital de estas a través del software MATLAB.

Por otra parte, se dará otra posible solución al mismo problema desde una perspectiva totalmente distinta. Propondremos la creación de una aplicación a través de la cual los ciudadanos puedan informar ellos mismos al Ayuntamiento sobre cualquier tipo de incidencia relacionada con la vía ciclista. Esta aplicación almacenará los datos que posteriormente analizaremos desde un enfoque big data, para sacar conclusiones al respecto. Para ello, utilizaremos dos softwares de visualización de datos: Carto y Tableau.

Finalmente, hablaremos de Hadoop MapReduce y Apache Spark, dos frameworks de computación en clúster que podremos utilizar para procesar nuestros datos de manera distribuida.

De todo ello se concluye que, una buena forma de automatizar la detección de señales desgastadas, es realizando una comparación pixel a pixel entre la fotografía real de la señal y una imagen ideal dónde la señal se encuentre en perfecto estado. Para ello, las fotografías deben ser tomadas respetando una serie de normas, como son la luminosidad, o la distancia y ángulo que hay entre la lente de la cámara y la propia señal.

Por otro lado, también se concluye que softwares como Carto y Tableau nos pueden proporcionar una buena alternativa al tratamiento con MATLAB. Para ello, se necesitaría que la ciudadanía participase activamente en labores de recolección de datos.

Por último, destacar MLib, una librería de machine learning que nos puede abrir una vía de desarrollo interesante, en la que nuestro sistema, además de funcionar mediante computación en clúster, sería capaz de aprender a discernir entre señales en buen y mal estado.

Índice

Resumen **ix**

Índice de Figuras **xiii**

1 Introducción **15**
1.1 La imagen digital *15*
1.2 Procesamiento digital de imágenes *16*
1.3 Visión Artificial *16*

2 Objeto del trabajo **17**
2.1 Señales a tratar *17*

3 Alcance y metodología **19**
3.1 Alcance y comparación con otros proyectos *19*
3.2 Metodología *20*
3.2.1 Toma de imágenes 20
3.2.2 Software utilizado para el procesamiento de las imágenes 21

4 Procesamiento de las imágenes del carril bici **22**
4.1 Clasificación previa al procesado *22*
4.2 Eliminación automática de regiones no interesantes de la imagen *23*
4.3 Reconocimiento de formas aplicado a señales del carril bici *27*
4.4 Clasificación mediante umbralización de la imagen y conteo de píxeles *29*

5 Localización de zonas afectadas mediante big data **34**
5.1 Recogida de datos referentes al carril bici a través de una supuesta aplicación móvil *34*
5.2 Visualización de los datos con Carto *36*
5.3 Visualización de los datos con Tableau Software *39*
5.4 Propuestas de procesamiento mediante Hadoop MapReduce y Apache Spark *44*
5.4.1 Proyecto Apache Hadoop 44
5.4.2 Apache Spark 45
45

6 Conclusiones **47**

Bibliografía y referencias **49**

ÍNDICE DE FIGURAS

Figura 1: Recorte noticia Viva Sevilla 21 febrero 2018 1
Figura 2: Computer Vision 2
Figura 3: Señal de mediana 3
Figura 4: Señal semáforo 3
Figura 5: Señales de flechas 4
Figura 6: Señal de stop 4
Figura 7: Logo Matlab 7
Figura 8: Degradación similar en ambos carriles 8
Figura 9: Región de interés 9
Figura 10: Primera forma de eliminar elementos externos a la región de interés 10
Figura 11: Puesta a cero de píxeles no verdosos 10
Figura 12: Coordenadas xy en polares 11
Figura 13: Matriz de votos 11
Figura 14: Detección automática de las líneas que delimitan la calzada ciclista 12
Figura 15: Puesta a cero de píxeles fuera del carril 12
Figura 16: Características SURF en imagen a procesar 13
Figura 17: Características SURF del objeto a detectar 13
Figura 18: Coincidencia de características SURF entre imágenes 14
Figura 19: Imagen original 15
Figura 20: Imagen tras *rgb2gray* *15*
Figura 21: Concepto de umbral 15
Figura 22: Imagen binaria resultado de umbralización 16
Figura 23: Comparación al detalle entre imagen original y binarizada 16
Figura 24: Imagen real e imagen ideal de punta de flecha 17
Figura 25: Clasificación en dos grupos 18
Figura 26: Clasificación en tres grupos 18
Figura 27: Imagen binarizada junto a imagen binarizada tras eliminación de zonas no pertenecientes al carril 18
Figura 28: Puesta a cero de los píxeles del margen superior 19
Figura 29: Portal de datos abiertos del Ayuntamiento de Sevilla con datos referentes a la instalación de aparcabicicletas 21
Figura 30: Página principal de Carto 22
Figura 31: Cargar datasets en Carto 22
Figura 32: Dataset de ejemplo creado manualmente en Excel 23

Figura 33: Dataset cargado en Carto 23
Figura 34: Mapa creado en Carto con incidencias producidas en cada distrito 24
Figura 35: Tableau Software 25
Figura 36: Logo Tableau Software 25
Figura 37: Fuente de datos de Tableau 26
Figura 38: Latitud y longitud como roles geográficos en Tableau 26
Figura 39: Latitud y longitud como dimensiones en Tableau 27
Figura 40: Visualización de incidencias sobre un mapa en Tableau 27
Figura 41: Puntos en el mapa con diferentes colores en función del distrito 28
Figura 42: Número de incidencias por distrito en Tableau 28
Figura 43: Número de incidencias por distrito sobre gráfico de barras de Tableau 29
Figura 44: Módulos básicos del ecosistema Hadoop 30
Figura 45: Logo Apache Spark 31

1 INTRODUCCIÓN

Actualmente, se vive una situación de proliferación a nivel nacional de planes y actuaciones relacionadas con la incorporación de vías ciclistas a nuestras calles, prueba de ello es la formulación del Plan Andaluz de la Bicicleta aprobado el 11 de diciembre de 2012 por el Consejo de Gobierno de la Junta de Andalucía.

La bicicleta ofrece una buena alternativa como medio de transporte y esto se debe, en gran medida, a que es un medio de transporte económico y saludable. La vía ciclista toma por tanto un hueco en nuestras ciudades, donde hasta hace unos días solo estaba ocupado por el vehículo a motor.

Como veremos posteriormente, en el presente trabajo se hace uso de la vía ciclista construida sobre la ciudad de Sevilla para llevar a cabo la tarea que se pretende, siendo esta la ciudad con mejores vías ciclistas del país según asegura la Organización Nacional de Usuarios (OCU), en un informe en el que el 8% de los encuestados dice usar la bici como su principal medio de transporte.

Uno de los puntos más importantes de tal informe en cuanto a lo que nos concierne, es aquel en el que se relaciona la calidad de la vía ciclista con la seguridad que esta nos ofrece, y es ahí donde entra en juego la señalización, punto clave, junto con la amplitud de la vía, para lograr dicha seguridad.

En la actualidad, Sevilla cuenta con aproximadamente 180 km de carril bici, el cual requiere, entre otras cosas, de señalización tanto vertical como horizontal. Dicha señalización requiere de un mantenimiento continuo que asegure la calidad de la vía y la seguridad de los ciudadanos.

viva MIÉRCOLES, 21 DE FEBRERO DE 2018 3

Sevilla

PUERTA OSARIO

Desmantelan la terraza de un bar que carecía de permisos

SEVILLA | Un dispositivo desplegado por el Ayuntamiento de Sevilla ha retirado este martes todos los elementos y el mobiliario instalados en la terraza de un establecimiento hostelero de la Puerta de Osario, al carecer todos ellos de autorización. Entre los elementos retirados, todos sin licencia, estaban trece mesas, 47 sillas, tres estufas verticales, un portamenú, dos sombrillas, siete cortavientos, seis tubos soporte de toldos y 16 piezas atornilladas al pavimento para anclaje de los cortavientos.

MOVILIDAD La estructura permitirá conectar el Aljarafe con Sevilla y se completará con la conexión municipal con la Avenida Juan Pablo II

Ya hay pasarela ciclista en la SE-30

LICITACIÓN_Queda pendiente el acuerdo con el Ministerio de Defensa para invertir casi 300.000 euros

CHARCO DE LA PAVA_Su conexión con otra pasarela ciclopeatonal está en estudio y tendrá 550 metros

LA ESTRENADA_La estructura salva la ronda de circunvalación y permite el tránsito de bicis y de peatones

Figura 1: Recorte noticia Viva Sevilla 21 febrero 2018

1.1 La imagen digital

Una imagen digital puede ser obtenida, o bien por algún dispositivo como por ejemplo una cámara fotográfica digital, que contiene un sensor en su interior el cual realiza una conversión analógico-digital para convertir luz en una señal eléctrica proporcional a la cantidad de fotones incidentes, almacenando esta información en una memoria, o directamente creada por nosotros mismos mediante un programa de ordenador.

La imagen digital está compuesta por una matriz de píxeles (acrónimo del inglés picture element), que son la mínima unidad de color que componen la imagen. Cada uno de estos píxeles, normalmente estará codificado en 1 byte de información, tomando un valor entre 0 y 255. De esta manera, cada píxel podrá representar 256 colores distintos, o niveles de gris si la imagen es en blanco y negro.

En una imagen digital, la resolución o el número de píxeles por unidad de superficie, se relacionan directamente con su calidad. Esto es fácil de entender si pensamos que, a menor cantidad de píxeles, menos información. Podemos llegar a pensar entonces que cuanta más información tengamos, mejor, pero tenemos que tener en cuenta que necesitaremos más memoria y aumentaremos el tiempo de procesado en el caso de que queramos tratar la imagen.

Matemáticamente, una imagen digital bidimensional en escala de grises puede ser expresada de la siguiente manera:

$$f(i,j) = \begin{pmatrix} V_{11} & \cdots & V_{1n} \\ \vdots & \ddots & \vdots \\ V_{m1} & \cdots & V_{mn} \end{pmatrix}$$

Para el caso de una imagen digital bidimensional en color, tendríamos tres matrices, dónde cada una de ellas contendría los niveles de intensidad de cada uno de los colores primarios, rojo, verde y azul, en el caso de que queramos hacer uso de este espacio de color (RGB) en concreto. Este espacio de color es el más usado en monitores, proyectores, etc.

1.2 Procesamiento digital de imágenes

Conjunto de técnicas y procedimientos que se aplican a la imagen digital para conseguir una mejora de la imagen. Por tanto, el resultado es una modificación de la propia imagen. Por otro lado, una mejora de la imagen, no siempre se tiene que corresponder con un aumento de su calidad, habrá veces que únicamente nos interese buscar cierta información y tengamos que tratar la imagen con el único objetivo de que nos facilite la búsqueda de dicha información.

El tratamiento puede ser tanto en el dominio espacial como el frecuencial. En el dominio espacial en procesamiento se basa en la manipulación directa de intensidades de píxeles en la imagen, en función de sus píxeles vecinos. En el dominio frecuencial, se realiza una modificación de la transformada de Fourier de la imagen.

1.3 Visión Artificial

Existen varias definiciones, pero atendiendo a la definición dada por la BMVA (British Machine Vision Association and Society for Pattern Recognition), podemos definir la visión artificial o visión por computador como la extracción automática, el análisis y la comprensión de información útil de una sola imagen o una secuencia de imágenes. Los humanos usan sus ojos y su cerebro para percibir visualmente el mundo que los rodea. La visión por computador es la ciencia que pretende brindar una capacidad similar, si no mejor, a una maquina o computadora.

Computer Vision System Toolbox es el producto que nos ofrece Matlab para trabajar con visión artificial.

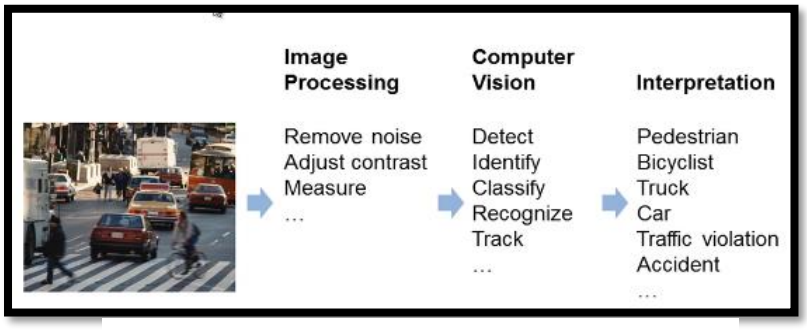


Figura 2: Computer Vision [1]

2 OBJETO DEL TRABAJO

La señalización horizontal está pintada sobre la vía y dicha pintura sufre un desgaste a causa de la interacción con el medio que hace que la señal desaparezca con el tiempo. Con la intención de agilizar la labor de mantenimiento, buscamos un método que automatice al máximo la detección de defectos en señales horizontales concretas, y que nos permita saber qué nivel de degradación sufre la señal, para poder actuar al respeto.

Para ello, haremos uso de MATLAB, un software que cuenta con tool boxes tanto de procesamiento de imagen como de visión artificial, que nos facilitarán esta tarea.

Finalmente, en el punto 5, daremos otra posible solución a la detección de fallos en carriles bici que nada tiene que ver con la visión artificial, ya que abordaremos este punto aplicando soluciones big data.

2.1 Señales a tratar

La Consejería de Fomento y Vivienda de la Junta de Andalucía elabora un texto titulado "Recomendaciones de diseño para las vías ciclistas en Andalucía", en el cual, en su versión 11 de junio de 2013, en el apartado señalización, vienen recogidas las tipologías que deberán seguir las señales.

A continuación, se presentan algunas de ellas.

Marca Longitudinal	
Línea de Separación de sentidos en vías ciclistas de doble sentido	
Ubicación	En el eje de la vía ciclista.
Diseño	En tramos urbanos: marca discontinua de trazos de 1 m separados por vanos de 1 m, con una anchura de 10 cm. En tramos interurbanos: marca discontinua de trazos de 1 m separados por vanos de 2.5 m, con una anchura de 10 cm. Para curvas sin visibilidad la marca será continua.
	

Figura 3: Señal de mediana [2]

Señalización Horizontal sobre Pavimento (Propuesta)		
Semáforo.		
Ubicación	Próximo de una intersección regulada con semáforos.	
Diseño	Inscrito en un rectángulo de 80 cm de alto por 30 cm de ancho.	

Figura 4: Señal semáforo [2]

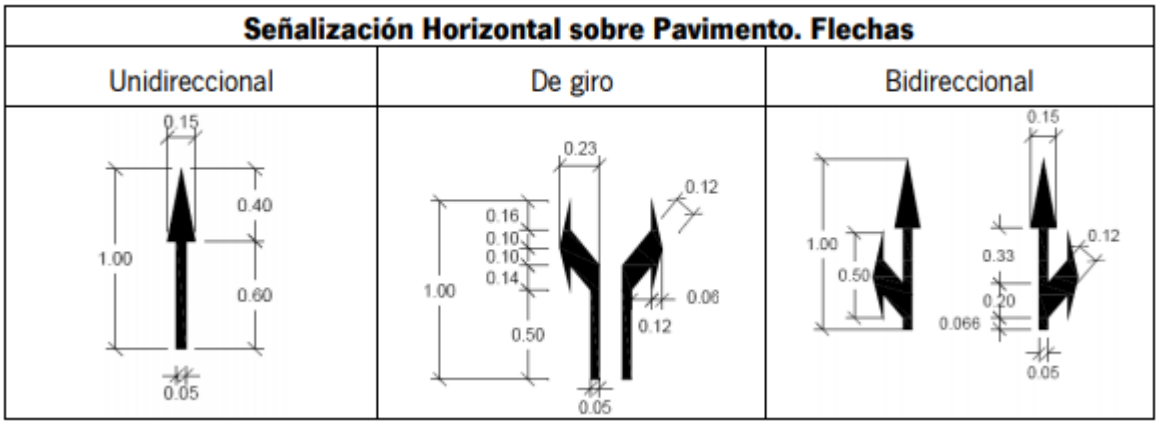


Figura 5: Señales de flechas [2]

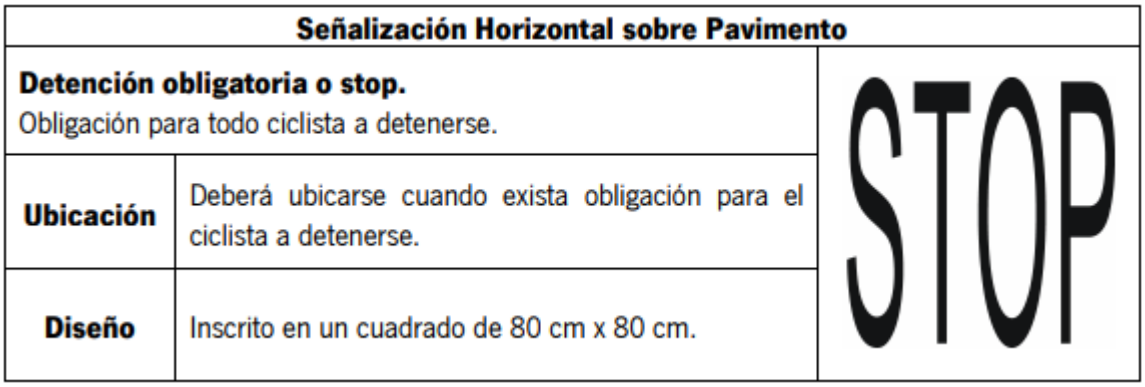


Figura 6: Señal de stop [2]

Estos solo son algunos ejemplos. Actualmente existen multitud de señales funcionando sobre la vía ciclista, algunas más usadas que otras, y por lo que parece, puede que se incluyan algunas más. A principio de 2017, el Ayuntamiento de Sevilla, impulsado por la demanda de padres y madres de alumnos del colegio Jardines del Valle, anunció el inicio de una campaña de señalización especifica en los carriles bici que discurren en entornos escolares.

La idea en este proyecto es trabajar con algunas de las señales que más se repiten a lo largo de la vía. El tratamiento a llevar a cabo con otras señales será muy parecido al que tratemos y totalmente extrapolable.

3 ALCANCE Y METODOLOGÍA

3.1 Alcance y comparación con otros proyectos

El presente trabajo desea servir de aporte a uno mayor, que pretende ser un programa de detección automática de anomalías en vías ciclistas. Sin duda, es una propuesta ambiciosa, como lo demuestra el proyecto llevado a cabo desde 2009 por el Departamento de Ingeniería Civil del Instituto de Tecnología de Georgia en Savannah (EEUU) aplicado a la señalización vertical de redes de carreteras secundarias, explicado en el artículo "Detection of Roadway Sign Condition Changes using Multi-Scale Sign Image Marching (M-SIM)" publicado por la revista PE&RS de abril de 2010. En él se analizan fallos comunes en la señalización vertical; como son, la desaparición de las propias señales, la incorrecta posición, o la poca visibilidad de las mismas. El protocolo llevado a cabo en susodicho trabajo nos puede dar una idea acerca de cómo actuar en el nuestro.

Este se basa en tres pasos fundamentales: Paso 1. Usando el posicionamiento mediante GPS se obtiene qué señal debe estar en ese punto (base de datos original). Mediante el algoritmo de SIFT (Scale Invariant Feature Transformation), que se encarga de extraer características distintivas o puntos característicos de las imágenes en escala de grises para buscar diferencias/coincidencias entre las imágenes iniciales y las finales y, apoyado en el algoritmo RANSAC (RANdom SAmple Consensus) –introducido por Martin L. Fishler y Robert C. Bolles en 1981- para la eliminación de valores atípicos (outliers), que es el conjunto de datos numéricamente distante del resto, para evitar la distorsión de las conclusiones del modelo establecido. Se extraen de las imágenes rasgos/características. Paso 2. En el que se analizan las características extraídas de la base de datos original. Paso 3. En el que se analiza y clasifica el tipo de defecto que se ha detectado por comparación entre las características obtenidas y las de referencia. En el mencionado artículo se extraen los siguientes resultados como fruto de la comparación de imágenes entre 2003 y 2005:

	Realidad	Encontradas	Perdidas	Tasa De Detección
Señales Perdidas	6	6	0	100.0 %
Señales Inclinadas	11	8	3	72.7 %
Señales Ocultas	18	12	6	66.7 %
Total Señales	35	26	9	74.3 %

Tabla 1: Índice de detección de fallos del proyecto publicado en la revista PE&RS

Echando un ojo a esta tabla, y haciendo una analogía con nuestro trabajo, podemos considerar que la identificación de señales ocultas es lo que más se acerca a lo que tenemos que hacer. Como vemos, esta clasificación es la que arroja peores resultados en este trabajo en concreto, aunque no tiene por qué ser así en el nuestro, ya que abordaremos el problema de manera distinta.

3.2 Metodología

3.2.1 Toma de imágenes

En el trabajo publicado en la revista PE&RS, citado anteriormente, contaban con imágenes de referencia pertenecientes a señales en perfecto estado que luego comparaban con imágenes posteriores correspondientes a las mismas señales para realizar la clasificación. Esto supone una ventaja en el sentido de que se cuenta con una imagen de referencia para cada una de las imágenes a clasificar, que nos facilita la clasificación ya que lo único que hay que hacer es analizar qué diferencias se producen entre una imagen y otra. Por el contrario, contar con estas imágenes de referencia supone el trabajo extra de tomar todas esas fotografías y almacenarlas.

En nuestro caso no contamos con imágenes de referencia, al menos en este sentido, ya que para ello tendríamos que haber tomado las fotografías en el momento en el que las señales fueron colocadas en su sitio. Por lo tanto, en principio, solo contaremos con las imágenes a clasificar.

Pensando en un proyecto futuro, las imágenes podrían ser tomadas por un dron que sobrevolara todo el carril bici y fuese disparando fotografías en aquellos puntos localizados mediante GPS dónde hubiera señales pintadas. Estas imágenes también podrían ser tomadas por el servicio de limpieza, aprovechando el turno de trabajo, por ejemplo, haciendo uso de una cámara colocada sobre el cubo de basura que transporta el operario.

En cualquiera de estos casos, el tratamiento posterior sería prácticamente el mismo, salvando algunas diferencias, como podría ser el cambio de perspectiva que supondría tomar las fotografías usando un dron.

Las imágenes usadas en este trabajo han sido tomadas a pie, usando un Smartphone, intentando hacer la toma lo más centrada posible con respecto al carril bici, con un ángulo similar entre ellas, y procurando que la señal ocupe un espacio de similares proporciones en cada una de las imágenes. Para ello se han tomado las fotografías a unos 5 metros de distancia entre nosotros y la señal. Esto es importante ya que nos servirá de vital ayuda a la hora de poder clasificar las señales correctamente. Otro punto importante, es tomar todas las fotos bajo buenas condiciones de luminosidad, ya que un exceso de sombras y/o luz puede impedirnos ver bien el estado delas señales.

El Smartphone utilizado para realizar la toma de fotografías del presente proyecto ha sido el iPhone 5, que cuenta con una cámara trasera de 8 megapíxeles (3264x2448).

Por otro lado, debemos considerar, no solo el almacenamiento, sino un ordenamiento de las imágenes que nos permita conocer con exactitud dónde fue tomada la imagen para poder actuar sobre ella en el caso de que hubiese que hacerlo. Esta tarea no es competencia directa de este trabajo, y es por eso que solo tocaremos el tema superficialmente.

En nuestro caso en concreto, una posible solución a la localización de las imágenes sería hacer uso de la geolocalización de fotos que permite iPhone 5. Para ello, antes tenemos que activar esta opción tocando los ajustes de privacidad del dispositivo. Una vez hecho esto, cada foto que tomemos incluirá la localización GPS en los metadatos de la imagen.

Posteriormente, usando MATLAB, podremos acceder a estos metadatos haciendo uso de la función *imfimfo,* cuya ejecución nos da como resultado una estructura de elementos, uno de los cuales es otra estructura llamada GPSInfo. En su interior, se encuentran los campos GPSLatitude y GPSLongitude entre otros.

3.2.2 Software utilizado para el procesamiento de las imágenes

Existen diferentes alternativas para llevar a cabo procesamiento de imágenes usando una computadora, pero como ya sabemos, nosotros usaremos Matlab. Este software, uno de los más usados en ingeniería, es una potente herramienta matemática, que además nos ofrece un entorno de desarrollo integrado (IDE) bastante amable.

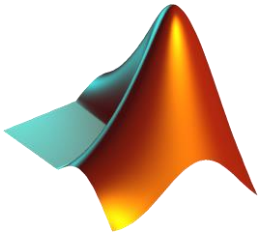


Figura 7: Logo Matlab

Matlab nos ofrece numerosos productos de distintas categorías, entre ellas, la que nos interesa es la que se titula Procesamiento de Imágenes y Visión Artificial. Esta categoría está compuesta a su vez por una serie de tool boxes, de los cuales destacamos dos de ellos, que serán los que usaremos:

- Image Processing Toolbox: "*proporciona un conjunto completo de algoritmos estándar de referencia y apps de flujo de trabajo para el procesamiento, el análisis y la visualización de imágenes, así como para el desarrollo de algoritmos. Puede llevar a cabo segmentación de imágenes, mejora de imágenes, reducción de ruido, transformaciones geométricas, registro de imágenes y procesamiento de imágenes 3D…"*
- Computer Vision System Toolbox: "*proporciona algoritmos, funciones y aplicaciones para diseñar y simular sistemas de procesamiento de video y visión por computadora. Puede realizar la detección, extracción y coincidencia de funciones, así como la detección y el seguimiento de objetos…"*

4 PROCESAMIENTO DE LAS IMÁGENES DEL CARRIL BICI

En este apartado realizaremos el desarrollo del tratamiento llevado a cabo con las imágenes. Primero hablaremos acerca de cómo tomar esas imágenes de la base de datos, para luego abordar las diferentes soluciones que hemos tratado de encontrar al problema de la detección automática de defectos en las señales contenidas en estas imágenes.

4.1 Clasificación previa al procesado

Una vez que tenemos las imágenes del carril bici guardadas en nuestra base de datos, el siguiente paso es procesarlas para poder llegar a la conclusión que estamos buscando, pero, realmente lo que tenemos que analizar son las señales horizontales contenidas en estas imágenes, que ocupan una porción muy pequeña de la imagen completa. Por otra parte, estas señales difieren unas de otras tanto en su forma, como en su tamaño, y como veremos posteriormente, el procesado llevado a cabo va a depender de qué señal estemos tratando.

Por las razones comentadas anteriormente, nos vemos obligados a realizar una clasificación previa de las imágenes en función de las señales contenidas en estas, por lo tanto, tendremos una base de datos por cada una de las distintas señales a tratar. Por simplicidad, nosotros solo analizaremos un total de tres señales distintas: la señal de separación de sentidos, la señal de advertencia de peligro, y la señal de bicicleta seguida de flecha.

Además de esto, debemos tener en cuenta que algunas de estas señales, sobre todo la señal de la bicicleta seguida de flecha, aparecen dos veces en la misma imagen, una en cada carril, paralelas y en sentido contrario. En estos casos, lo lógico sería pensar que tenemos que analizar dos señales distintas en la misma imagen, y tendremos que hacerlo por separado, ya que una señal se puede encontrar en un estado de degradación diferente de la otra, pero mirando estas imágenes, nos damos cuenta de en la gran mayoría de los casos, las dos señales se encuentran en el mismo estado de degradación, por lo tanto, trataremos a estas dos señales como si fueran una sola.

Figura 8: Degradación similar en ambos carriles

4.2 Eliminación automática de regiones no interesantes de la imagen

Todas y cada una de las imágenes tomadas procedentes de carriles bici, comparten una estructura muy similar entre ellas. Esta estructura consta del carril bici propiamente dicho, en color verde, y de todo lo que no es carril bici, en el exterior de la calzada. Lo que no forma parte del carril bici, suele ser de un color distinto al verde, a excepción de zonas dónde hay vegetación pegada al carril. En la figura 8 se muestra la región que no nos interesa de la imagen bordeada de color rojo, y la región de interés bordeada de color verde.

Figura 9: Región de interés

Aprovechando el color verde distintivo del carril bici, podemos preprocesar la imagen para intentar eliminar todo aquello que no sea carril bici. Para ello, es evidente que tendremos que trabajar con las tres componentes de color, RGB, que componen la imagen.

Se proponen dos formas distintas:

- En la primera, recorremos la imagen pixel a pixel, empezando por la esquina superior izquierda de la misma. Se ponen a cero aquellos píxeles dónde no predomine el color verde y paramos cuando nos encontremos con un píxel donde predomine el verde, que debería de ser el inicio del carril. Luego hacemos esto mismo, pero empezando desde la esquina superior derecha. Procesemos de esta forma para no distorsionar píxeles contenidos dentro del carril bici.

  Esta forma tiene la ventaja de que no distorsiona la información contenida dentro del carril bici, pero la desventaja de que si en alguna de las columnas de la fila que estamos recorriendo aparece un píxel verdoso, paramos de recorrer la fila, y esto no siempre pasa cuando nos topamos con el carril bici, ya que hay elementos externos al carril, como por ejemplo vegetación, donde predomina el verde. En la siguiente figura vemos un claro ejemplo de ello:

Figura 10: Primera forma de eliminar elementos externos a la región de interés

- La segunda forma es algo más compleja, pero proporciona un resultado más limpio.

  Esta vez, recorremos la imagen completa píxel a píxel y ponemos a cero todos aquellos píxeles que no sean verdosos. El resultado se muestra en la siguiente figura:

Figura 11: Puesta a cero de píxeles no verdosos

Este paso, se da única y exclusivamente para facilitar una detección de bordes, que es lo que vamos a hacer a continuación.

El objetivo es conseguir localizar el borde de la vía ciclista. Para ello, primero pasamos la imagen a escala de grises, luego se obtendrá una imagen de bordes binarizada, mediante la función *edge(),* con la opción *Sobel* y eligiendo adecuadamente el umbral de binarización. Sobre esta imagen binarizada, se aplicará la transformada de Hough para detectar las dos líneas más dominantes presentes en la imagen, eligiendo adecuadamente el rango angular de las líneas buscadas (parámetro theta de la función *hough*), el umbral (parámetro Threshold) y el tamaño del entorno de supresión alrededor de los máximos (parámetro NHoodSize). Estos últimos parámetros pertenecen a la función *houghpeaks*, que luego comentaremos.

La transformada de Hough es una herramienta que permite detectar figuras en una imagen. En nuestro caso, utilizaremos la transformada de Hough para detectar líneas rectas. Estas rectas, se representan mediante su parametrización en coordenadas polares, donde el parámetro ρ representa la distancia entre el punto (x,y) y el origen de coordenadas y el parámetro θ, que representa el ángulo entre el eje x y la recta perpendicular a la recta original y que pasa por el origen de coordenadas. El rango para el ángulo θ es ±90º.

$$\rho = x * \cos\theta + y * \sin\theta$$

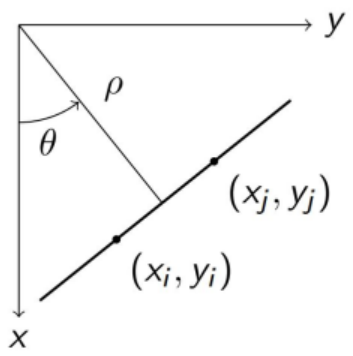


Figura 12: Coordenadas xy en polares

De este modo, Matlab crea una matriz de votos, donde cada punto de contorno vota a todas las posibles rectas que pasan por él. Una vez finalizado el conteo, seleccionamos los pares (ρ,θ) que más votos hayan recibido.

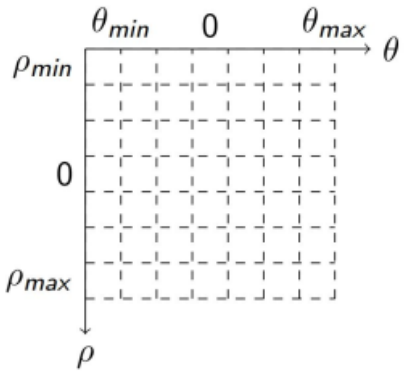


Figura 13: Matriz de votos

Por otro lado, la función *hough* de Matlab solo realiza el computo de votaciones de los pares (ρ,θ). Una vez hecho esto, tendremos que hacer uso de dos funciones más para hallar las rectas. La función *houghpeaks* selecciona picos (máximos relativos) en la matriz de votos, y la función *houghlines* calcula los puntos inicial y final de cada una de estas rectas en la imagen. Cabe destacar que, para nuestro caso en concreto, la función *houghlines* no nos vale, ya que podría trazar solo un tramo de las líneas y no cubrir el carril completo. Por tanto, nos vemos obligados a programar una función que nos calcule los puntos inicial y final de estas rectas dentro de nuestra imagen. El resultado de esta operación sobre una de las fotos se muestra en la figura 14.

Figura 14: Detección automática de las líneas
que delimitan la calzada ciclista

Llegado a este punto, solo nos quedaría poner a cero todos aquellos píxeles situados en el exterior de la calzada según las líneas detectadas.

Figura 15: Puesta a cero de píxeles
fuera del carril

El último paso llevado a cabo es bastante conflictivo, ya que, hacer uso del comando *line* para pintar las líneas que delimitan el carril nos obliga a guardar la foto a partir de una figura de Matlab, y hacer esto nos comprime y modifica la imagen, dejando un margen color blanco en cada uno de los cuatro lados de la imagen.

4.3 Reconocimiento de formas aplicado a señales del carril bici

Como ya vimos en la sección 3.2.2, Matlab dispone de una herramienta llamada Computer Vision System Toolbox. Esta herramienta nos permite, entre otras cosas, detectar objetos contenidos en una imagen, a partir de sus características, usando funciones especialmente diseñadas para esta labor.

Antes de que existiera Computer Vision System Toolbox, ya se realizaban tareas de visión artificial con Matlab, usando el clásico Image Processing Toolbox. En este caso, la detección de objetos se hacía con técnicas como *template matching* (coincidencia de plantillas), o *segmentation and blob detection* (reconocimiento de regiones).

Con la coincidencia de plantillas usamos una imagen como plantilla que represente el objeto que estamos buscando y buscamos coincidencias con respecto a la imagen dónde queramos realizar la detección de objetos usando correlación cruzada. Este método no es robusto a la rotación, oclusión o cambios de iluminación por lo que no sería para nada viable hacer uso de él en este trabajo. Por otro lado, la técnica de segmentation and blob detection tampoco nos serviría ya que no funciona bien en imágenes complejas como las que estamos usando, en las que la segmentación se antoja difícil.

La visión artificial aplicada a la detección de objetos, comúnmente se conoce como basada en características. Haciendo uso de la herramienta Computer Vision System Toolbox, nombrada anteriormente, aplicaremos la técnica de *feature matching* o coincidencia de características, para detectar señales horizontales en nuestras imágenes. Esta técnica, aunque parecida, es mucho más robusta que la coincidencia de plantillas, porque, aunque la señal a detectar no se encuentre en perfecto estado, podemos detectar suficientes características como para encontrar la señal en la imagen a procesar.

Para aplicar esta técnica, usaremos una función de Computer Vision System Toolbox llamada *detectSURFFeatures,* que nos servirá para detectar las características SURF de la imagen. SURF son las siglas de Speeded-UP Robust Features**,** un algoritmo de visión artificial.

Figura 16: Características SURF en imagen a procesar

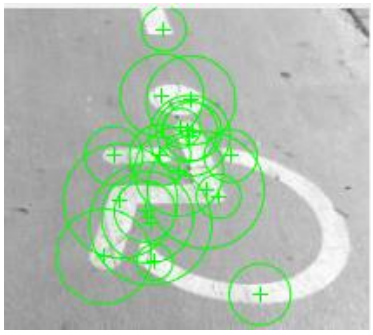


Figura 17: Características SURF del objeto a detectar

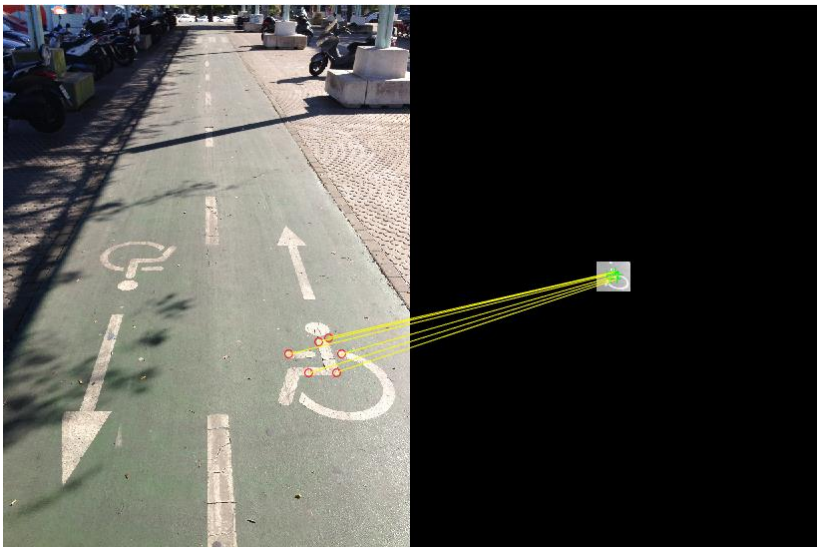


Figura 18: Coincidencia de características SURF entre imágenes

La coincidencia de características, aunque no es una buena forma de evaluar el estado de la señal, podría servirnos para realizar la clasificación previa al procesado, ya comentada en el punto 4.1. Las señales irían a parar a sus carpetas correspondientes a través de la coincidencia de características, así pues, si la imagen de entrada contiene un número determinado de características SURF coincidente con la imagen patrón de una de las carpetas, la imagen iría a parar a esta.

De esta forma, la clasificación de imágenes previa al procesado de realizaría de forma completamente automática, al menos en aquellos casos en los que la señal no esté muy dañada.

4.4 Clasificación mediante umbralización de la imagen y conteo de píxeles

En este apartado, aprovecharemos el hecho de que las señales estén pintadas de blanco, y a medida que se van degradando, van perdiendo la pintura, tomando un tono más oscuro.

Para ello, partiremos de la imagen de carril bici en escala de grises, haciendo uso de la función *rgb2gray*.

Figura 19: Imagen original

Figura 20: Imagen tras *rgb2gray*

Una vez hecho esto, cada píxel contenido en la imagen tiene un valor que va desde 0 a 255. Esto es lo mismo que decir que tenemos 256 posibles tonos de gris contenidos en la imagen, dónde un píxel cuyo valor sea 0 será de color negro, y un píxel de valor 255 será de color blanco.

El siguiente paso será binarizar la imagen. Una imagen binarizada no es más que una imagen cuyos píxeles solo pueden tomar dos posibles valores, en nuestro caso, 0 o 255. Para binarizar la imagen, tenemos antes que establecer un umbral de binarización. Esto es, un valor intermedio, que se encuentre entre 0 y 255, por encima del cual todos los valores serán puestos a 255, y por debajo del cual serán puestos a 0.

Figura 21: Concepto de umbral

Utilizando la imagen de la figura 19, y estableciendo el umbral en 180, el resultado es el siguiente:

Figura 22: Imagen binaria resultado de umbralización

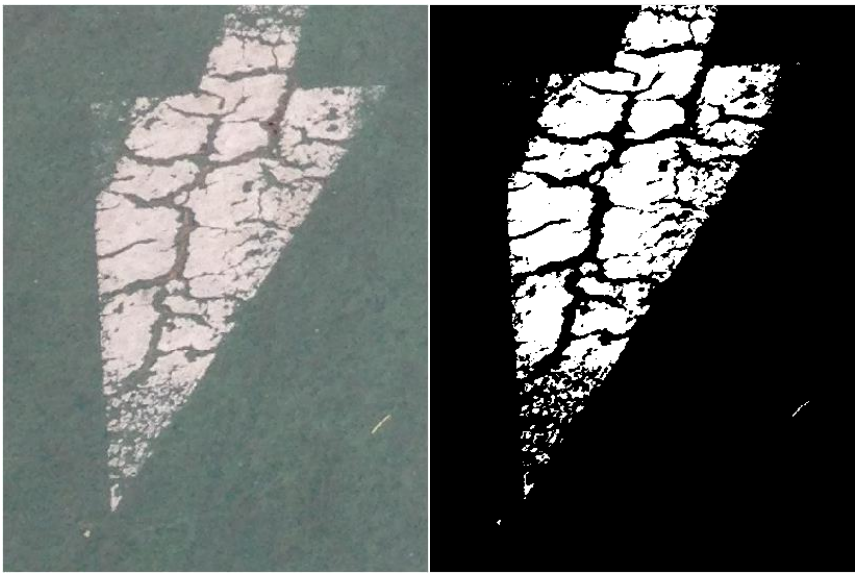


Figura 23: Comparación al detalle entre imagen original y binarizada

Como vemos, la imagen binarizada presenta unos en aquellos puntos dónde la señal horizontal aún conserva el blanco de la pintura, mientras que muestra ceros en zonas deterioradas dónde ya no hay pintura.

Tras este proceso, la idea es contar el número de píxeles blancos que contiene la señal y comparar con el número de píxeles blancos que contiene la misma señal en el caso ideal de que estuviese recién pintada.

Para ello, en vez de buscar por todo el carril bici alguna señal que se encontrase en perfecto estado, esta se ha recreado manualmente utilizando un programa de edición. El resultado se muestra a continuación.

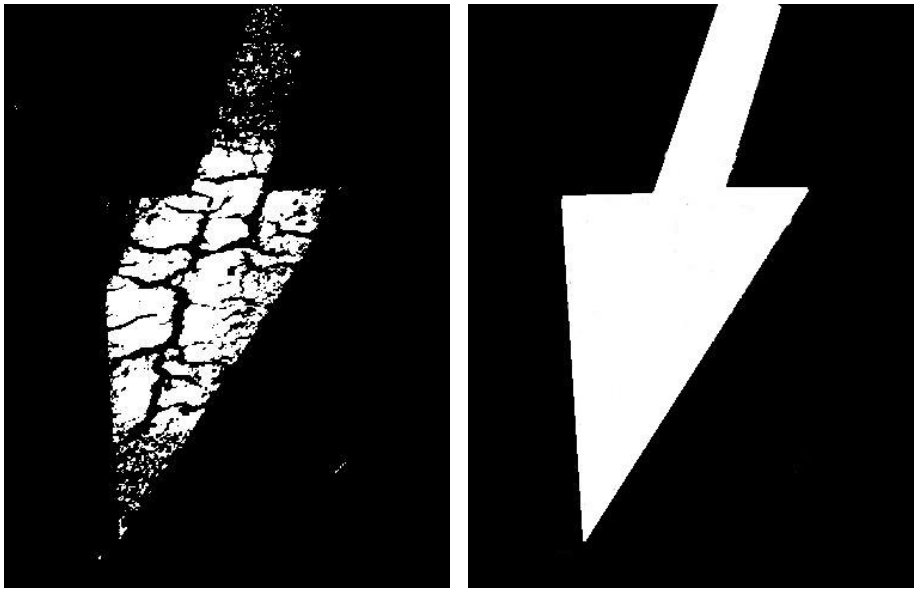


Figura 24: Imagen real e imagen ideal de punta de flecha

Y el resultado obtenido tras comparar la imagen real con la imagen ideal se obtiene a través de la siguiente formula:

$$Desgaste\ (\%) = 100 - 100 * \left(\frac{i}{j}\right) = 48{,}33$$

Donde:

$$i = \sum píxeles\ blancos\ imagen\ real$$

$$j = \sum píxeles\ blancos\ imagen\ ideal$$

Según este resultado, el trozo de señal procesado anteriormente presenta un desgaste del 48,33%.

A partir de aquí, si quisiéramos catalogar la señal como defectuosa o no defectuosa, bastaría con establecer un nivel de desgaste a partir del cual la señal entraría en el grupo de señales defectuosas, y por debajo del cual la señal se catalogaría como no defectuosa. Otra idea es dividir las señales en tres grupos. En un primer grupo estaría las señales en buen estado, seguidas de señales no catalogadas como defectuosas, pero cerca de serlo y, por último, el grupo de las señales catalogadas como defectuosas.

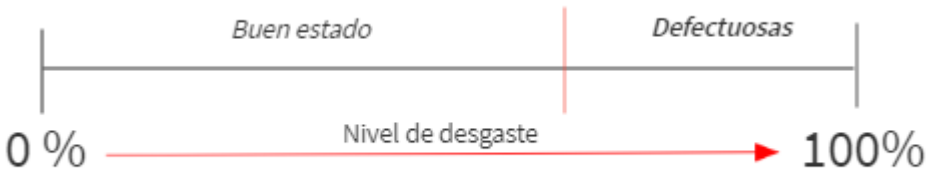


Figura 25: Clasificación en dos grupos

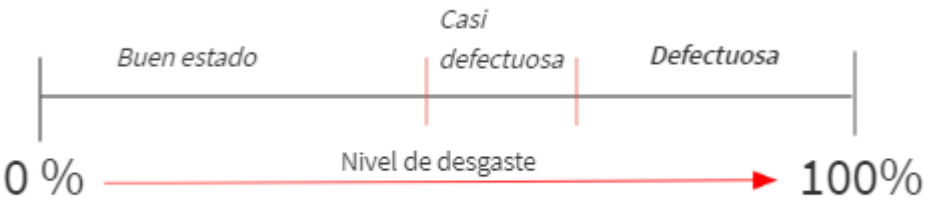


Figura 26: Clasificación en tres grupos

Lo que hemos hecho hasta ahora es procesar solo un trozo de la señal, el correspondiente a la punta de flecha. Ahora vamos a tratar de hacer lo mismo, pero haciendo uso de la imagen completa, mostrada anteriormente en la figura 22.

En este caso el procesado no es tan sencillo, ya que la imagen contiene zonas blancas no pertenecientes a señal. Intentamos primero deshacernos de ellas empleando métodos vistos en la sección 4.2. A continuación se muestra el resultado de aplicar el primer método explicado en dicha sección, junto a la imagen binarizada normal, para observar la mejora que proporciona este método:

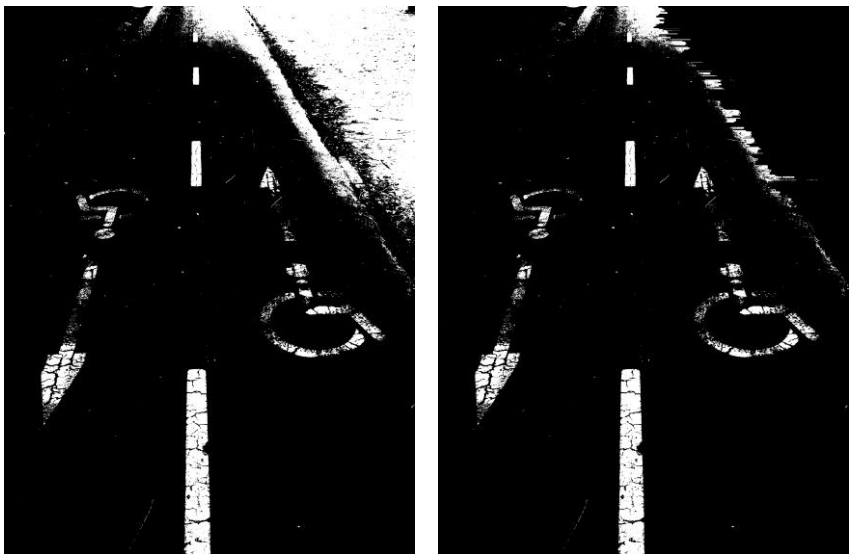


Figura 27: Imagen binarizada junto a imagen binarizada tras eliminación de zonas no pertenecientes al carril

Como vemos, la aplicación de este método no consigue eliminar todos los puntos blancos no pertenecientes a señal. Esto supone un problema, ya que, a la hora de realizar el conteo de pixeles blancos, contaremos más pixeles blancos de la cuenta, y esto nos dará como resultado un desgaste de la señal menor al real.

Una posible solución podría ser suponer que la señal siempre se encuentra centrada en la imagen, y que, gracias a ello, podemos borrar (pintar de negro o poner a cero) la zona superior de la imagen dónde no hay señal.

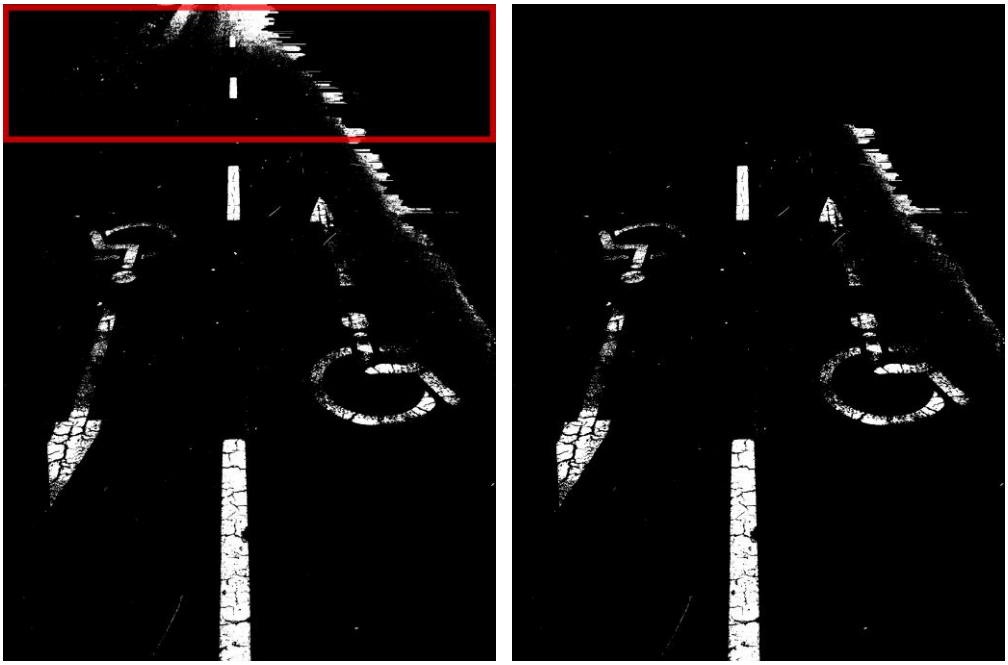


Figura 28: Puesta a cero de los píxeles del margen superior

5 LOCALIZACIÓN DE ZONAS AFECTADAS MEDIANTE BIG DATA

5.1 Recogida de datos referentes al carril bici a través de una supuesta aplicación móvil

La idea es poder contar con algún sistema que permita a los ciudadanos alertar o notificar al Ayuntamiento de Sevilla de cualquier tipo de incidencia que este encuentre oportuna. Para nuestro caso en particular, el sistema nos permitiría notificar la degradación, desgaste o falta de señales de carril bici.

El sistema anteriormente descrito se podría llevar a cabo a través de una aplicación móvil. La aplicación sería sencilla, contaría con una serie de campos a rellenar, los cuales se proponen a continuación:

- Tipo de incidencia
    - Desgaste pintura carril bici
    - Anomalía en la calzada
    - Peligrosidad
- Distrito
    - Bellavista – La Palmera
    - Casco Antiguo
    - Cerro – Amate
    - Este – Alcosa- Torreblanca
    - Los Remedios
    - Macarena
    - Nervión
    - Norte
    - San Pablo – Santa Justa
    - Sur
    - Triana
- Calle
    - Ubicación actual (GPS)
    - Introducir calle y número
- Fecha
    - Actual (mm/dd/aaaa)

Los campos anteriores son solo un ejemplo de cómo podría estar diseñada la aplicación. En nuestro caso, supondremos que son estos datos los que se recogen a través de esta.

El campo calle almacenaría el dato introducido en forma de coordenadas, sea cual sea el método utilizado, y el campo fecha se rellenaría de forma automática con la fecha actual en la que se esté realizando la notificación.

La aplicación almacenaría los datos en formato CSV, dónde en la primera fila estaría compuesta por el nombre de los campos separados por punto y coma, y las siguientes líneas estarían compuestas por el valor de los campos, para cada uno de los avisos introducidos en la aplicación.

Estos datos podrían ser publicados en el portal de datos abiertos del Ayuntamiento de Sevilla (http://datosabiertos.sevilla.org/), con una frecuencia de actualización determinada, y podría contener miles de entradas. Nosotros, ya que no contamos con estos datos, crearemos un archivo CSV a mano, el cual nos servirá de ejemplo para poder tratar los siguientes apartados.

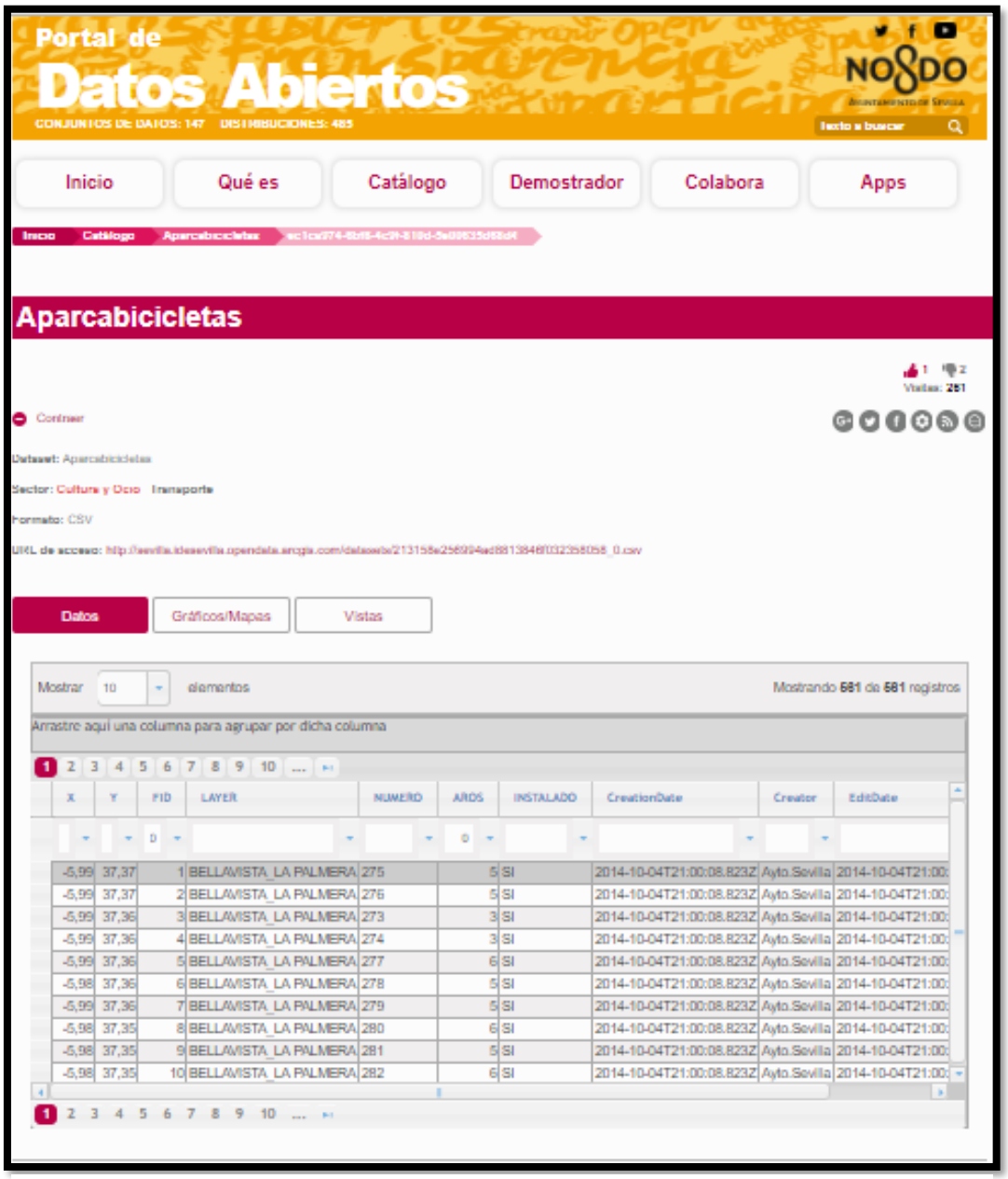


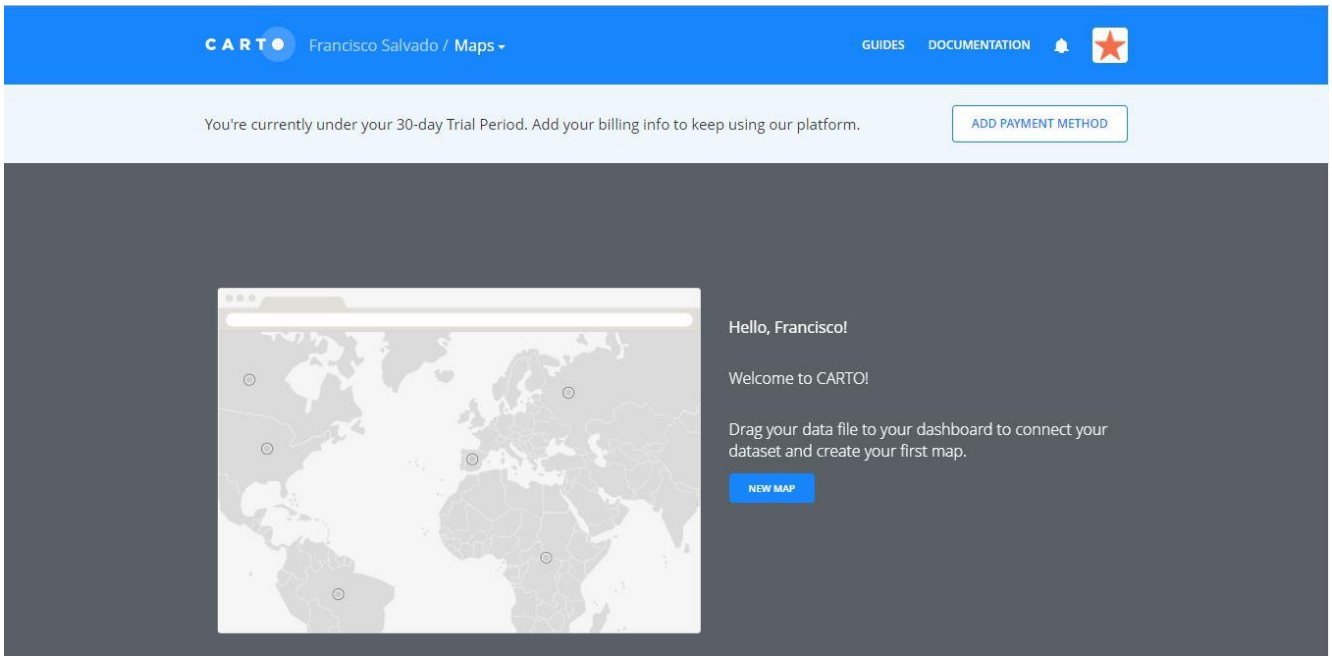
Figura 29: Portal de datos abiertos del Ayuntamiento de Sevilla con datos referentes a la instalación de aparcabicicletas

5.2 Visualización de los datos con Carto

Una vez que tenemos todos los datos referentes a las incidencias producidas en carriles bici contenidos en un archivo, con su correspondiente formato, una buena forma de extraer valor de los mismos es utilizando alguna herramienta de visualización.

En este punto utilizaremos Carto, un software de computación en la nube que proporciona herramientas de información geográfica y mapeo web. Aunque la licencia de uso es de pago, podemos probarlo gratis durante treinta días introduciendo nuestra dirección de correo electrónico.

Una vez dentro, este es el aspecto que presenta la página:



Pulsamos sobre el botón azul “new map” y pasamos a la siguiente ventana:

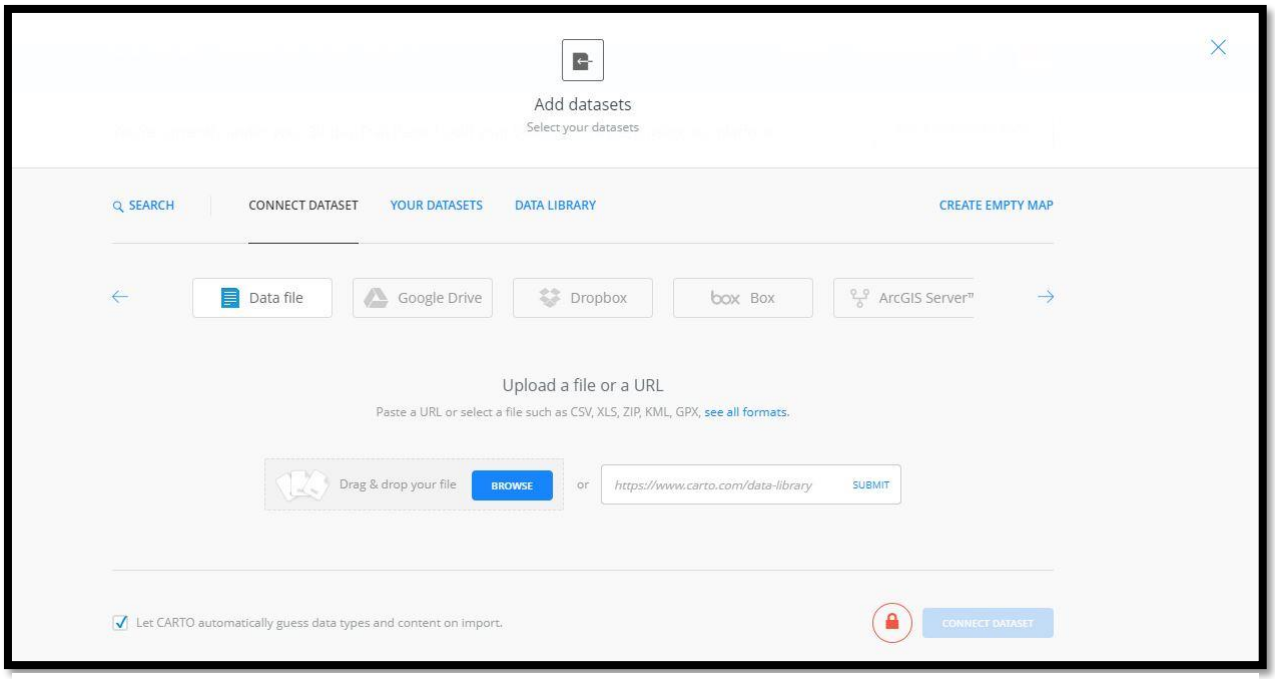


Figura 31: Cargar datasets en Carto

Como vemos, Carto nos da la opción de subir nuestros datasets desde diferentes plataformas y en diferentes formatos. Para el ejemplo mostrado aquí, hemos pulsado sobre el botón "browse" y hemos buscado el archivo Excel creado anteriormente con los datos del carril bici.

	A	B	C	D	E	F
1	Tipo de incidencia	Distrito	Dirección	Latitud	Longitud	Fecha
2	Desgaste pintura carril bici	Macarena	Calle Don Fadrique, 2A	37,403265	-5,989326	04/01/2018
3	Desgaste pintura carril bici	Macarena	Calle Don Fadrique, 43	37,404471	-5,989597	05/01/2018
4	Desgaste pintura carril bici	Macarena	Av. Sánchez Pizjuán, 12	37,407408	-5,989025	06/01/2018
5	Desgaste pintura carril bici	Macarena	Av. La Cruz Roja, 28	37,401045	-5,982124	07/01/2018
6	Desgaste pintura carril bici	Macarena	Av. Concejal Alberto Jiménez-Becerril	37,406377	-5,993775	08/01/2018
7	Desgaste pintura carril bici	Macarena	Calle José Díaz, 9	37,409065	-5,990788	09/01/2018
8	Desgaste pintura carril bici	Macarena	Plaza Dr. Barraquer, 2	37,408299	-5,988805	10/01/2018
9	Desgaste pintura carril bici	Macarena	Calle Dr. Marañón	37,407830	-5,987714	11/01/2018
10	Desgaste pintura carril bici	Macarena	Calle Dr. Marañón	37,407571	-5,986129	12/01/2018
11	Desgaste pintura carril bici	Triana	Calle Américo Vespucio	37,396512	-6,010573	13/01/2018
12	Desgaste pintura carril bici	Triana	Calle Francisco de Montesinos	37,395663	-6,010650	14/01/2018
13	Desgaste pintura carril bici	Triana	Calle de Rodrigo de Triana, 42	37,385066	-6,011396	15/01/2018
14	Desgaste pintura carril bici	Triana	Calle Manuel Arellano, 32	37,385788	-6,011449	16/01/2018
15	Desgaste pintura carril bici	Triana	Ronda de Triana, 14	37,387267	-6,011053	17/01/2018
16	Desgaste pintura carril bici	Triana	Ronda deTriana, 1	37,388183	-6,010798	18/01/2018
17	Desgaste pintura carril bici	Triana	Ronda deTriana, 9	37,389211	-6,010502	19/01/2018
18	Desgaste pintura carril bici	Triana	Av. De la República de Argentina, 14	37,379607	-5,998642	20/01/2018
19	Desgaste pintura carril bici	Triana	Av. De la República de Argentina, 3-2	37,379131	-5,999771	21/01/2018
20	Desgaste pintura carril bici	Triana	Calle san Vicente de Pail, 1	37,385877	-6,007928	22/01/2018
21	Desgaste pintura carril bici	Casco Antiguo	Calle Torneo, 69	37,391369	-6,003336	23/01/2018
22	Desgaste pintura carril bici	Casco Antiguo	Calle Torneo	37,392206	-6,003587	24/01/2018
23	Desgaste pintura carril bici	Casco Antiguo	Pasarela de la Cartuja	37,395495	-6,003231	25/01/2018
24	Desgaste pintura carril bici	Casco Antiguo	Calle Resolana, 37	37,403445	-5,993172	26/01/2018
25	Desgaste pintura carril bici	Casco Antiguo	Calle Muñoz León	37,400605	-5,985099	27/01/2018
26	Desgaste pintura carril bici	S. Pablo - Sta. Justa	Avenida de Kansas City, 32b	37,390619	-5,974962	28/01/2018
27	Desgaste pintura carril bici	S. Pablo - Sta. Justa	Av. José Laguillo	37,390692	-5,975505	29/01/2018
28	Desgaste pintura carril bici	S. Pablo - Sta. Justa	Calle Éfeso, 5	37,394353	-5,962909	30/01/2018
29	Desgaste pintura carril bici	S. Pablo - Sta. Justa	Av. De Kansas city 3A	37,397959	-5,964639	31/01/2018
30	Desgaste pintura carril bici	S. Pablo - Sta. Justa	Calle Samaniego, 48	37,396187	-5,974824	01/02/2018

Figura 32: Dataset de ejemplo creado manualmente en Excel

Una vez seleccionado este archivo tras haber pulsado sobre "browse", pulsamos sobre un botón situado abajo a la derecha que dice "connect dataset". Carto carga el dataset y nos lo muestra tal y como aparece a continuación:

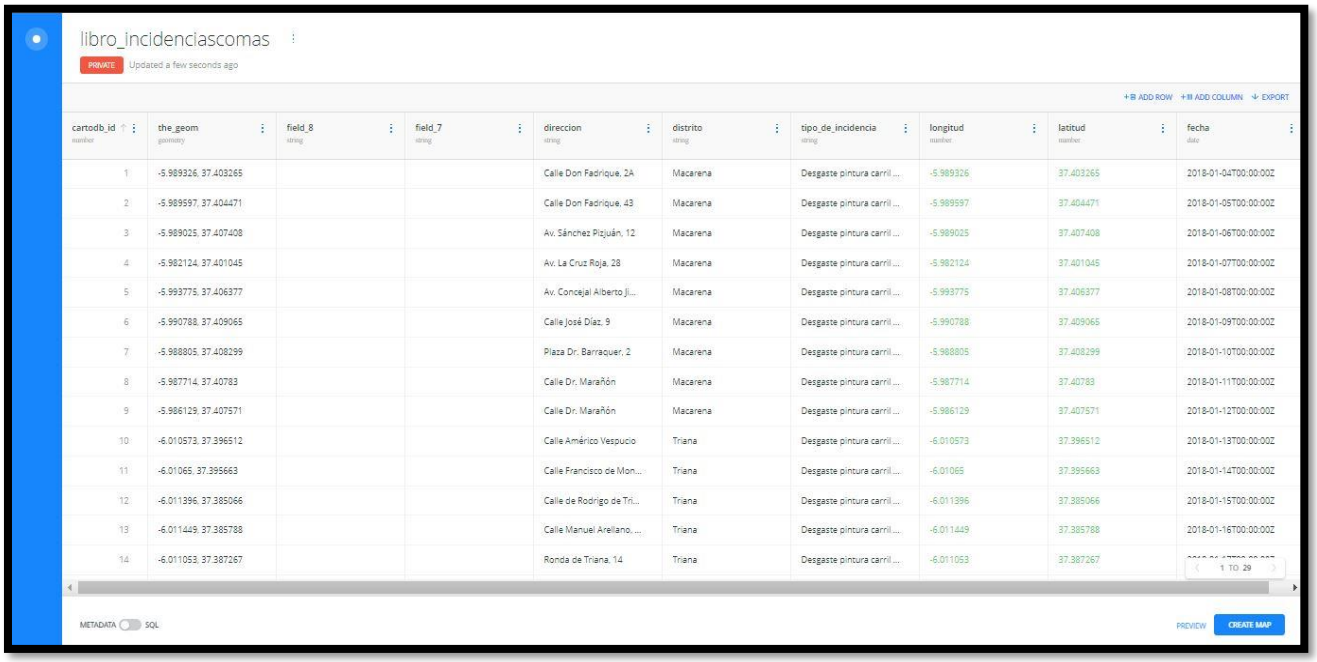


Figura 33: Dataset cargado en Carto

Como vemos, Carto crea campos propios adicionales a los que ya teníamos en nuestro archivo. Por otro lado, Carto es capaz de reconocer los campos latitud y longitud automáticamente, para colocar los puntos sobre el mapa que veremos a continuación, y si no lo hiciese, siempre tendremos la opción de indicar manualmente qué campos debe tomar como referencia geográfica a la hora de representar los puntos.

Lo siguiente que haremos será pulsar sobre "create map". Carto representará los puntos sobre un mapa base que podremos cambiar por otro a elegir entre varias opciones. Una de las cosas que podemos hacer es darle colores diferentes a los puntos en función del distrito en que se encuentren. El resultado de realizar esta operación se muestra en la siguiente figura:

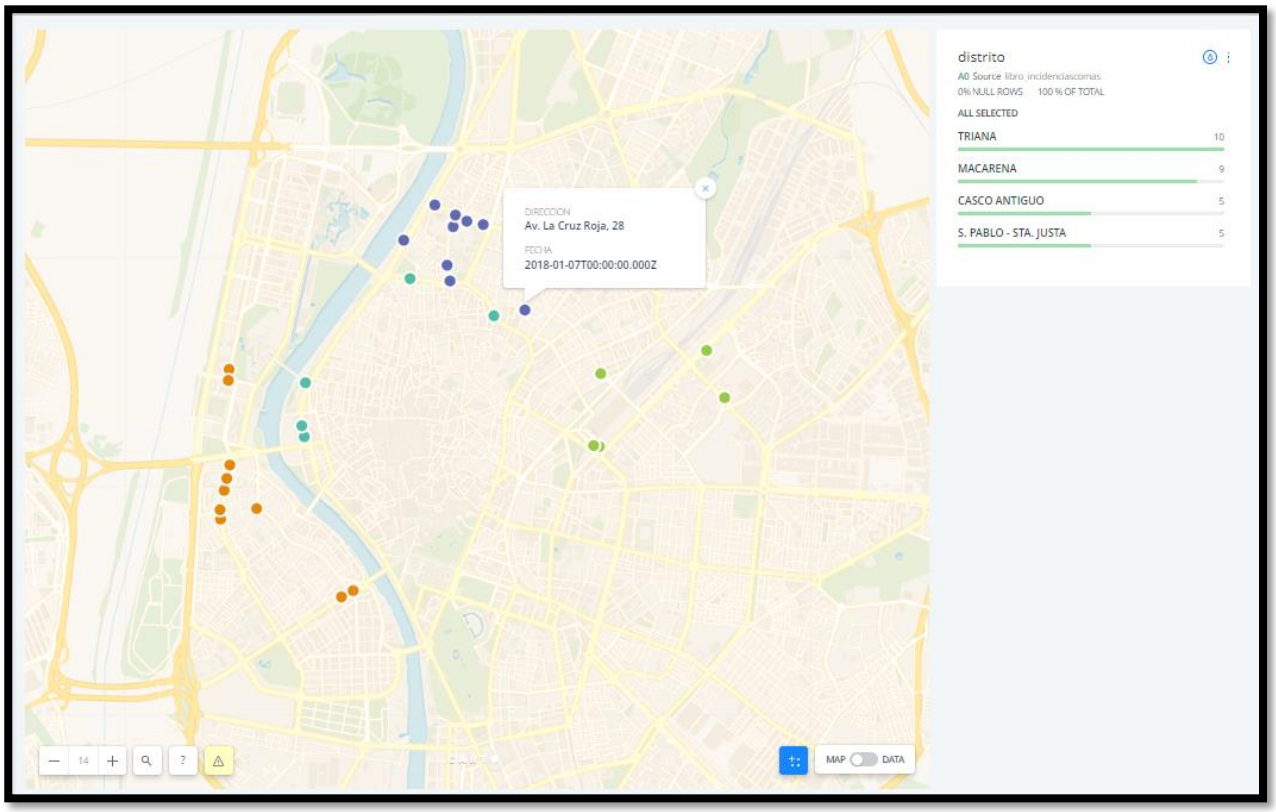


Figura 34: Mapa creado en Carto con incidencias producidas en cada distrito

5.3 Visualización de los datos con Tableau Software

Otra de las herramientas que podemos usar para llevar a cabo la visualización de los datos referentes a las incidencias recogidas por nuestra aplicación, es Tableau Software. Al igual que Carto, Tableau es un software de pago, aunque podremos disfrutar de su versión de prueba gratuitamente durante dos semanas, y a diferencia de este, en vez de trabajar desde su web, trabajaremos desde nuestro ordenador, usando Tableau Desktop.

Una vez instalado y abierto, el aspecto que presenta la página inicial de Tableau es el siguiente:

Figura 35: Tableau Software

En la barra lateral izquierda, dónde pone conectar, vemos las diferentes opciones que nos ofrece Tableau a la hora de trabajar con nuestra fuente de datos, dándonos la opción de trabajar tanto con bases de datos locales como con servidores remotos.

Por otro lado, en la parte central inferior nos aparecen una serie de visualizaciones de muestra. Tableau cuenta con una página llamada Tableau Public, dónde cualquiera que pertenezca a la comunidad puede hacer público su trabajo.

Finalmente, sobre la barra lateral derecha, nos encontramos una serie de recursos de ayuda para aprender a usar Tableau.

Figura 36: Logo Tableau Software

En nuestro caso, procedemos a conectar a un archivo Excel, el cual contiene nuestros datos. Una vez cargado el archivo, nos aparecerá la siguiente pantalla:

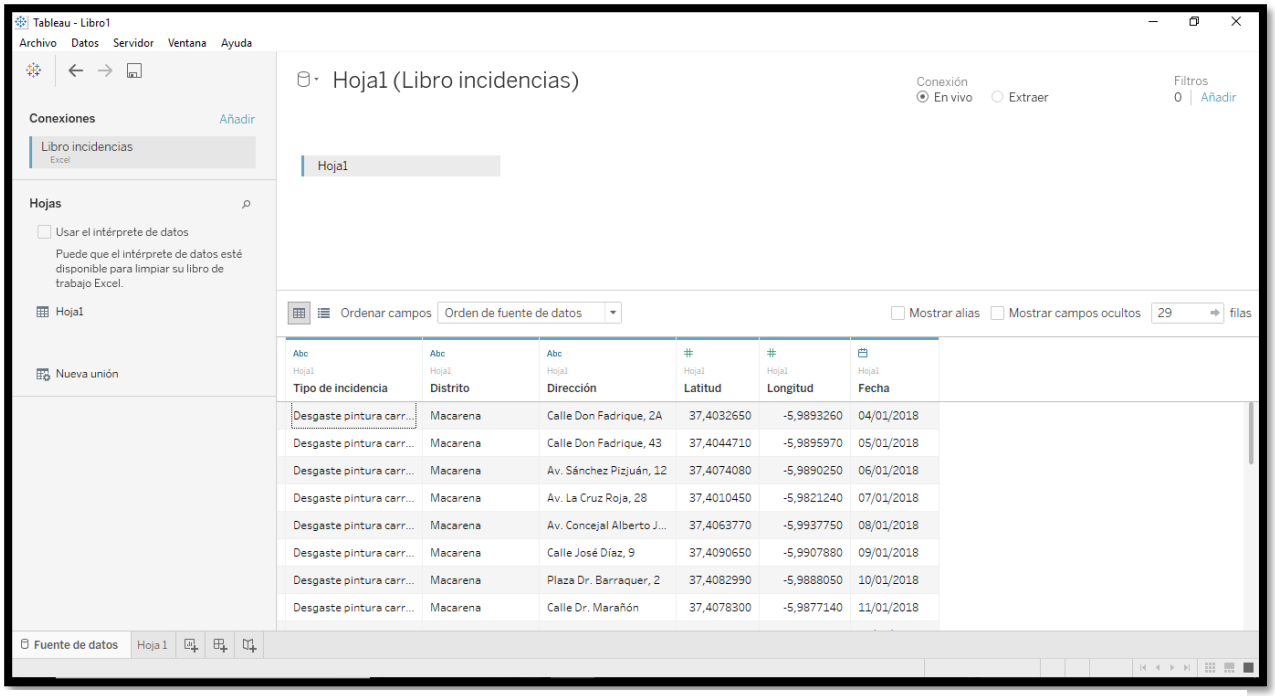


Figura 37: Fuente de datos de Tableau

En la esquina inferior izquierda, podemos identificar que nos encontramos situados sobre la pestaña "Fuente de datos". Justo a su derecha, si pulsamos sobre "Hoja 1", pasaremos a una pantalla dónde poder trabajar visualizaciones de nuestra fuente de datos.

Para que Tableau reconozca nuestras coordenadas, le damos a la longitud y latitud su correspondiente "rol geográfico" tal y como aparece en la siguiente imagen:

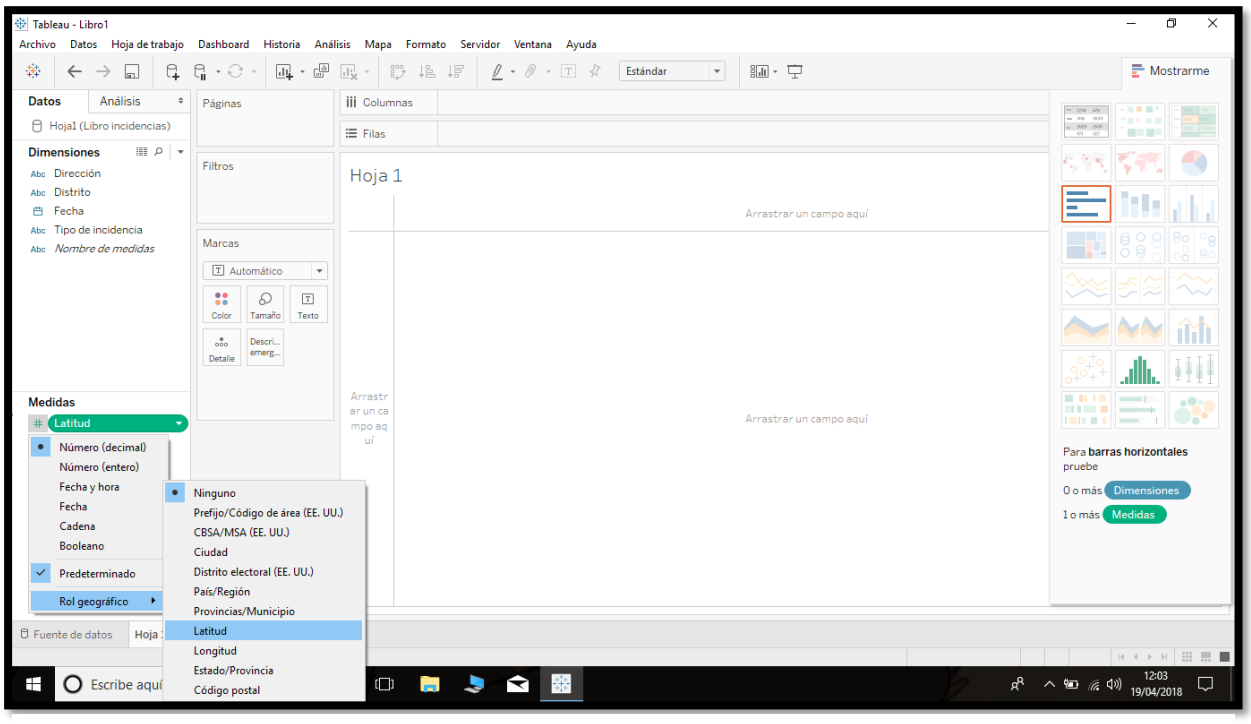


Figura 38: Latitud y longitud como roles geográficos en Tableau

Una vez realizada la operación anterior, nos disponemos a visualizar las incidencias contenidas en nuestra base de datos sobre un mapa, tal y como hicimos con Carto. Para ello, arrastramos latitud a la barra superior "Columnas", y longitud hasta la barra superior "Filas". Al hacer esto, Tableau nos muestra un promedio de todos los puntos, por lo que lo que visualizaremos por ahora será un único punto, el correspondiente al centroide o centro geográfico. Para evitar esto, le decimos a Tableau que tome latitud y longitud en forma de dimensión, en vez de en forma de promedio, tal y como se muestra a continuación:

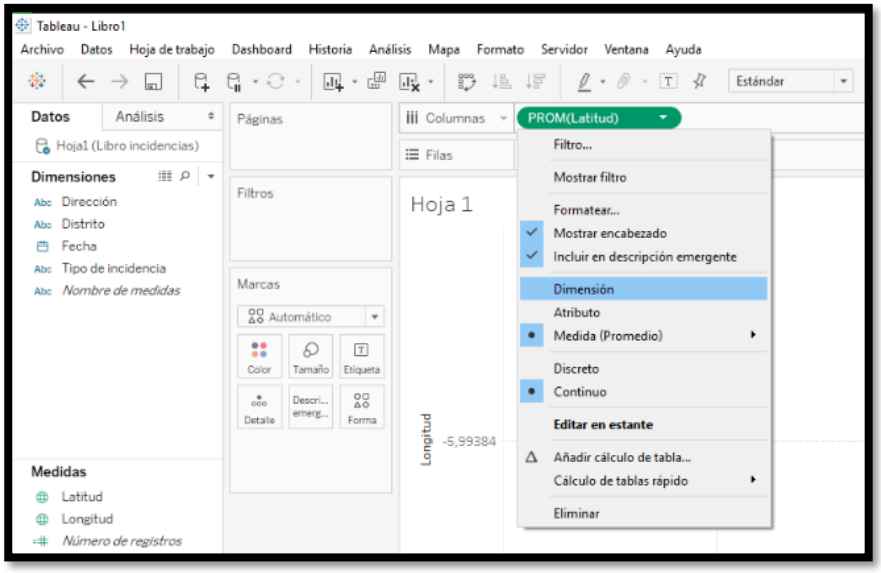


Figura 39: Latitud y longitud como dimensiones en Tableau

Como resultado de la operación anterior, ya podremos visualizar nuestro primer mapa:

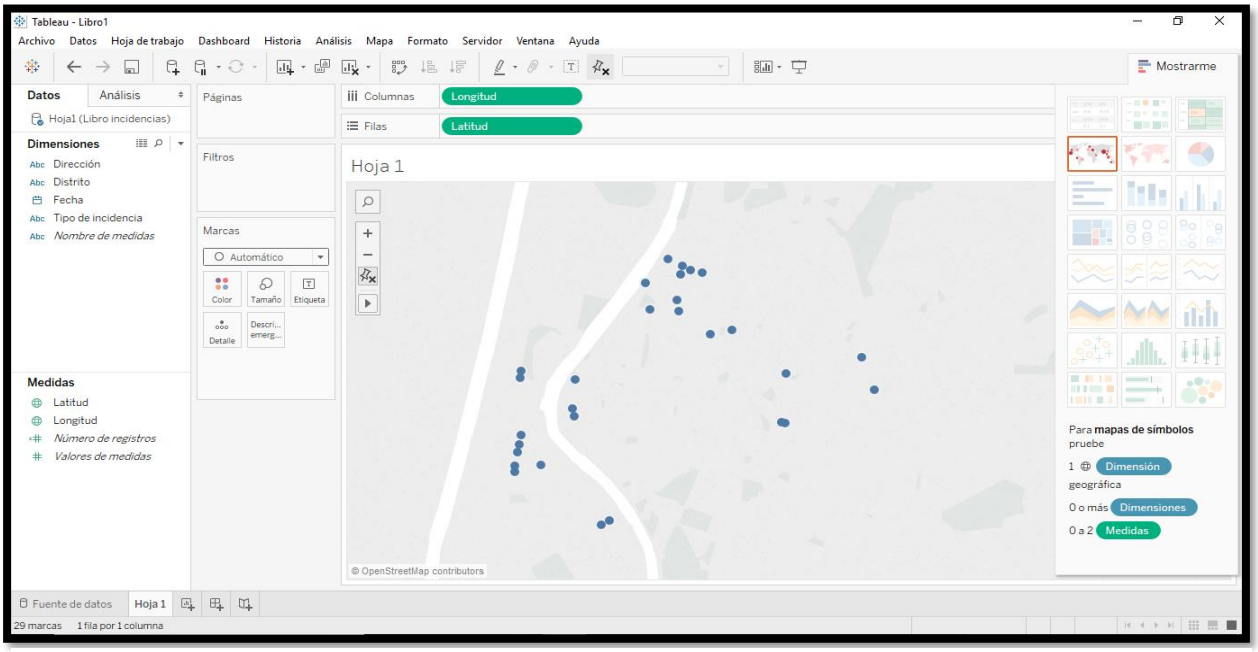


Figura 40: Visualización de incidencias sobre un mapa en Tableau

Para darle un poco de color al mapa, arrastramos "Distrito" desde su posición de origen, en "Dimensiones", hasta la pestaña "Color" situada dentro de "Marcas". Por otro lado, podemos arrastrar "Fecha" y "Dirección" hasta la pestaña "Descripción emergente" para que nos muestre esta información cuando colocamos el ratón sobre uno de los puntos.

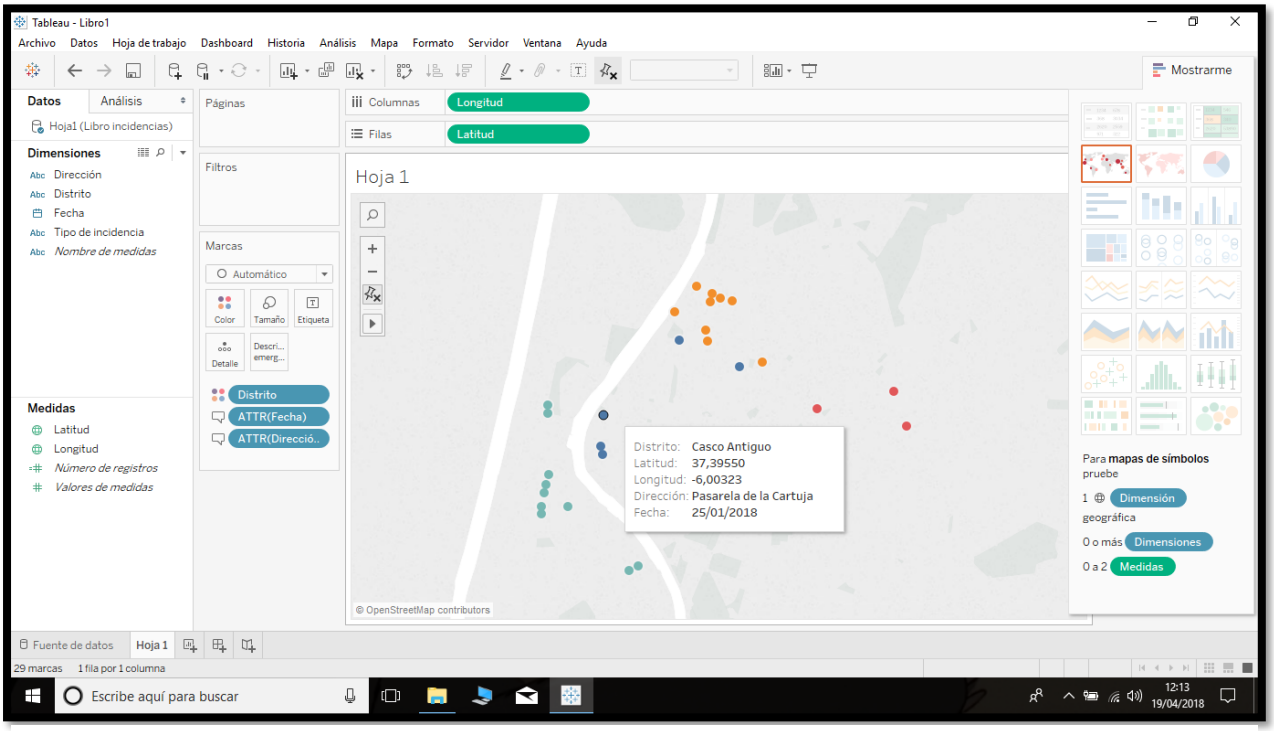


Figura 41: Puntos en el mapa con diferentes colores en función del distrito

Al hacer uso de la latitud y la longitud, Tableau entiende que lo que queremos mostrar es un mapa, como se observa en la barra lateral derecha, la cual nos muestra todos los tipos de gráficos que contiene el programa. Con Tableau podremos realizar multitud de operaciones en función de la información que queramos visualizar. Por ejemplo, podemos mostrar sobre diferentes tipos de gráficos el número de incidencias en función del distrito, como veremos en las siguientes capturas:

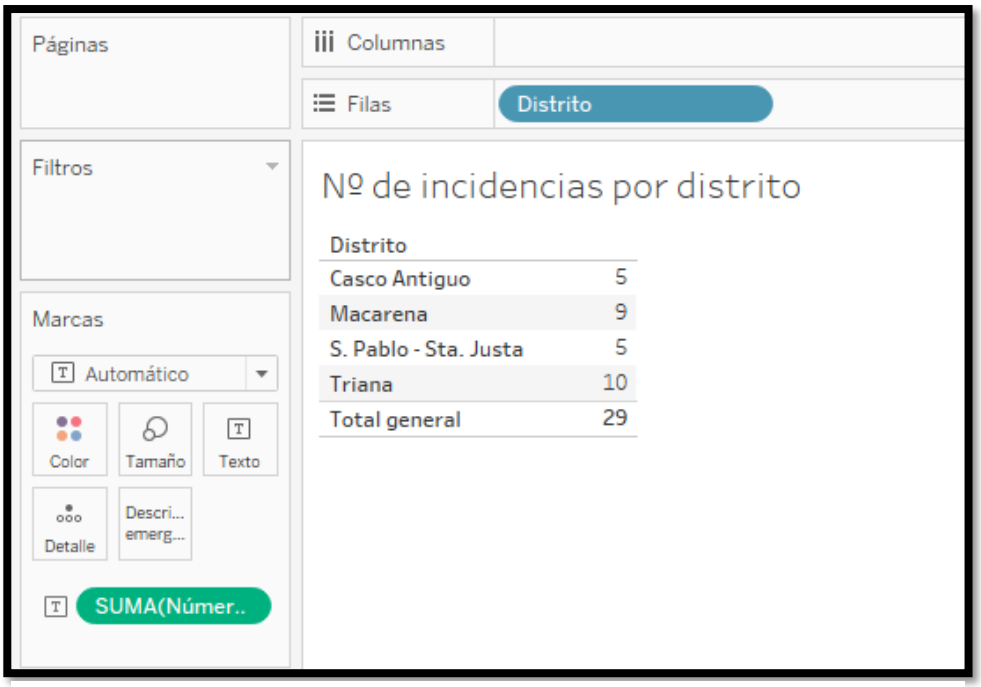


Figura 42: Número de incidencias por distrito en Tableau

Representamos ahora la misma información, pero sobre un gráfico de barras:

Figura 43: Número de incidencias por distrito sobre gráfico de barras de Tableau

5.4 Propuestas de procesamiento mediante Hadoop MapReduce y Apache Spark

Como alternativa a la extracción de resultados empleando softwares de visualización de datos vista en los apartados anteriores, en este punto veremos una manera de cómo procesar los datos de forma distribuida. Para ello, veremos dos alternativas similares, Hadoop MapReduce y Spark. Estas tecnologías, basada en clúster, harán posible que nuestra aplicación pueda trabajar con multitud de nodos y una cantidad ingente de datos.

5.4.1 Proyecto Apache Hadoop

Apache Hadoop es un software multiplataforma, escrito en Java, y de código abierto, lanzado en diciembre de 2011 por los desarrolladores de Apache Software Fundation. Diseñado para resolver problemas que involucran grandes cantidades de datos y requieren gran capacidad de computación, permitiendo el procesamiento distribuido de estos.

El entorno de trabajo de Hadoop está constituido por los siguientes módulos:

- **Hadoop common**: módulo compuesto por librerías y utilidades comunes que dan soporte a los otros módulos.
- **Hadoop Distributed Filed System (HDFS)**: es el sistema de archivos distribuidos creado para Hadoop. Proporciona un acceso de alto rendimiento a los datos.
- **Hadoop YARN**: framework dedicado a la gestión del clúster. Administra y gestiona los recursos del ecosistema.
- **Hadoop MapReduce**: una implementación del modelo de programación MapReduce, un modelo de programación que da soporte a la computación paralela sobre grandes colecciones de datos en un clúster. Implementa una arquitectura maestro/esclavo.

Figura 44: Módulos básicos del ecosistema Hadoop

5.4.2 Apache Spark

Al igual que Hadoop MapReduce, Apache Spark es un framework de computación en clúster, open source y de propósito general. Spark integra, pero no depende de Hadoop, ya que posee diferentes alternativas compatibles tanto para la gestión del clúster como para el almacenamiento de datos distribuido. Está escrito en Scala, aunque también proporciona APIs en Python, R y Java.

Se podría decir que Apache Spark es la evolución de Hadoop MapReduce, ya que aparte de ser mucho más veloz, ha ganado mucha fama gracias a la facilidad de uso que proporcionan sus APIs. Un punto a favor de Hadoop MapReduce es que requiere menor cantidad de memoria RAM que Spark, y esto reduce los costos asociados a la utilización del software.

Figura 45: Logo Apache Spark

Un punto a destacar de Spark es el conjunto de herramientas de alto nivel que ofrece, ya que algunas de ellas, como son Spark SQL y MLib nos podrían ser de gran utilidad en nuestro proyecto. MLib es una librería de machine learning, que nos proporciona utilidades adecuadas para tareas como clasificación y agrupación en clústeres. Una buena idea sería intentar que nuestro sistema aprendiera a clasificar las señales de manera automática aplicando los algoritmos de machine learning que nos ofrece esta herramienta.

6 CONCLUSIONES

Cada vez son más los metros de carril bici construido y, por tanto, cada vez son más las labores de mantenimiento que habrá que hacer para mantener la vía ciclista en buen estado y así poder conseguir mantener la seguridad de sus usuarios. Para agilizar dichas labores, se han planteado dos soluciones diferentes:

- La primera de estas soluciones hace uso de visión artificial. Hemos encontrado un método por el cual poder calcular el desgaste sufrido por las señales horizontales comparando el número de pixeles blancos que contiene la señal contenida en la imagen con el número de píxeles que debería tener esa misma señal en perfecto estado. Para que este método funcione, es imprescindible que las fotografías se tomen siguiendo unas estrictas reglas de encuadre y luminosidad.
- La segunda solución requiere de la participación ciudadana, pues serán los ciudadanos quienes informes del mal estado de estas señales a través de una aplicación móvil. Los datos referentes a las incidencias reportadas por los ciudadanos irán a parar a una base de datos, y se propone que estén disponibles en el portal de datos abiertos del Ayuntamiento en cuestión. Una buena forma de analizar estos datos es usando softwares de visualización como Carto o Tableau. También se propone el uso de computación distribuida a través de Hadoop MapReduce y Apache Spark para el tratamiento de los datos. Esta última solución requiere de una infraestructura quizás demasiado ambiciosa teniendo en cuenta que nos encontramos en la fase inicial del proyecto.

BIBLIOGRAFÍA Y REFERENCIAS

- R. C. Gonzalez, R. E. Woods y S. L. Eddins. (2010). "Digital image proccesing using MATLAB" (2ª ed). Mcgraw Hill.
- Tsai Y., Z. Hu y C. Alberti. "Detection of Roadway Sign Condition Changes using Multi-Scale Sign Image Matching (M-SIM)". Photogrammetric Engineering & Remote Sensing No.4 pp.391-405 Abril 2010
- The Apache Software Fundation. (2014). Welcome to Apache Hadoop. Recuperado de http://hadoop.apache.org/index.pdf
- The Apache Software Fundation. (2018). Apache Spark. Recuperado de https://spark.apache.org/docs/latest/
- Duda, R. O. and P. E. Hart, "Use of the Hough Transformation to Detect Lines and Curves in Pictures, Comm. ACM", Vol. *15*, pp. 11–15 (Enero, 1972)
- Shapiro, Linda and Stockman, George. (2001). "Computer Vision", Prentice-Hall, Inc.
- Herbert Bay, Andreas Ess, Tinne Tuytelaars, Luc Van Gool, "SURF: Speeded Up Robust Features", Computer Vision and Image Understanding (CVIU), Vol. 110, No. 3, pp. 346--359, 2008

- [1]: Garcia, L. *Visión Artificial con MATLAB*. [Figura]. Recuperado de https://es.mathworks.com/videos/computer-vision-with-matlab-for-object-detection-and-tracking-82036.html
- [2]: Consejería de Fomento y Vivienda de la Junta de andalucía. (2013). *Recomendaciones de diseño para las vías ciclistas en Andalucía*. [Figura]. Recuperado de http://www.juntadeandalucia.es/fomentoyvivienda/estaticas/sites/consejeria/areas/transportes_infraestructuras/plan_bici/documentos_plan_bici/recomendaciones_diseno_vias_ciclistas.pdf