



Detección de baterías de coche



Daniel Puente Ramírez

Hardware de Aplicación Específica
Grado en Ingeniería Informática
Universidad de Burgos
Burgos, España

26 de diciembre de 2021

Índice general

Índice general	2
0.1. Introducción	1
0.2. Agradecimientos	1
0.3. Planteamiento del problema	2
0.4. Resolución del problema	3
0.5. Opinión personal y trabajos futuros	9
0.6. Código	12
0.7. Material utilizado	15

0.1. Introducción

Esta práctica se encuentra enmarcada dentro de la asignatura *Hardware de Aplicación Específica* del Grado en Ingeniería Informática, 4º curso, de la Universidad de Burgos.

En ella se plantea el problema de contar baterías de coche, ver Sección 0.3. Para la resolución de la misma se desean poner en práctica, y con ello demostrar lo aprendido a lo largo de la asignatura, la gran mayoría de conceptos y técnicas utilizadas a lo largo de las prácticas.

0.2. Agradecimientos

Agradecer a Clarios Iberia S.L. la colaboración con la práctica, cediendo el material que se va a utilizar de forma desinteresada.

0.3. Planteamiento del problema

El problema ante el que nos encontramos se encuentra dentro de la etapa final de fabricación de baterías para coches. Después de todo el proceso de producción, llenado, reposo, *testing* y etiquetado, las baterías han de ser paletizadas para su transporte. Debido a las características de los equipos de trabajo, no se desea contar con un sistema de Inteligencia Artificial, sino que se desea realizar un procesamiento de imágenes en tiempo real.

La empresa desea contabilizar el número de baterías que han sido finalizadas por turno, para ello desea instalar una serie de cámaras sobre las cintas transportadoras previas al robot de paletizado, de cada línea, ver Figura 0.1.

NOTA: El planteamiento del problema es para despliegue del producto en un sistema de tiempo real. Por motivos obvios y coherentes se va a realizar sobre vídeos *a mano* en 3 líneas diferentes de la planta de fabricación.

Además, cada línea está organizada de manera diferente, por lo tanto, no todas cuentan con el mismo espacio vertical para colocar la cámara, el sistema deberá de ser capaz de detectar las baterías en un rango de entre veinticinco y cincuenta centímetros a la cinta de transporte.



(a) Zona de paletizado



(b) Una sola zona de paletizado



(c) Zona de implantación



(d) Inicio cinta transporte



(e) Entrada al robot



(f) Zona de detección

Figura 0.1: Vista de más a menos de la zona de trabajo

0.4. Resolución del problema

Consideraciones importantes

Antes de analizar la solución propuesta, debemos de tener en cuenta los siguientes apartados, ya que todos ellos influirán en mayor o menor medida en la codificación de la solución.

NOTA. No se encuentran ordenadas de ninguna manera.

1. La distancia de la cámara a la cinta de transporte.
2. La distancia de la cámara a la batería.
3. El ángulo de la cámara con respecto a la cinta. Al encontrarse en una zona altamente robotizada y con muchas vibraciones, la cámara puede «moverse» con el paso del tiempo.
4. La cinta no está siempre en movimiento. Si el robot de paletizado se encuentra realizando la operación de carga, es decir, cogiendo baterías de la cinta, para evitar colisiones con las siguientes baterías, la cinta se detiene. Por lo que una batería puede estar n instantes temporales delante de la cámara.
5. El *frame rate* de la cámara.
6. El color de la batería.

Aproximación a la solución propuesta

Con las consideraciones definidas, pasamos a la parte del análisis de las mismas y posterior planteamiento de la solución. Se van a comentar punto a punto, $[1 - 1, 2 - 2, \dots, n - n]$.

Una de las principales características que se desean es la generalización, tal y como se ha estado realizando a lo largo de las prácticas del curso, es decir, que no sea una solución específica para una línea, vídeo en este caso, en concreto.

1. Conocemos que la distancia se va a situar en un rango de entre veinte y cincuenta centímetros. Además, comentando el problema con el Ingeniero Jefe de Mantenimiento de la planta, comenta que el ancho de la cinta de transporte es de 170 milímetros, luego incluso con la máxima cercanía, tendremos la cinta al completo.
2. Las baterías poseen una altura de entre 127 y 190 milímetros, por lo tanto en ningún caso tendremos problemas de alturas. El Vid00 posee una distancia mayor a la cinta que los Vid01 y Vid02.
3. Para hacer un programa robusto y tolerante a vibraciones y pequeños movimientos de la cámara, el Vid00 está «más o menos» recto, el Vid01 está torcido a la izquierda, y el Vid02 a la derecha.
4. La lógica del producto soporta que una batería se encuentre un tiempo indefinido delante de la cámara. Sólo será contabilizada una vez.

5. Debido a que la cámara graba, independientemente si es para una aplicación de tiempo real o procesado de imágenes, a 60fps 1080x1920 con un codec H.264, el programa configura una variable *jump*, la cual será la encargada de analizar un *frame* cada *jump* de *frames*. Por ejemplo, si *jump* = 45, cada 45 *frames* se analizará un *frame*.
6. A pesar de que el *monoblock*¹ generalmente es negro, marcas como Mercedes lo utilizan blanco, otras en gris; y consecuentemente la tapa que es lo que detecta el producto también cambia de color en función de pedidos, pudiendo ser negra, azul o gris.

La solución propuesta tiene un planteamiento sencillo. Para cada batería que pasa por la cinta, se cuenta y se añade a un contador. El *Stateflow* del código se puede consultar en la Figura 0.8.

Lo primero de todo se calcula la media acumulativa de **todo** el vídeo, con lo que se obtiene una buena imagen de lo que vendría a ser el fondo del que se dispone. Tenemos que destacar que los vídeos han sido grabados «a mano» por lo tanto hay cierto movimiento o temblor de la cámara. Junto con la problemática de que el fondo no está quieto, la cinta de transporte se mueve, sumado a que no es un entorno preparado para la grabación, luego está lleno de luces y sombras, reflejos, etcétera. El *Stateflow* de la función encargada del cálculo del fondo se puede consultar en la Figura 0.7.

$$\begin{aligned} B(t) &= (1 - \alpha) \times B(t - 1) + \alpha \times I(t - 1) \\ F(t) &= |I(t) - B(t)| > \text{umbral} \\ \alpha &= 0,05 \rightarrow \text{Parámetro de aprendizaje} \end{aligned}$$

En una aplicación de tiempo real el fondo sería una imagen conocida, puesto que sería fijo para cada línea, pudiendo ser mucho más precisos los cálculos. En el caso de este proyecto se calcula con la media acumulativa, con un parámetro de aprendizaje de $\alpha = 0,05$. Los fondos obtenidos para cada uno de los tres vídeos se pueden consultar a continuación, en las Figuras 0.2a 0.2b 0.2c.

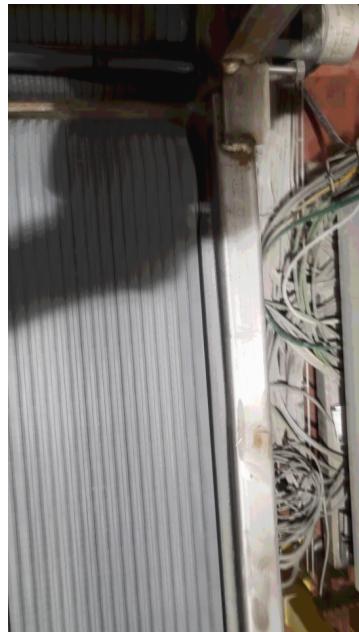
El fondo solo es necesario calcularlo una vez, puesto que queda almacenado para cada vídeo dentro de la carpeta `./Videos/`, para su uso futuro, ya que los tiempos de cómputo de los mismos son de entre uno y dos minutos. Los ficheros se guardan con la extensión `.mat` de forma que únicamente se guardan matrices, sin cabeceras de figuras ni nada.

Una vez que se tiene el fondo, comienza el análisis del contenido del vídeo. Para ello se analizan los *frames* indicados en función del *jump* descrito anteriormente.

¹El *monoblock* es la caja de plástico que forma la batería, en vacío.



(a) Fondo Vid00.



(b) Fondo Vid01.



(c) Fondo Vid02

Cada *frame* sufre un proceso morfológico para poder detectar los objetos que hay en él. Para ello se siguen los siguientes pasos:

1. Convertir el *frame* a escala de grises.
2. Se resta el *frame* en escala de grises con el fondo general previamente calculado y pasado a escala de grises.
3. Como no trabajamos con una distancia a la cinta de transporte fija, se trabaja únicamente con un ancho de [250:750] píxeles, no con el *frame* completo. Esta modificación ayuda a la detección de objetos y eliminación de ruido cuando la cámara se encuentra lejos de la cinta, y cuando estamos cerca no sufre efecto contradictorio alguno.
4. Se convierte en imagen binaria en función de un valor umbral calculado para cada *frame*.
5. Se erosiona el *frame* mediante «discos» de 12 píxeles de tamaño.
6. Se rellenan los «huecos» del *frame*.
7. Se dilata el *frame* mediante «cuadrados» de 80 píxeles de tamaño.
8. Se realiza un filtrado medio del *frame* en dos dimensiones. Cada píxel de salida contiene el valor mediano en una vecindad de 3 por 3 alrededor del píxel correspondiente en la imagen de entrada.
9. Se eliminan todos aquellos componentes conectados (objetos) que tienen menos de 200.000 píxeles, recordamos que trabajamos con una resolución de 1080x1920.

10. Detectamos los objetos restantes en la imagen.
11. Si hay objetos detectados y se encuentran dentro del área de detección y suma, área reservada donde se cuentan las baterías, éste área es la mitad inferior de la imagen, ver Figura 0.3; se realizan cálculos para saber si es la misma batería que en el *frame* anterior o no.
12. Cuando finalizan todos los *frames* se obtiene una salida con el número de baterías pasadas, ver Figura 0.6a.

Se adjuntan dos ficheros, *source.m* y *source_testing.m*, la diferencia sustancial es la visualización en «tiempo real» del procesado. En *source_testing.m* podemos ver la Figura 0.3, en la cual tenemos 7 columnas. Cada una de las columnas hace representa un paso del proceso anterior, de izquierda a derecha:

1. Imagen resultante de la resta del *frame* con el fondo, después de haber sido pasadas a escala de grises y acotadas.
2. Imagen binaria resultante.
3. Imagen resultante al proceso de erosión.
4. Imagen resultante después del llenar los «huecos».
5. Imagen resultante después de dilatar.
6. Imagen resultante después de pasar el filtro de media.
7. Imagen sobre la que buscamos los objetos después de eliminar los objetos pequeños.

Se puede comprobar como el paso 6 es muy similar al 5, según el reflejo de las pegatinas de la propia batería, hay determinados momentos en cuales produce una pequeña diferencia, simplemente está puesto para hacer un pequeño ajuste de píxeles rebeldes, la máscara de 3x3 es muy pequeña como para alterar la imagen significativamente.

Sobre la imagen número 7 podemos ver dos líneas verdes, una en el medio de la imagen y otra en el límite inferior, simbolizan el área de detección. Cuando el centroide del objeto detectado se encuentra dentro de ese área es cuando se cuenta la batería. Si en el *frame* anteriormente analizado ya había una batería el sistema entiende que es la misma, si no la había, la computa como una nueva. Con esto da soporte a que una batería se quede parada en el área de detección, lo cual ocurre en los vídeos de prueba Vid01 y Vid02. Debido a que el avance de la cinta de transporte es mejor a 3km/h, no hay opción a que entre dos *frames* salga y entre una batería, imposibilitando la no detección de una batería. En este caso esta

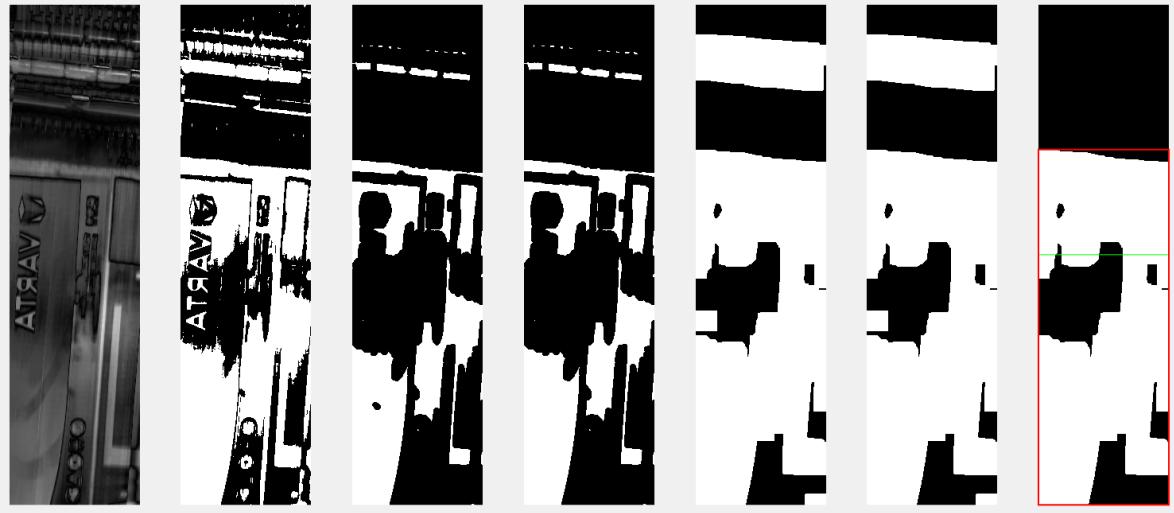


Figura 0.3: Procesado batería con tapa azul.



Figura 0.4: Procesado batería con tapa azul.

figura está actuando sobre el Vid02, a parte del ángulo de la cinta con respecto a la cámara, la tapa de batería es azul y está relativamente cerca de la misma.

En la Figura 0.4 se aprecia como en el momento de la detección las líneas del área de detección cambian de color a rojo. Esta figura es resultante del Vid00, en este caso estamos más alejados de la cinta de transporte y no tenemos «de lleno» la intersección con la máquina anterior (parte de arriba) y una sombra considerable en la parte inferior, después de todo el procesado se es capaz de contabilizar la batería.

Sobre el fichero listo para desplegar en producción (haciendo los ajustes de vídeo para tiempo real), *source.m*, nos proporciona un procesado en tiempo real de todo el *frame* con sus correspondientes objetos detectados, ver Figura 0.5a y 0.5b.

0.4. RESOLUCIÓN DEL PROBLEMA



(a) Detección antes de llegar a la franja.



(b) Detección en la franja.

Salida por pantalla

La salida por pantalla que obtenemos es un *pop-up* con el número de baterías que han sido contabilizadas.



(a) 12 baterías detectadas en el Vid00 (b) 8 baterías detectadas en el Vid01 (c) 13 baterías detectadas en el Vid02

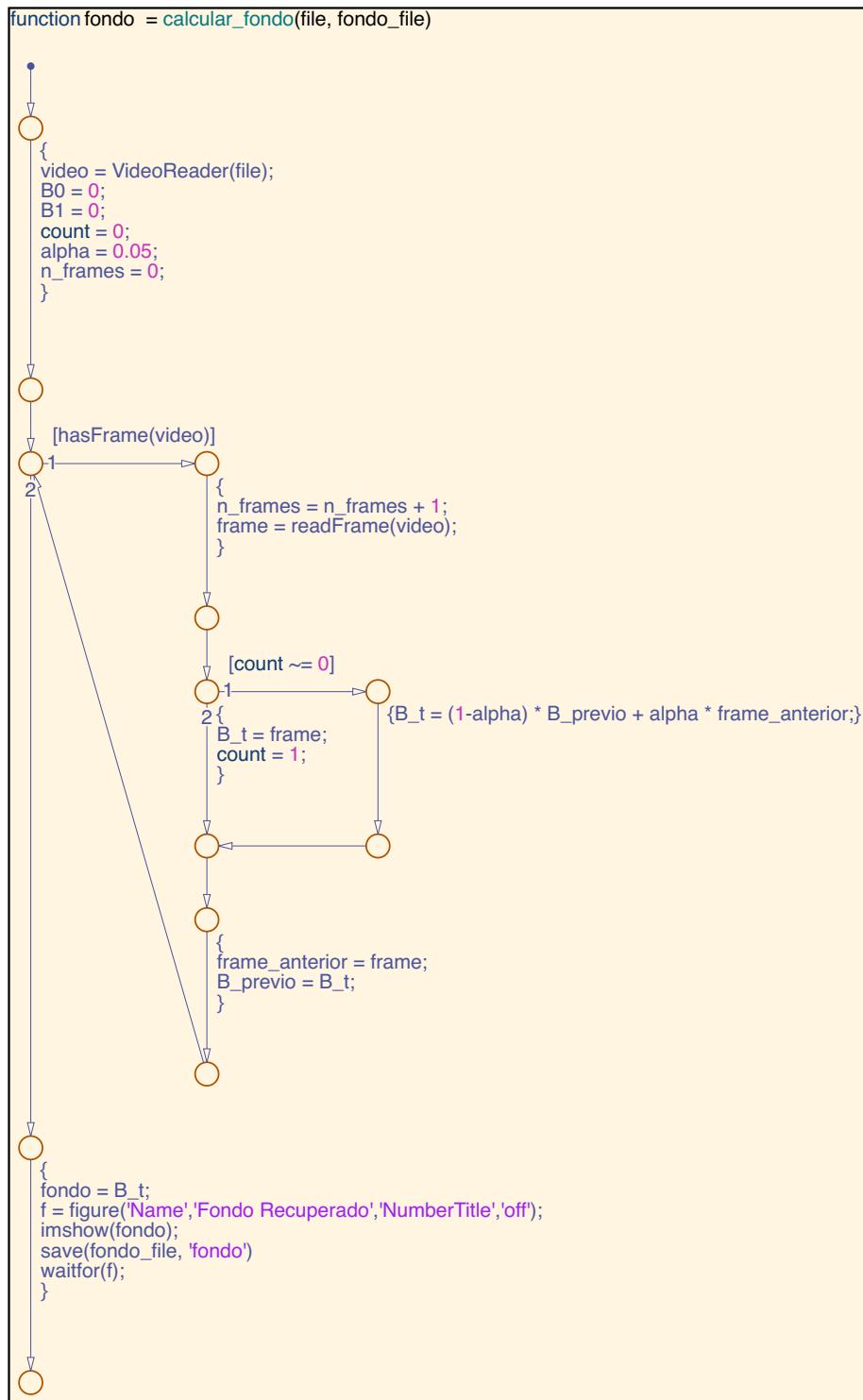
Finalmente podemos comprobar como son correctos los cálculos automáticos que ha hecho el *software*.

0.5. Opinión personal y trabajos futuros

Si bien este software podría ser interesante para la aplicación real, tal y como está planteado. No es necesario, puesto que es el propio robot de paletizado el que se encarga de «contar» cuántas baterías lleva paletizadas y por ende, cuántas han pasado.

Cuando plantee esta práctica no sabía cuántos conocimientos o técnicas vistas a lo largo del curso iban a ser consideradas, a mediados de noviembre ya con el material preparado comencé con el trabajo, antes de conocer que las últimas prácticas de la asignatura, así como el reto iban a tener una influencia tan importante en la aproximación a utilizar.

Si bien no se emplean técnicas como mapas de color, o *chroma-key* o similares, no son técnicas necesarias para este proyecto.

Figura 0.7: *Stateflow* de la función que calcula el fondo de un vídeo dado.

ÍNDICE GENERAL

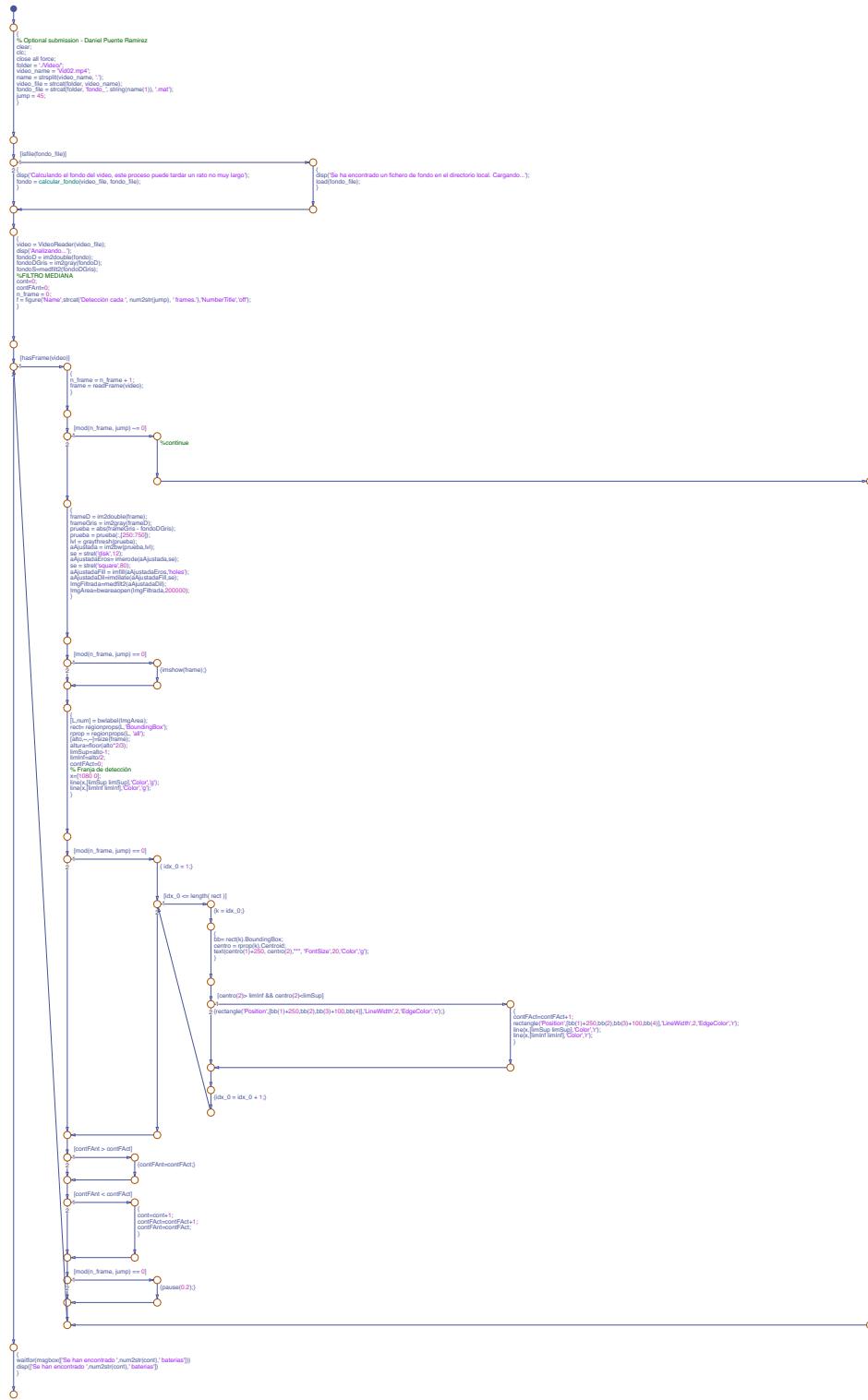


Figura 0.8: *Stateflow* del código completo



0.6. Código

```
1 % Optional submission – Daniel Puente Ramirez
2 clear;clc;close all force;
3 folder = './Video/';
4 video_name = 'Vid02.mp4';
5 name = strsplit(video_name, '.');
6 video_file = strcat(folder, video_name);
7 fondo_file = strcat(folder, 'fondo_', string(name(1)), '.mat');
8
9 jump = 45;
10
11 if isfile(fondo_file)
12     disp('Se ha encontrado un fichero de fondo en el directorio local.')
13     load(fondo_file);
14 else
15     disp('Calculando el fondo del video, este proceso puede tardar un rato no muy largo');
16     fondo = calcular_fondo(video_file, fondo_file);
17 end
18
19
20 video = VideoReader(video_file);
21 disp('Analizando ...');
22 fondoD = im2double(fondo);
23 fondoDGris = im2gray(fondoD);
24 fondoS=medfilt2(fondoDGris); %FILTRO MEDIANA
25 cont=0;
26 contFAnt=0;
27
28
29 n_frame = 0;
30 f = figure('Name', strcat('Deteccion_cada ', num2str(jump), ' frames .'), 'NumberTitle', 'off');
31 while hasFrame(video)
32     n_frame = n_frame + 1;
33     frame = readFrame(video);
34     if mod(n_frame, jump) ~= 0
35         continue
36     end
37     frameD = im2double(frame);
38     frameGris = im2gray(frameD);
39     prueba = abs(frameGris - fondoDGris);
40     prueba = prueba(:, [250:750]);
41
42     lvl = graythresh(prueba);
43     aAjustada = im2bw(prueba, lvl);
```

```

44 se = strel( 'disk' ,12);
45 aAjustadaEros= imerode(aAjustada , se );
46
47 se = strel( 'square' ,80);
48 aAjustadaFill = imfill(aAjustadaEros , 'holes' );
49 aAjustadaDil=imdilate(aAjustadaFill , se );
50
51 ImgFiltrada=medfilt2(aAjustadaDil);
52
53 ImgArea=bwareaopen(ImgFiltrada ,200000);
54 if mod(n_frame , jump) == 0
55     imshow(frame);
56 end
57 [L,num] = bwlabel(ImgArea);
58 rect= regionprops(L , 'BoundingBox' );
59 rprop = regionprops(L , 'all' );
60
61 [ alto ,~ ,~]=size(frame);
62 altura=floor( alto *2/3);
63
64 limSup=alto -1;
65 limInf=alto /2;
66 contFAct=0;
67
68 % Franja de deteccion
69 x=[1080 0];
70 line(x,[ limSup limSup] , 'Color' , 'g');
71 line(x,[ limInf limInf] , 'Color' , 'g');
72 if mod(n_frame , jump) == 0
73     for k=1: length(rect)
74         bb= rect(k).BoundingBox;
75
76         centro = rprop(k).Centroid;
77         text(centro(1)+250, centro(2), "*" , 'FontSize' ,20 , 'Color'
78             , 'g');
79
80         if centro(2)> limInf && centro(2)<limSup
81             contFAct=contFAct+1;
82             rectangle('Position' ,[bb(1)+250,bb(2),bb(3)+100,bb
83                 (4)] , 'LineWidth' ,2 , 'EdgeColor' , 'r' );
84             line(x,[ limSup limSup] , 'Color' , 'r' );
85             line(x,[ limInf limInf] , 'Color' , 'r' );
86         else
87             rectangle('Position' ,[bb(1)+250,bb(2),bb(3)+100,bb
                 (4)] , 'LineWidth' ,2 , 'EdgeColor' , 'c' );
88         end

```

```

88      end
89    end
90    if contFAnt > contFAct
91      contFAnt=contFAct;
92  end
93  if contFAnt < contFAct
94    cont=cont+1;
95    contFAct=contFAct+1;
96    contFAnt=contFAct;
97 end
98
99  if mod(n_frame, jump) == 0
100    pause(0.2);
101 end
102 end
103
104 waitfor(msgbox(['Se han encontrado ', num2str(cont), ' baterias']));
105 disp(['Se han encontrado ', num2str(cont), ' baterias'])
106
107
108 function fondo = calcular_fondo(file, fondo_file)
109
110 video = VideoReader(file);
111
112 B0 = 0;
113 B1 = 0;
114 count = 0;
115 alpha = 0.05;
116 n_frames = 0;
117 while hasFrame(video)
118   n_frames = n_frames + 1;
119   frame = readFrame(video);
120   if count ~= 0
121     B_t = (1-alpha) * B_previo + alpha * frame_anterior;
122   else
123     B_t = frame;
124     count = 1;
125   end
126   frame_anterior = frame;
127   B_previo = B_t;
128 end
129 fondo = B_t;
130 f = figure('Name', 'Fondo_Recuperado', 'NumberTitle', 'off');
131 imshow(fondo);
132 save(fondo_file, 'fondo')
133 waitfor(f);
134 end

```

0.7. Material utilizado

A continuación se dejan enlaces a todo el material utilizado, para uso en futuros trabajos, independientemente de si son de la universidad, proyectos públicos o privados, se ha de consultar al autor del documento.

- Vídeos utilizados.
 - Vid00
 - Vid01
 - Vid02
- Repositorio de Github
 - *source.m*
 - *source-testing.m*



UNIVERSIDAD
DE BURGOS



Copyright © 2021 Daniel Puente Ramírez All rights reserved.