Universidad del País Vasco Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

GRADO EN INGENIERÍA EN TECNOLOGÍA INDUSTRIAL

TRABAJO FIN DE GRADO

GENERACIÓN DE TRAYECTORIA DE UN OBJETO EN UN ENTORNO 3D MEDIANTE VISIÓN ARTIFICIAL

ANEXO 1 – *CÓDIGO DEL ALGORITMO*

Alumno Uribe Sáenz de Cámara, Sergio

Director: Eguiraun Martinez, Harkaitz

Curso: 2017-2018

Fecha: 12 de Julio de 2018

INDICE DE CONTENIDOS

INDICE DE FIGURAS

1. OBJETO DEL DOCUMENTO

El objeto del presente documento es el de proporcionar el código del algoritmo generado, de forma que el usuario pueda hacer uso de éste en cualquier momento.

El algoritmo ha sido desarrollado en la herramienta MATLAB, que es un programa matemático ideal para el tratamiento de imágenes, pero con un lenguaje propio. Por tanto, para la aclaración del lector, se harán explicaciones breves del contenido.

No obstante, todas las funciones usadas en el algoritmo están detalladamente explicadas en la guía de ayuda de MATLAB.

2. PASOS SEGUIDOS EN LA IMPLEMENTACIÓN DEL CÓDIGO

Como se ha explicado en la memoria del proyecto, el diagrama de bloques del proceso seguido es el de la Figura 1. El primer paso a realizar es determinar la colocación o la disposición de las cámaras, y una vez realizado este paso, tenemos dos caminos posibles. El segundo de ellos, trata del código generado en MATLAB mediante 5 scripts como se ve en la figura.

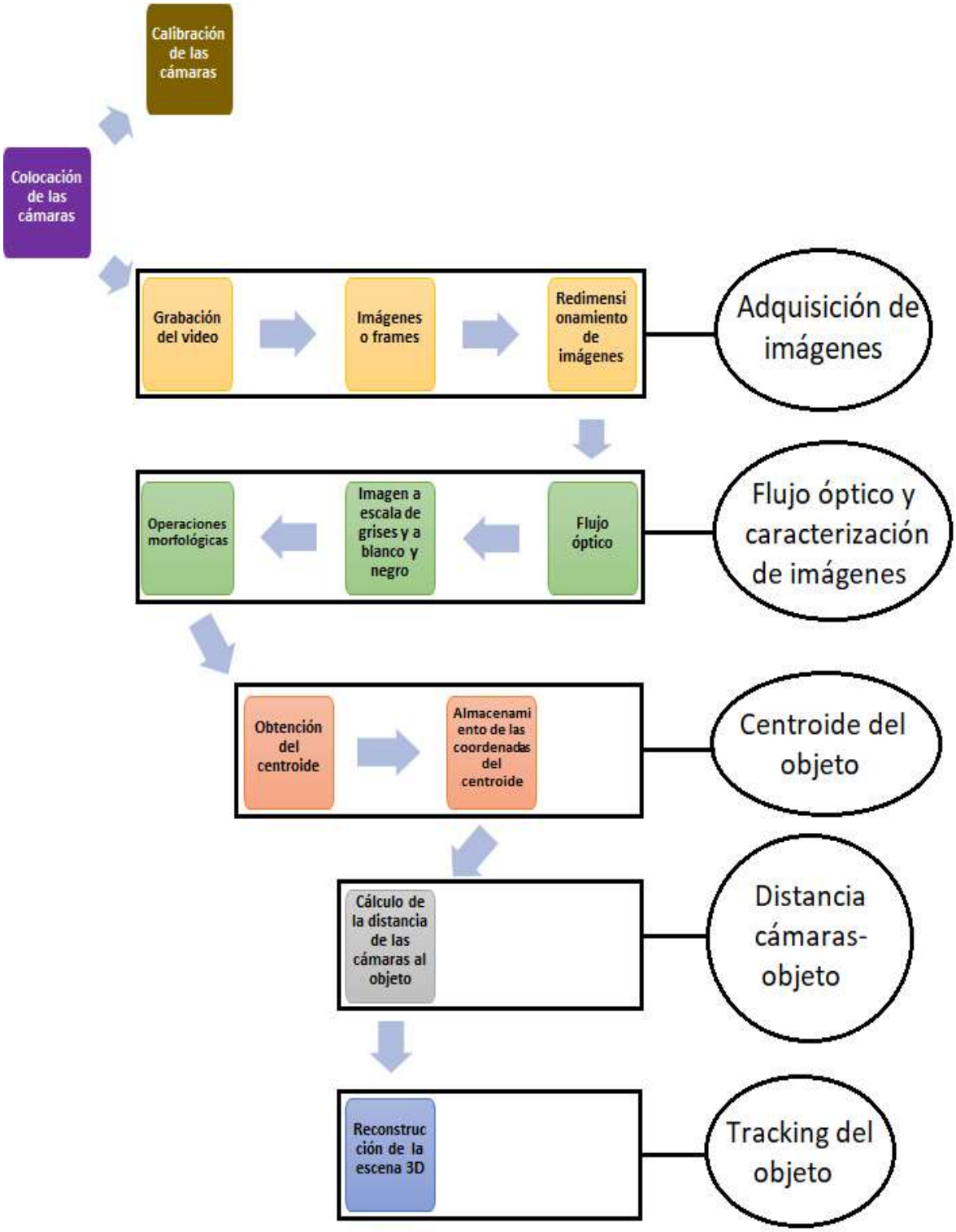

Figura 1: Diagrama de bloques del proceso seguido.

2.1 ADQUISICIÓN DE IMÁGENES

```
%% DESCOMPONER EL VIDEO EN FRAMES

% Left-images

cd 'C:/Users/sergi/Desktop/SERGIO URIBE/TFG/Scripts MATLAB/Script TFG coordenadas del objeto/Intento 6/GOPR002_L/GOPR002_L_Vuelta'

a=VideoReader('GOPR002_L_Vuelta.MP4');

for img = 1:a.NumberOfFrames;
    b = read(a, img);


    b=imresize(b,2.025);

    cd 'C:/Users/sergi/Desktop/SERGIO URIBE/TFG/Scripts MATLAB/Script TFG coordenadas del objeto/Intento 6/GOPR002_L/GOPR002_L_Vuelta/Frames'

   filename=strcat('GOPR002_L_',num2str(img),'.jpg');

   imwrite(b,filename);

end
```

1 2 3 4

1- Con el comando *'cd'*, MATLAB nos sitúa en la carpeta que se le indica. En dicha carpeta es donde se encuentra el video, que lo leemos con la función VideoReader como una variable a.
2- Con un ciclo FOR, leemos del video cada frame. El número total de ellos vendrá dado por el *NumberOfFrames* función vinculada al vídeo en la variable *a*.
3- Como se ha explicado en la Memoria, el tamaño de las imágenes del video vienen dadas en el tamaño 1280*960 pixeles, mientras que las de calibración son de: 2952*1944 pixeles. La relación entre las imágenes es de: 2.025. Por ello, modificamos su tamaño con un redimensionamiento.
4- Nos volvemos a colocar en la carpeta deseada para guardar las nuevas imágenes redimensionadas, dándoles el nombre que queremos, el cual se encuentra almacenado en la variable *filename*. Estas serán las imágenes que utilizaremos en el siguiente script.

El mismo script habrá que aplicar al video obtenido con la otra cámara.

2.2 FLUJO ÓPTICO Y CARACTERIZACIÓN DE IMÁGENES

```
%% CARACTERIZACIÓN DE LAS IMÁGENES RECTIFICADAS

for i=1:159

I1=imread (fullfile(sprintf('C:/Users/sergi/Desktop/SERGIO URIBE/TFG/Scripts MATLAB/Script TFG coordenadas del objeto/Intento 6/GOPR002_L/GOPR002_L_Vuelta/Frames/GOPR002_L_%d.jpg',i)));
J1=imread (fullfile(sprintf('C:/Users/sergi/Desktop/SERGIO URIBE/TFG/Scripts MATLAB/Script TFG coordenadas del objeto/Intento 6/GOPR002_L/GOPR002_L_Vuelta/Frames/GOPR002_L_%d.jpg',160 )));
    H = I1-J1;                                                                    1


   % Escala de grises y BW
    H_gray = rgb2gray(H);
    H_bw = im2bw(H_gray,0.085);                                                   2

   % Erosionar
    H_bw = bwmorph(H_bw,'clean');
    H_bw = bwareaopen(H_bw, 200);                                                 3


   % Agrandar las superficies
    se1=strel('disk',350);
    H_bw = imfill(H_bw,'holes');
    H2_bw = imclose(H_bw,se1);                                                    4

   %% Guardamos las imágenes en BW
    cd 'C:/Users/sergi/Desktop/SERGIO URIBE/TFG/Scripts MATLAB/Script TFG coordenadas del objeto/Intento 6/GOPR002_L/GOPR002_L_Vuelta/Frames BW '
    nombre=['GOPR002_L_BW_',num2str(i),'.jpg'];
    imwrite(H2_bw,nombre,'jpeg');                                                 5
end
```

1- En la variable *I1*, leemos las imágenes redimensionadas. Como son 159 las que deben ser leídas, habrá que realizar un ciclo FOR. Las imágenes tienen el siguiente nombre, GOPR002_L_%d.jpg, donde *%d* representa el número de frame, desde 1 hasta 159. Es por este motivo por lo que al final de la sentencia *I1*, aparece en negrita la varible i, que es la variable que recorre todos los frames, la cual se introduce en el parámetro %d.
   En la variable *J1*, leemos todo el rato la misma imagen, la del fondo o "background", la cual se llama GOPR002_L_160.jpg.
   Así, restando las variables *I1* y *J1*, obtenemos el flujo óptico.
2- Obtenidas las imágenes con el flujo óptico únicamente, éstas se convierten primeramente en escala de grises y posteriormente en blanco y negro aplicando el **umbral de binarización,** que en este caso hemos seleccionado el valor de 0.085. Se ha elegido de forma que junto al filtrado se llegase a la forma aproximada del objeto real, como se explica en la Memoria.
3- Se aplica el filtrado de erosión. Con el filtro *bwmorph(H_bw,'clean')* se consigue eliminar pixeles verdaderos aislados, es decir, pixeles con 1s en su celda pero rodeados de 0s.
   Con el filtro *bwareaopen* eliminamos pequeños objetos detectados que contienen menos de 200 pixeles.
4- Filtrado de dilatación. Con el filtro *strel,* agrandamos las superficies en la forma que queremos. Podemos hacerlo en forma de disco o "disk", pero también en forma de rectángulo, cuadrado, esfera, octógono… El 350 indica la distancia del centro del objeto hasta la superficie, es decir, está agrandando dicho área en forma de disco hasta un radio de 350 pixels.

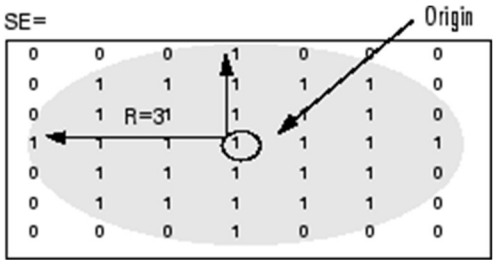

Figura 2: Filtro strel para un radio de 3 pixeles.

   Los filtros *imfill y imclose* cierran pequeños huecos que pueden quedar en el objeto.

5- Por último, guardamos con el nombre deseado las imágenes en blanco y negro, con el objeto de interés segmentado y aproximado a su forma real, de nuevo con el comando *'cd'*.

El mismo script habrá que aplicar a las imágenes de la otra cámara.

2.3 CENTROIDE DEL OBJETO

```
. %% OBTENCIÓN DEL CENTROIDE


inicio=1;
final=159;


for ii=inicio:final

%Insertamos la imagen en BW!!!!
%Left images
A=imread(fullfile(sprintf('C:/Users/sergi/Desktop/SERGIO
URIBE/TFG/Scripts MATLAB/Script TFG coordenadas del
objeto/Intento 6/GOPR002_L/GOPR002_L_Vuelta/Frames
BW/GOPR002_L_BW_%d.jpg',ii)));


  [Object num]=bwlabeln(A,8);

   if num==0

     CoorX_L(1,iii)=0;
     CoorY_L(1,iii)=0;
      continue %sale del ciclo y pasa al siguiente iii
   end

   data=regionprops(Object);
   idx = find([data.Area] > 100);


  props(1,iii).Centroide = data(idx,1).Centroid;

  CoorX_L(1,iii)=props(1,iii).Centroide(1,1);
  CoorY_L(1,iii)=props(1,iii).Centroide(1,2);
end
```

1

2

3

4

1- Insertamos las imágenes en blanco y negro una a una obtenidas como resultado en el script anterior. De nuevo, lo haremos con el parámetro %d.
2- La función bwlabeln proporciona los objetos encontrados en la imagen y el número de objetos que existen. En caso de que no encuentre ninguno, que no va a ser en nuestro caso ya que el objeto está siempre dentro de la imagen, se le dice que salga del ciclo y pase a la siguiente imagen.
3- La función *regionprops* mide las propiedades de las regiones de la imagen. Por defecto, nos proporciona el área, el centroide y el "bounding box" o el rectángulo más pequeño que lo contiene de cada objeto que encuentra en la imagen.
   Es posible que se encuentren más de un objeto a pesar de saber que sólo se tiene uno. Pero observando el área, estos tienen un área minimo, probablemente de 1x1 pixel, mientras que nuestro objeto de interés va a tener un área mucho mayor siempre.
   Por este motivo, mediante la función *find* se busca nuestro objeto de interés en función del área, que será el único con un área mayor a 100.
4- Almacenamos las coordenadas del centroide en dos matrices, una para las coordenadas en el eje X y otra para el eje Y. Estas matrices serán las que usaremos en el próximo script.

El mismo script habrá que aplicar a las imágenes de otra cámara.

2.4 DISTANCIA DE LAS CÁMARAS AL OBJETO

```
%% CÁLCULO DE LA DISTANCIA DEL OBJETO A LAS CÁMARAS
%Habrá que implementar la ecuación Z=f*b/d donde
%f=distancia focal.
%b=distancia entre cámaras
%d=disparidad. diferencia de las coordenadas en ambas imágenes

f=1224; %pixel
b= 24.6; %cm

load( 'C:/Users/sergi/Desktop/SERGIO URIBE/TFG/Scripts MATLAB/Script TFG coordenadas del objeto/Intento 6/Scripts TFG/CoorX_Vuelta/CoorX_L_Vuelta.mat');
load( 'C:/Users/sergi/Desktop/SERGIO URIBE/TFG/Scripts MATLAB/Script TFG coordenadas del objeto/Intento 6/Scripts TFG/CoorX_Vuelta/CoorX_R_Vuelta.mat');

[i,final]=size (CoorX_R);

for i=1:1:final %% for i=1:final
  d(i)=abs(CoorX_L(1,i)-CoorX_R(1,i)); %pixels  %%

  Z(i)=f*b/d(i); %%en cm
end

cd 'C:/Users/sergi/Desktop/SERGIO URIBE/TFG/Scripts MATLAB/Script TFG coordenadas del objeto/Intento 6/Scripts TFG/DistanciaZ'
nombre=['DistanciaZ.mat'];
save(nombre,'Z');
```

1 2 3 4

1- Se introducen los valores de la distancia focal y la distancia entre las cámaras. Además, con la función *load* cargamos las matrices de coordenadas en el eje X.
2- Se realiza el cálculo de la disparidad para las dimensiones de la matriz de coordenadas, que corresponde con el número de imágenes. La disparidad viene dado por: $d = x_I - x_D$, siendo x_I y x_D las coordenadas del objeto en el eje X de la imagen izquierda y derecha respectivamente. Se le aplica el valor absoluto, porque la distancia al objeto debe ser positiva.
3- Se realiza el cálculo de la distancia de las cámaras la objeto, dado por la ecuación:

$$Z = f\frac{b}{d}$$

4- Se guarda la variable Z en la carpeta deseada para el uso de ésta en el siguiente script.

2.5 RECONSTRUCCIÓN DE LA ESCENA 3D.

```
%% RECONSTRUCCIÓN Y GENERACIÓN DE TRAYECTORIA

load('C:/Users/sergi/Desktop/SERGIO URIBE/TFG/Scripts MATLAB/Script TFG coordenadas del objeto/Intento 6/Scripts TFG/CoorX_Vuelta/CoorX_L_Vuelta.mat');
load('C:/Users/sergi/Desktop/SERGIO URIBE/TFG/Scripts MATLAB/Script TFG coordenadas del objeto/Intento 6/Scripts TFG/CoorY_Vuelta/CoorY_L_Vuelta.mat');
load('C:/Users/sergi/Desktop/SERGIO URIBE/TFG/Scripts MATLAB/Script TFG coordenadas del objeto/Intento 6/Scripts TFG/DistanciaZ/DistanciaZ.mat');
load('C:/Users/sergi/Desktop/SERGIO URIBE/TFG/Scripts MATLAB/Script TFG coordenadas del objeto/Intento 6/Calibración/stereoParams.mat');

%% Punto central de la imagen (Xo)
Xo=stereoParams.CameraParameters1.PrincipalPoint(1);      %% LEFT = CAMERA 1
Yo=stereoParams.CameraParameters1.PrincipalPoint(2);
%% Implementamos las fórmulas para sacar Xc e Yc
f=1224; %pixel

for i=1:1:160
    Xc(1,i)=Z(1,i)*abs((CoorX_L(1,i)-Xo))/f;
    Yc(1,i)=Z(1,i)*abs((CoorY_L(1,i)-Yo))/f;
end

plot3(Z,Xc,Yc)
xlabel 'Z';
ylabel 'X';
zlabel 'Y';
```

1

2

3

1- Habrá que introducir como elementos de entrada: las matrices de coordenadas del centroide del objeto, la distancia Z de las cámaras al objeto, los parámetros de calibración para obtener las coordenadas del punto principal y la distancia focal.
2- Se implementan las ecuaciones para determinar las coordenadas X e Y del objeto, y al igual que antes se aplica el valor absoluto.

$$X = \frac{Z}{f}(x_I - x_o) \qquad Y = \frac{Z}{f}(y_I - y_o)$$

3- Se realiza la representación gráfica con la función plot3, que nos permite realizar un gráfico en 3 dimensiones. El tracking del objeto será lo representado y por tanto, el resultado final.