```
Mat H = findHomography( obj, scene, CV_RANSAC );

//*********PASO 5: Uso de la matriz de Homografía, para alinear ambos fotogramas***********//
cv::Mat result;
warpPerspective(gray_image1,result,H,gray_image2.size());      // Transformación de fotograma
subtract(gray_image2,result,rest);
subtract(result,gray_image2,rest2);
rest=rest+rest2;
imwrite("final.bmp",rest);
imshow("Resta entre el fotograma previo transformado y el fotograma actual",rest);

//*************PASO 6: Imprimimos los resultados de todos los pares de fotogramas*************//
// Se almacenan todos los resultados de los alineamientos entre todos los pares de fotogramas
//en una carpeta.
snprintf(file_output,80,"/home/jhon/Desktop/OpenCV/opencv-2.4.9/samples/cpp/
        Nombre_carpeta/Name_%03d.bmp", j);
imwrite(file_output,rest);
j=j+1;


}
waitKey(0);
return 0;
}

// Fin del programa que se encarga de generar y almacenar las imágenes resultantes  del
alineamiento y resta entre cada transformada de fotograma previo y fotograma actual.
```

ANEXO 2

CODIGO DESARROLLADO EN MATLAB

Primera etapa de detección y seguimiento de blobs

Este programa permite detectar todos los blobs que se encuentran en las imágenes resultantes del alineamiento y resta de los fotogramas previos y actuales. Posteriormente se procede a almacenar todas las posiciones de todos los blobs.

```
% Se procede a extraer todas las imágenes resultantes almacenadas en el fichero
file=dir('C:\Users\Jhon\drive3\Nombre_carpeta\*.bmp');
imagenes=imageSet('drive3\Nombre_carpeta\');
filtro=fspecial('gaussian',5,1.5);
se=strel('disk',1);
a=0; contador=0;
aux=0;                          %permite que el los frames diferentes al primero se lean
frameini(1).x=0;
frameini(1).y=0;
frameini(1).idx=0;
frameni(1).Area=0;
DistanciaObjetos=30;            % Distancia máxima entre centroides, definido por nosotros
dist=360;                       %inicializamos la distancia entre 2 centroides

% se define una estructura que posee tres campos, la cual almacenará las características de
%los blobs detectados, además “g”, representa el parámetro de número de fotogramas,
%indicándonos en que fotograma se encuentra el blob; e “inicio” representa el número de blobs
%que se espera encontrar.
% se procede a colocar valores de ceros ’0 ’, de tal manera que tenga un valor inicial.
blob=struct('Coordenadas',struct('x',0,'y',0,'area',0));
for inicio=1:300
        for g=1:1000
        blob(inicio).Coordenadas(g).x=0;
        blob(inicio).Coordenadas(g).y=0;
        blob(inicio).Coordenadas(g).area=0;
        end
end
%  Se procede a leer todas las imágenes contenidas en el fichero
%%%%%%%%%%%%%%%% SEGMENTACIÓN DE IMAGEN %%%%%%%%%%%%%%%%%
for i=1:length(file)
        ima=read(imagenes,i);
        ima=imfilter(ima,filtro);
        imshow(ima);
        im=im2bw(ima,0.08);                 %Umbral para vehículos 0.08, personas 0.05
        [alto, ancho]=size(im);             % Se  extrae las dimensiones de la imagen
        im=imclearborder(im);               % Se inicia el uso de operaciones morfológicas
        BW=bwmorph(im,'clean');
        BWb=bwmorph(BW,'bridge',7);
        BWfill=bwmorph(BWb,'fill');
        Barea=bwareaopen(BWfill,30);
        Bf=imfill(Barea,'holes');
        distancia=bwdist(Bf) <=1;
        bds=bwareaopen(distancia,300);
```

```
        bds=imclose(bds,se);
        mascara=zeros(alto,ancho);              %Crea una matriz con dimensiones extraídas

        % Se procede a contar todos los blobs de una imagen resultante
        s = regionprops(bds,'Centroid','BoundingBox','Area');
        bds=bwlabel(bds);
        numobj=numel(s);
        % se hace uso de los criterios de eliminación de los blobs que no cumplan con lo
        %acordado en el método y solo se quedan los blobs que pasen dichos criterios y se
        %almacenan en una matriz.
        for k=1:numobj
                arearect= s(k).BoundingBox(3)* s(k).BoundingBox(4);
                relacion=s(k).Area/arearect;

                if( s(k).Area>350 && ~(s(k).BoundingBox(3)>3.5* s(k).BoundingBox(4)) &&
                ~(s(k).BoundingBox(4)>3.5* s(k).BoundingBox(3)))
                        mascara=mascara+(bds==k);
                end
        end
% Algunas operaciones morfológicas
mascara=bwdist(mascara) <=1;
mascara=bwmorph(mascara,'bridge',3);
mascara=bwmorph(mascara,'fill');
mascara=imfill(mascara,'holes');
mascara=imerode(mascara,se);
% se crea las imágenes que contienen solo blobs que pasaron los criterios de %eliminación.
fileoutput=sprintf('to%03d.bmp',i);
imwrite(mascara,fileoutput);

%%%%%%%%%%%%%%% SEGUIMIENTO DE BLOBS %%%%%%%%%%%%%%%%%%%

% Se inicia analizando la primera imagen resultante obtenida después de la etapa de
%segmentación de imagen.
if(contador<1)
        r=regionprops(mascara,'Centroid','Area','BoundingBox');
        region=bwlabel(mascara);
        numBlob1=numel(r);
        for h=1:numBlob1
                frameini(h).x=r(h).Centroid(2);
                frameini(h).y=r(h).Centroid(1);
                frameini(h).idx=h;
                frameini(h).area=r(h).Area;
                blob(h).Coordenadas(1).x=r(h).Centroid(2);
                blob(h).Coordenadas(1).y=r(h).Centroid(1);
                blob(h).Coordenadas(1).area=r(h).Area;
        end
        contador=contador+1;                    % Solo se una vez y es para la primera imagen
        previo=frameini;
        numPuntos=numBlob1;
        numero_objetos=numBlob1;        % Indicador de numero de blobs en la secuencia
        f=2;
        b=2;
end
% Se inicia analizando las distancias entre pares de imágenes resultantes
if(aux>0)
r=regionprops(mascara,'Centroid','Area','BoundingBox');
region=bwlabel(mascara);
numBlob2=numel(r);
        for j=1:numBlob2
        framesec(j).x=r(j).Centroid(2);
```

```
            framesec(j).y=r(j).Centroid(1);
            framesec(j).idx=j;
            framesec(j).area=r(j).Area;
            end
% Se inicia el cálculo de distancias entre blobs de pares de imágenes
b=1;
for Operaciones=1:numBlob2              % Blobs de la segunda imagen (Imagen posterior)
   for numDist=1:numPuntos              % Blobs de la primera imagen (imagen inicial)
    dist_calculada=((frameini(numDist).y-framesec(Operaciones).y)^2 +(frameini(numDist).x-
    framesec(Operaciones).x)^2)^0.5;

          if(dist_calculada<=dist && dist_calculada<= DistanciaObjetos)
             dist=dist_calculada;
             tx=frameini(numDist).x;
             ty=frameini(numDist).y;
             m=numel(blob);

             for objetos=1:numero_objetos
                for cuadros=1:f
                   if(blob(objetos).Coordenadas(cuadros).x == tx)
                   blob(objetos).Coordenadas(cuadros+1).x=framesec(Operaciones).x;
                   blob(objetos).Coordenadas(cuadros+1).y=framesec(Operaciones).y;
                   blob(objetos).Coordenadas(cuadros+1).area=framesec(Operaciones).area;
                   end
                end
             end
          end
          % se termino de calcular las distancias de un blob con los blobs de otra imagen
          if(numDist==numPuntos)
             if(dist>=32)

          blob(numero_objetos+1).Coordenadas(f).x=framesec(Operaciones).x;   % Coordenadas
          del centroide del nuevo BLOB
                blob(numero_objetos+1).Coordenadas(f).y=framesec(Operaciones).y;
                blob(numero_objetos+1).Coordenadas(f).area=framesec(Operaciones).area;
                numero_objetos=numero_objetos+1;            % indicador de que se CREO y
ALMACENO un nuevo BLOB
             end
          end
   end
dist=360;         %Se inicializa el valor de dist con 360, pues ya se terminó de analizar el blob
                  %de una imagen con los blob de la siguiente imagen
   a=0;
end
f=f+1;              %nos permite cargar los valores al paso de la 3era imagen (imagen posterior
                     %de imagen 2 )
frameini=framesec;       %Recien ahora podemos trabajar con imagen inicial y posterior
numPuntos=numBlob2;     %Para que los parametros pasen de la imagen posterior y se
                         %carguen a la imagen inicial,
end
aux=1;

end

% Una vez realizado este proceso para todas las imágenes resultantes obtenidas de la
%segmentación de imagen, se generará una variable conteniendo todos los blobs detectados a
%lo largo de la secuencia de imágenes.

nu=numero_objetos;
```

Segunda etapa de detección y seguimiento de blobs

En esta etapa eliminaremos aquellas posiciones iguales a cero “0”, puesto que estas coordenadas solo indica que el blob detectado no se encontraba presente en la imagen.

```
% Creamos una nueva estructura donde se almacenarán las coordenadas distintas de cero.

bl=struct('autos',struct('x',{},'y',{},'frame',{}, 'area',{}));
sec=1;
numero_objetos=nu;
% eliminaremos aquellas coordenadas iguales a 0
for b=1:numero_objetos
   no_mas=2; a=1;frame=1;
   while(frame<=f && no_mas~=1)
         if(no_mas>1)
           if(blob(b).Coordenadas(frame).x ~=0)
                  bl(sec).autos(a).x=blob(b).Coordenadas(frame).x;
                  bl(sec).autos(a).y=blob(b).Coordenadas(frame).y;
                  bl(sec).autos(a).area=blob(b).Coordenadas(frame).area;
                  bl(sec).autos(a).frame=frame;
                  a=a+1;
                  if(blob(b).Coordenadas(frame+1).x==0)
                        no_mas=1;
                  end
            end
         end
         frame=frame+1;
   end
   sec=sec+1;
end
```

Tercera etapa de detección y seguimiento de blobs

Esta etapa se encarga de contar los blobs detectados, los cuales se encuentran presentes más de 3 veces en la secuencia de imagen, ya que se estimó que un objeto detectado no puede aparecer en un fotograma, desparecer en otro y finalmente aparecer en un siguiente fotograma. Con esto logramos obtener el número de vehículos y personas moviéndose en una secuencia de imágenes tomadas desde un UAV(cámara en movimiento)

```
objetos_finales=0;
%contar los autos
sec=sec-1;  % disminuimos en uno, puesto cuando sale del FOR anterior se le aumentó

for obj=1:sec
    aa=length(bl(obj).autos);
    if(aa>3)
        objetos_finales=objetos_finales+1;
        mayor=bl(obj).autos(1).area;

    end
end
```