A Annexes

A.1 Guia d'usuari i configuració del software

A.1.1 Requisits

Per executar el programa és necessari disposar d'un ordinador on estigui instal·lat el programa MATLAB. Aquest esta disponible per Linux, OS X i Windows.

Els arxius del software de detecció i de tractament de dades anomenats programa Principal.m i tractamentDades.m.

Un fitxer de vídeo on s'hi vulguin detectar els vehicles en format de vídeo: mov, avi, mp4, etc.

A.1.2 Recomanacions

La versió que s'ha utilitzat de MATLAB és la 2016a. Pot funcionar perfectament en versions anteriors però empra paquets de visió per computador que pot ser que les versions velles no els disposin.

Tenir instal·lat un programa de fulls de càlculs per llegir el format resultant. Com pot ser OpenOffice o Microsoft Office.

A.1.3 Passos previs

Tenir els fitxers programaPrincipal.m, tractamentDades.m i el vídeo localitzats en l'ordinador. Com per exemple a l'escriptori.

Saber la distància real entre dos punts que apareguin dins el vídeo.

A.1.4 Execució pas per pas

A continuació es resum en 7 passos com executar el programa.

1. Obrir el programa MATLAB.
2. Canviar el directori de treball per aquell on estiguin els fitxers del programa i el vídeo.

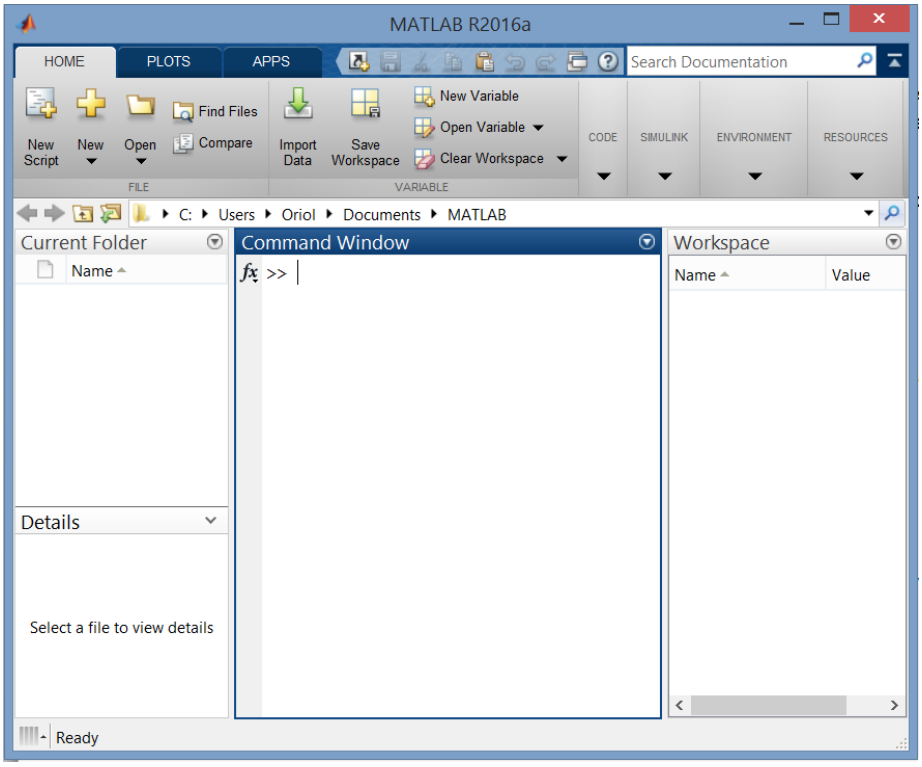

Figura A.1: Pantalla principal de MATLAB.

Per canviar el directori de treball s'ha de fer clic a la barra superior que en aquest cas és:

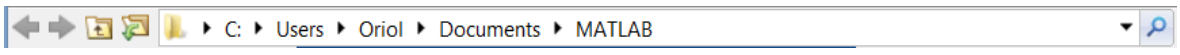

Figura A.2: Barra de direcció de MATLAB.

I s'habilita l'opció d'introduir manualment la ruta dels fitxers. Si es troben a l'escriptori la ruta a utilitzar és per defecte `C:\Users\Usuari\Desktop`.

Figura A.3: Canvi del directori de treball.

Si s'ha fet bé al requadre on diu *Current Folder* haurien d'aparèixer els arxius.

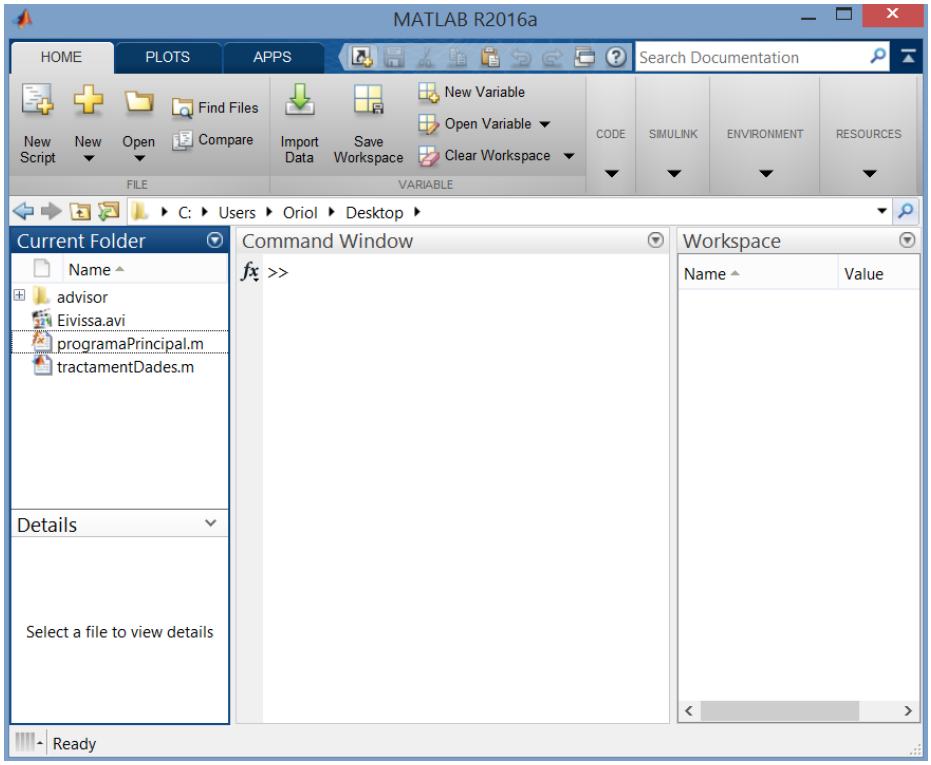

Figura A.4: Pantalla principal de MATLAB amb el directori de treball corresponent.

3. Execució del programa.

Per fer-ho s'ha de cridar el programa principal a la consola de comandes, Command Window, de la següent manera:

T = programaPrincipal(fitxerVideo)

Al camp de fitxerVideo s'ha de posar el nom del vídeo amb el format. El vídeo d'exemple s'anomena Eivissa.avi per tant s'introdueix entre cometes dins del parèntesis:

T = programaPrincipal('Eivissa.avi')

Es prem introduir i el programa s'executarà.

4. Calibratge de la distància real.

Amb el programa executat el primer que demana és el calibratge de la distància real. En una finestra ho recorda. Prem OK.

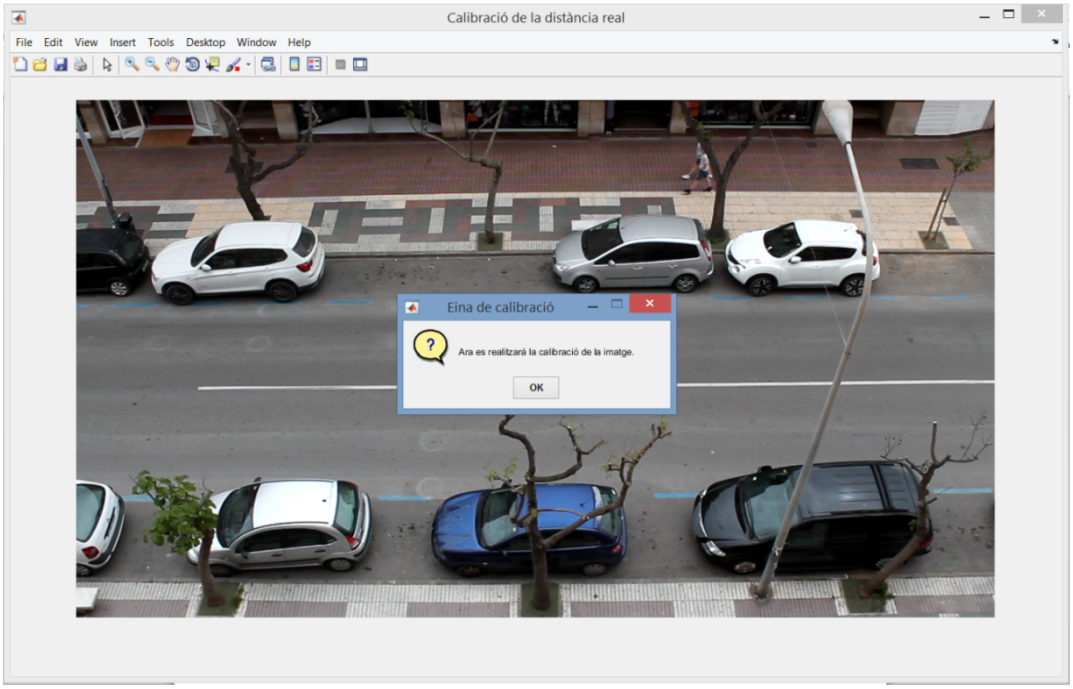

Figura A.5: Finestra de calibratge de la distància real.

Sortirà una altra finestra que explica com fer el calibratge i la introducció dels punts i distància entre aquests. L'eina és visual. Primer demana que es dibuixi una recta entre dos punts fent dos clics. El primer clic s'introdueix amb el clic esquerra del ratolí i el segon clic és realitza amb el clic dret. Prem OK i saltarà l'eina de dibuix.

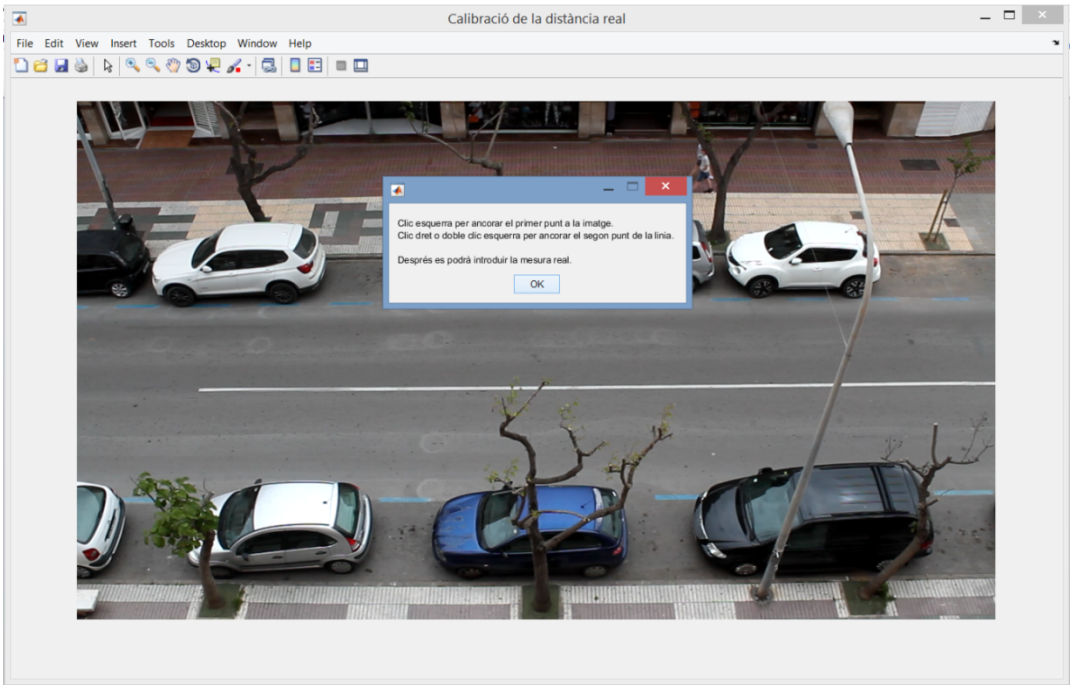

Figura A.6: Finestra de calibratge de la distància real.

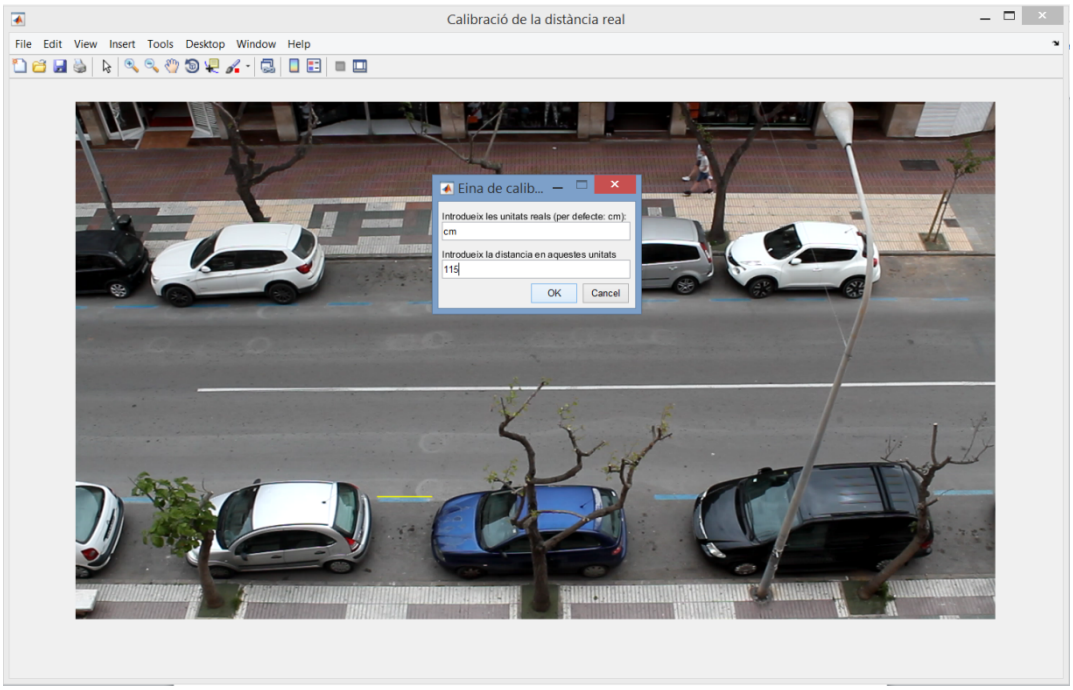

Figura A.7: Introducció de la mesura real.

Per defecte el programa empra centímetres i és recomanable deixar-ho així perquè el programa de càlcul d'acceleracions i velocitats després fa els factors de conversió cap a m/s^2 i km/h respectivament en base a centímetres. La distància d'aquesta mesura és de 115 cm aproximadament s'introdueix dins el camp amb el teclat de l'ordinador i es prem OK.

5. Selecció de l'àrea de treball.

Ara pregunta per l'àrea de treball. Aquesta àrea és aquella on s'hi vulguin detectar objectes. Es prem OK i començarà la selecció.

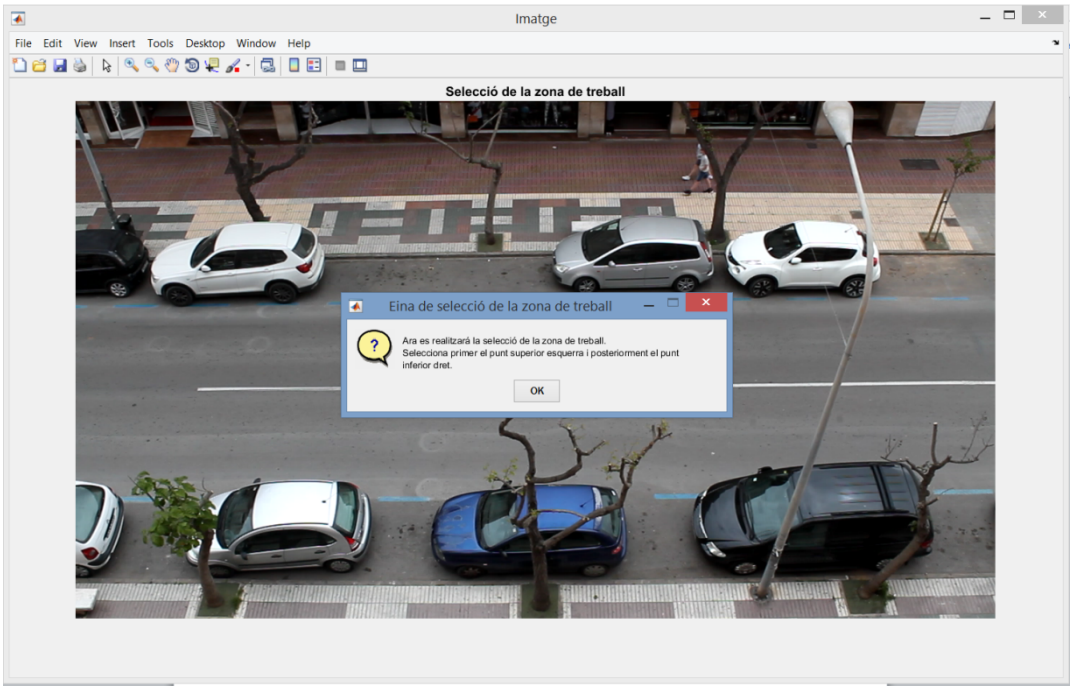

Figura A.8: Finestra de selecció de l'àrea de treball.

Primer s'ha de seleccionar el vèrtex superior esquerra i posteriorment l'inferior dret. Tot amb el clic esquerra del ratolí. Com son vehicles es selecciona tota la part de la calçada. A la figura A.9 es detalla gràficament l'ordre dels clics.

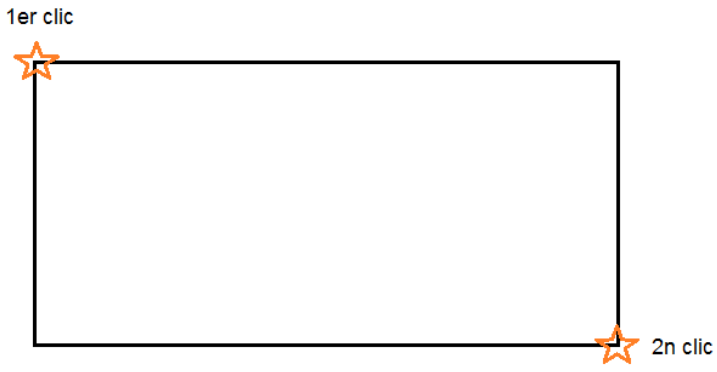

Figura A.9: Ordre en que s'han de fer els clics.

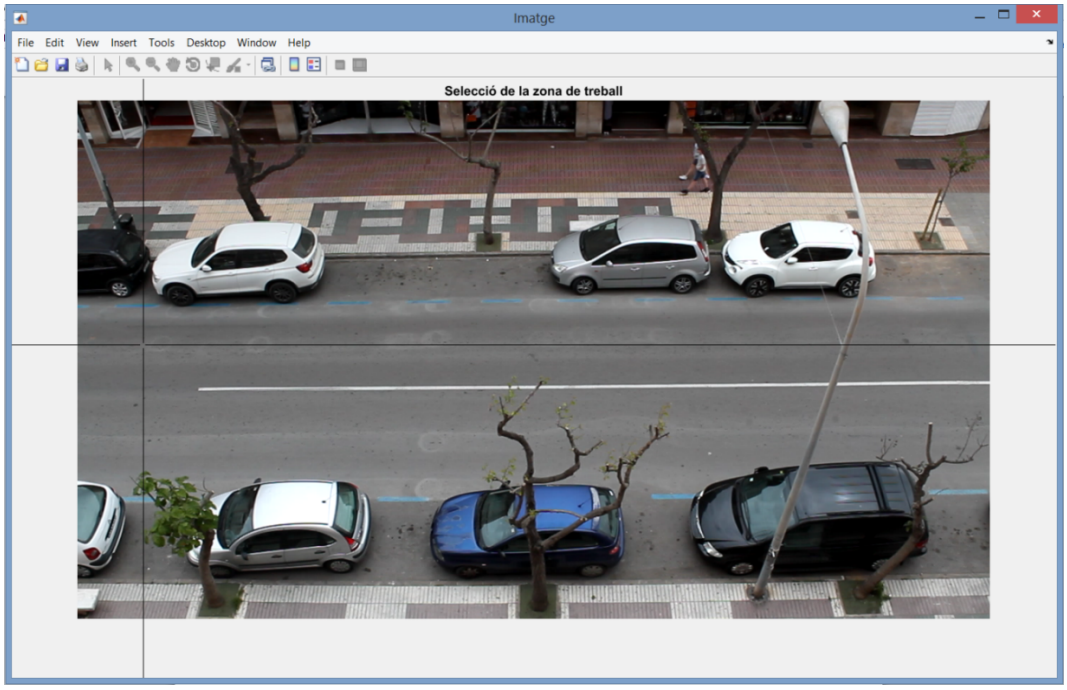

Figura A.10: Eina de selecció de l'àrea de treball.

A partir d'aquest punt el programa s'executarà per tot el vídeo. S'obriran dos reproductors de vídeo on s'hi veu el vídeo principal amb l'àrea de treball seleccionada emmarcada en un rectangle blanc i el vídeo on s'estan realitzant les deteccions en l'àrea de treball.

Figura A.11: Reproductor de vídeo amb deteccions.

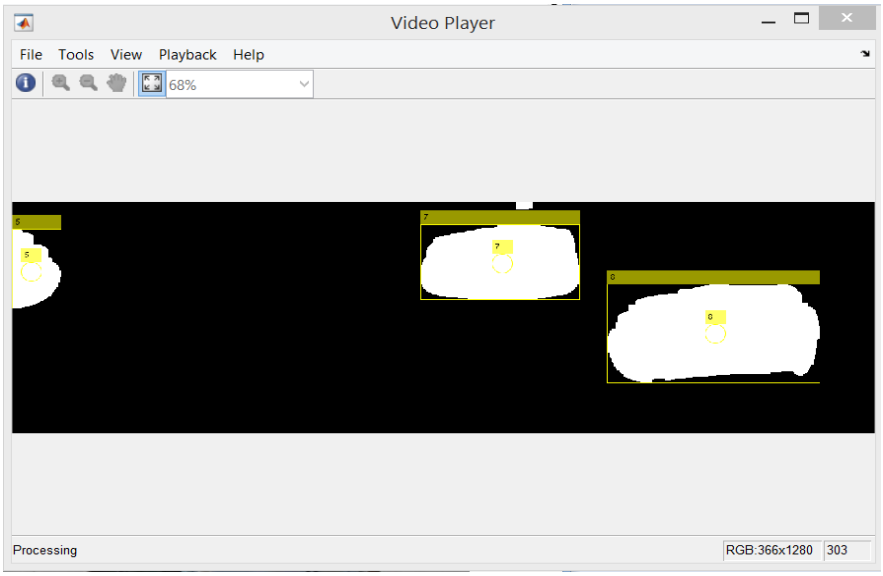

Figura A.12: Reproductor de vídeo de la màscara.

6. Resultats de les deteccions.

Quan acaba l'execució al *Workspace* ha d'aparèixer la variable T que conté tots els vehicles detectats.

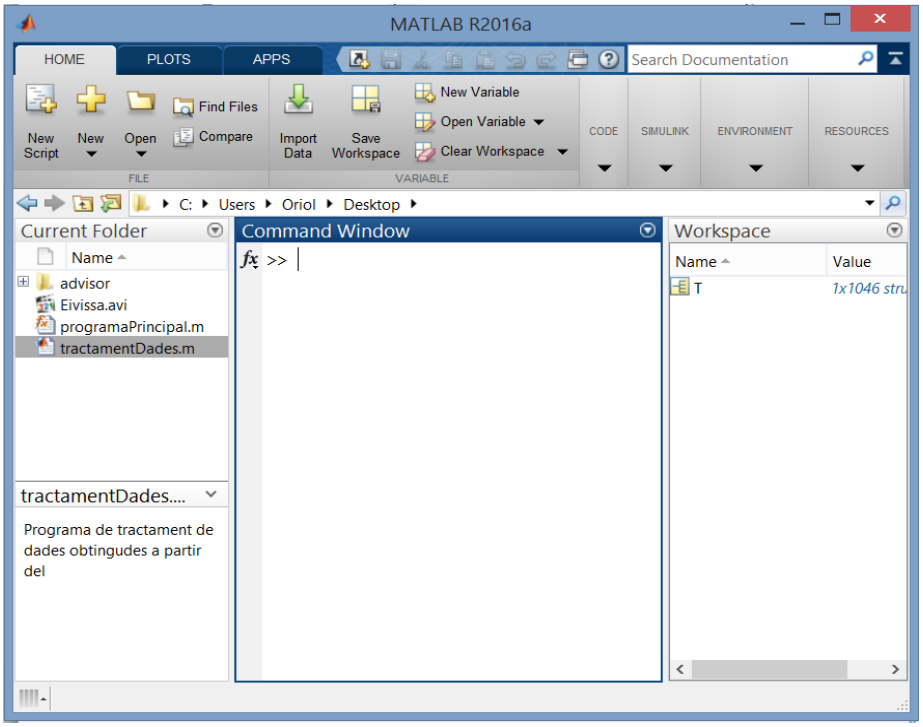

Figura A.13: Finestra principal de MATLAB amb la variable que emmagatzema els vehicles detectats.

7. Càlcul de les acceleracions i velocitats.

Amb la variable T al Workspace ara es pot executar el programa tractamentDades.m. Per a fer-ho es pot fer de dues maneres:

i. Arrossegant el fitxer des de Current Folder fins a Command Window

ii. Obrint el programa amb l'editor fent doble clic esquerra i prement el botó Run.

Si es fa de la segona manera:

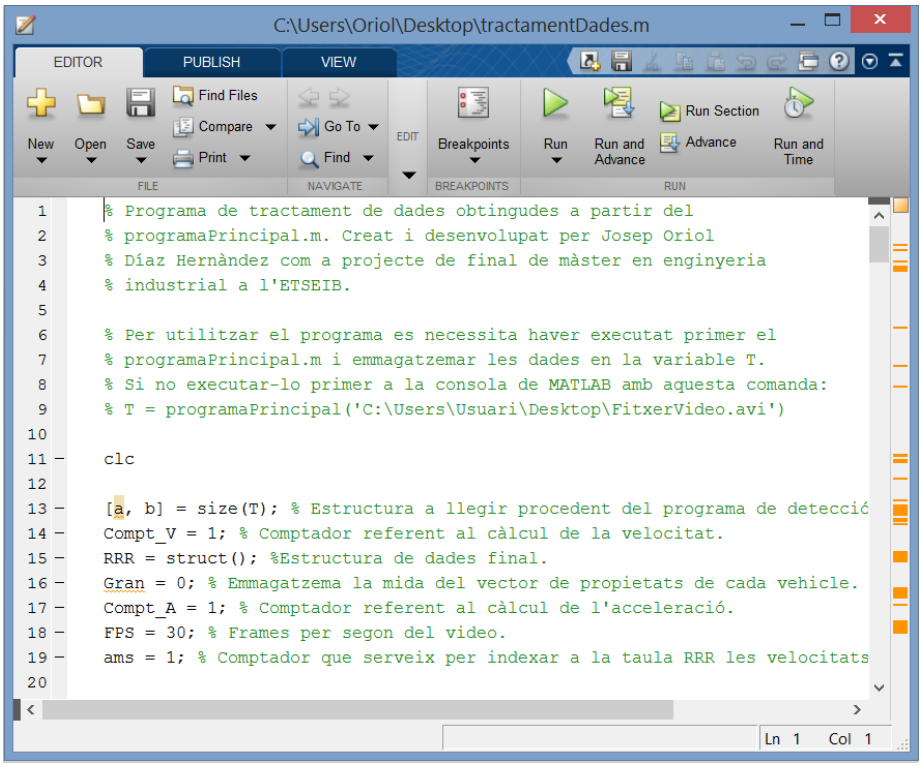

Figura A.14: Editor de codi de MATLAB.

Es prem el boto *Run* i s'executa. Quan acaba s'haurien d'haver creat els fitxers de resultats en text i full de càlcul. Al full de càlcul es troben separats per fulles els vehicles detectats. Inclou una darrer fulla anomenada "Resultats" on es troben tots els resultats de tots els vehicles. Tot en un mateix arxiu de full de càlcul.

Cada fulla és una etiqueta de vehicle. Es possible que hi hagi alguns fulls en blanc per falta de punts.

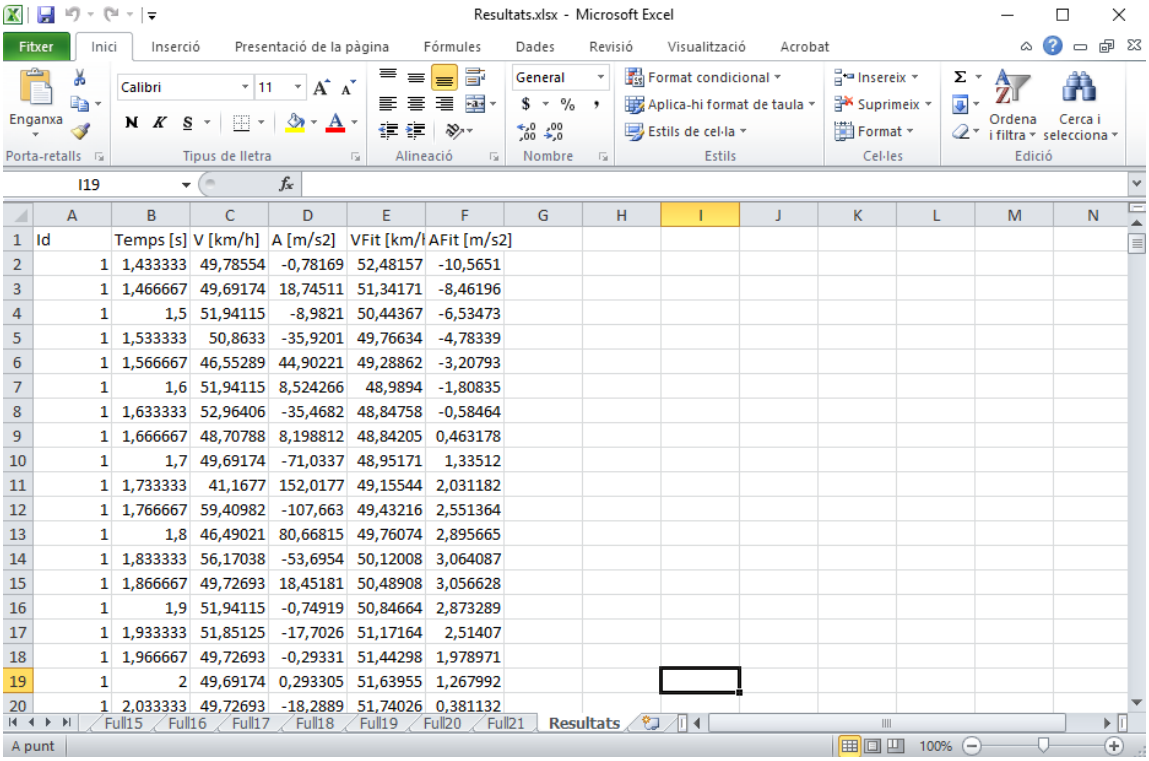

Figura A.15: Resultats en el fitxer de full de càlcul.

A.1.5 Configuració del programa principal de detecció i seguiment de vehicles

El programa de detecció té quatre paràmetres que es poden configurar per adaptar-se a les condicions del vídeo d'entrada. És indispensable tenir-los ben configurats per poder realitzar correctament les deteccions en l'escenari que es vulgui.

A.1.5.1 Configuració de la mida dels vehicles en el detector d'objectes

La mida dels vehicles pot variar en funció de l'altura de gravació. El programa de detecció conté una funció que s'anomena *vision.BlobAnalysis* situat a la línia del codi número 73.

```
73      % Configuració del blob analyzer.
74 -    obj.blobAnalyser = vision.BlobAnalysis('BoundingBoxOutputPort', true, ...
75              'AreaOutputPort', true, 'CentroidOutputPort', true, ...
76              'MinimumBlobArea', 5000, 'MaximumBlobArea', 45000);
```

Figura A.16: Línia de codi on es troba la configuració del detector d'objectes.

Els paràmetres que s'han de canviar són els de MiniumBlobArea i MaxiumBlobArea. Respectivament 5000 i 45000 són les àrees en píxels mínimes i màximes que es detecten.

MiniumBlobArea: si es poden àrees molt petites qualsevol element petit l'entendrà com un objecte i posarà moltes etiquetes a tot. Si es posen àrees massa grans pot ser que no detecti res.

MaxiumBlobArea: limitar la grandària és fonamental perquè no detecti dos o més cotxes com un quan hi ha efecte de solapament. Aquest efecte s'explica en detall a l'apartat 4.6.1. Si es posa un valor petit pot ser que tampoc detecti res.

A.1.5.2 Configuració de la creació de tracks segons el cost de no assignació

El cost de no assignació es troba a la línia del codi número 191.

```
189        % Resol el problema d'assignació. assignDetectionsToTracks funció
190        % matlab.
191 -      costDeNoAssignacio = 19;
192 -      [assignacions, tracksAmbVehiclesInvisibles, novesDeteccionsSenseTrack] = ...
193            assignDetectionsToTracks(cost, costDeNoAssignacio);
```

Figura A.17: Línia de codi on es troba el cost de no assignació.

Aquest valor s'ha de calibrar experimentalment. Si es posa un valor petit s'incrementa la probabilitat de crear un nou *track* i pot resultar en una fragmentació de les deteccions. És a dir que el mateix vehicle rebi més d'una etiqueta. Per contra si es deixa massa baix pot ser que només es creï un sol *track* corresponent a diferents vehicles en moviment. És a dir que agafi amb la mateixa etiqueta més d'un vehicle diferent.

Per saber més sobre que és un *track* i com es creen les etiquetes de cada vehicle i s'assignen les deteccions es recomanable llegir en profunditat l'apartat 4.3.3.

A.1.5.3 Quan es dóna un vehicle per perdut?

Pot ser que es detecti un vehicle però que per certes circumstàncies es perdi. Per esborrar aquest vehicles dels que no es tindran casi dades es poden canviar dos valors. S'ha d'anar a la línia de codi número 235.

```
235        % Valor que borra els tracks segons el numero de fotogrames que no
236        % s'ha vist o segons la edat minima.
237 -      invisibleForTooLong = 10;
238 -      edatMinima = 8;
```

Figura A.18: Línia de codi on es troben els paràmetres d'edat mínima i invisibilitat.

invisibleForTooLong és la variable que s'encarrega de dir que si s'ha passat 10 fotogrames invisible s'esborri.

edatMinima és la vida mínima del vehicle. L'edat és el nombre de fotogrames que han passat des de la primera detecció.

Els *tracks* s'esborren quan l'edat es inferior a la mínima i la visibilitat és inferior a 1/3 o quan la invisibilitat és superior o igual al inivisbleForTooLong.

Per saber més consultar en detall l'apartat 4.3.3.

A.1.5.4 Visibilitat mínima per començar l'enregistrament i visualització

La darrer variable que es pot configurar és el minVisibleCount. Aquesta es troba a la línia de codi número 297 i fa referència al nombre mínim de fotogrames que han de passar per començar a enregistrar i visualitzar en els reproductors el vehicle detectat.

```
295     %Valor minim de fotogrames que s'ha vist el vehicle perquè sigui
296     %considerat als resultats.
297 -   minVisibleCount = 6;
```

Figura A.19: Línia de codi on es troba el minVisibleCount.

A.2 Codi del programa de detecció i seguiment de vehicles

```
% Programa de detecció de vehicles creat i desenvolupat per Josep Oriol
% Díaz Hernàndez com a projecte de final de màster en enginyeria
% industrial a l'ETSEIB. Basat en Motion-Based Multiple Object Tracking.

% Per executar el programa escriure a la consola de comandes de MATLAB
% T = programaPrincipal('C:\Users\Usuari\Desktop\FitxerVideo.avi')
% Introdueix la ruta del vídeo a analitzar entre cometes.

function T = programaPrincipal(fitxerVideo)
% Crea els objectes necessaris per llegir el vídeo, detecta els objectes
% en moviment i mostra els resultats.
obj = configuraObjectesDelSistema(fitxerVideo);
% Inicia la estructura buida T de resultats que després empra el programa
% de tractament de dades.
T = struct();
fons = []; % Inicia la variable global del fons.
tracks = iniciaTracks(); % Inicia l'estructura buida de tracks.
proximaId = 1; % ID del pròxim track.
count = 0; % Contador del numero de frames.

% Detecció dels objectes en moviment i seguiment durant els vídeo frames.
while ~isDone(obj.reader)
    % Part del programa dedicat al tractament d'imatge.
    detectorFons(2000); % Introdueix el número de frames per detectar ...
        el fons.
    [frame, frameSencer] = llegeixFrame();
    [area, centroides, bboxes, mask] = detectorVehicles(frame);

    % Part que s'encarrega de fer el seguiment dels objectes detectats.
    prediuNovesLocalitzacionsFutures();
    [assignacions, tracksAmbVehiclesInvisibles, ...
        novesDeteccionsSenseTrack] = ...
         assignaDeteccionsAlsTracks();
    actualitzaVehiclesVisiblesAmbTrack();
    actualitzaVehiclesInvisiblesAmbTrack();
    borraTracksPerduts();
    creaNousTracks();

    % Mostra els resultats en temps real.
    Resultats();
    count = count + 1;
end

 function obj = configuraObjectesDelSistema(video)
    % Inicialització
    % Crea un lector de fitxers de vídeo.
    obj.reader = vision.VideoFileReader(video);
    obj.video = VideoReader(video);

    % Pren una captura del fitxer de vídeo.
    imatgeTransit = step(obj.reader);
```

```
    figure, imshow(imatgeTransit);

    % Calibratge de la distancia.
    set(gcf,'name','Calibratge de la distància real','numbertitle','off')
    message = sprintf('Ara es realitzarà la calibració de la imatge.');
    questdlg(message, 'Eina de calibratge', 'OK', 'OK');
    obj.calibratge = Calibrate();
    close

    % Selecciona l'àrea de treball.
    figure, imshow(imatgeTransit);
    title('Selecció de la zona de treball');
    set(gcf,'name','Imatge','numbertitle','off')
    message2 = sprintf('Ara es realitzarà la selecció de la zona de ...
       treball.\nSelecciona primer el punt superior esquerra i ...
       posteriorment el punt inferior dret.');
    questdlg(message2, 'Eina de selecció de la zona de treball', 'OK', ...
       'OK');
    [obj.x1, obj.x2, obj.y1, obj.y2] = Area(imatgeTransit);
    close

    % Crea dos reproductors de vídeo, un per mostrar el vídeo original ...
       i un
    % altre per mostrar la mask.
    obj.videoPlayer = vision.VideoPlayer('Position', [20, 200, 700, 400]);
    obj.maskPlayer = vision.VideoPlayer('Position', [740, 200, 700, 400]);

    % Configuració del blob analyzer.
    obj.blobAnalyser = vision.BlobAnalysis('BoundingBoxOutputPort', ...
       true, ...
            'AreaOutputPort', true, 'CentroidOutputPort', true, ...
            'MinimumBlobArea', 5000, 'MaximumBlobArea', 45000);
end

function tracks = iniciaTracks()
% Crea una estructura buida de tracks.
    tracks = struct(...
        'id', {}, ...
        'bbox', {}, ...
        'centroid', {}, ...
        'filtreKalman', {}, ...
        'edat', {}, ...
        'totalVisibleCount', {}, ...
        'consecutiveInvisibleCount', {});
end

function [frame, frameSencer] = llegeixFrame()
    % Llegeix un frame del vídeo.
    frameSencer = obj.reader.step() * 255;
    Retall_x1 = obj.x1;
    Retall_y1 = obj.y1;
    Retall_x2 = obj.x2;
    Retall_y2 = obj.y2;
    frame = frameSencer(Retall_y1:Retall_y2,Retall_x1:Retall_x2,:);
end
```

```
function detectorFons(ttfons)
    % Primer mira si hi ha suficients frames al vídeo.
    if ttfons > obj.video.frameRate*obj.video.Duration
       ttfons = obj.video.frameRate*obj.video.Duration;
    end
    % Calcul del fons segons el frame en que es troba.
    if count >= ttfons && mod(count,ttfons) == 0
        vectorDeModa = [];
        fons = [];
        if count+ttfons > (obj.video.duration*obj.video.frameRate)
            comptadorFrames = ...
               round(obj.video.duration*obj.video.frameRate,0);
        else
            comptadorFrames = count+ttfons;
        end
        for j = count:15:comptadorFrames
            frameDeModa = read(obj.video,j);
            vectorDeModa = cat(4,vectorDeModa,frameDeModa);
        end
        for i = obj.x1:obj.x2
            CalcFons = mode(vectorDeModa(obj.y1:obj.y2,i,:,:),4);
            fons = horzcat(fons,CalcFons);
        end
    elseif count == 0
    vectorDeModa = [];
    fons = [];
        for j = 1:15:ttfons
            frameDeModa = read(obj.video,j);
            vectorDeModa = cat(4,vectorDeModa,frameDeModa);
        end
        for i = obj.x1:obj.x2
            CalcFons = mode(vectorDeModa(obj.y1:obj.y2,i,:,:),4);
            fons = horzcat(fons,CalcFons);
        end
    end
end

function [area, centroides, bboxes, mask] = detectorVehicles(frame)
    %Obtenció de la mask.
    fons = single(fons);
    mask = frame - fons;
    mask = abs(mask);
    mask = uint8(mask);
    mask = imbinarize(rgb2gray(mask),0.1);

    % Operacions morfològiques.
    % Obre la imatge, la erosiona i dilata.
    mask = imopen(mask,strel('disk',3));
    % Tanca la imatge i emplena els espais buits entre objectes.
    mask = imclose(mask, strel('rectangle', [15, 15]));
    mask = imfill(mask, 'holes');  % Emplena els buits dins dels objectes

    % Realitza el blob analysis. mask ha der ser logical.
    [area, centroides, bboxes] = obj.blobAnalyser.step(mask);
```

```
end

function prediuNovesLocalitzacionsFutures()
   for i = 1:length(tracks)
       bbox = tracks(i).bbox;

       % Prediu nous tracks. El camí que fan els cotxes.
       % La funció predict preveu la resposta temporal de la dada que
       % s'introdueix.
       CentroidePredit = predict(tracks(i).filtreKalman);

       % El predict troba el centroide amb origen blob anterior i
       % s'ha de centrar el nou blob.
       % (3:4) Agafa l'amplada i altura, bbox = [x y width height].
       CentroidePredit = int32(CentroidePredit) - bbox(3:4) / 2;

       % Concatena les coordenades x y predites amb les d'amplada i
       % altura que es mantenen.
       tracks(i).bbox = [CentroidePredit, bbox(3:4)];
       pC = tracks(i).bbox(3:4);
       tracks(i).centroid = CentroidePredit + pC/2;
   end
end

function [assignacions, tracksAmbVehiclesInvisibles, ...
   novesDeteccionsSenseTrack] = ...
           assignaDeteccionsAlsTracks()
   numTracks = length(tracks);
   numDeteccions = size(centroides, 1);

   % Calcula el cost d'assignació de cada detecció a cada track.
   cost = zeros(numTracks, numDeteccions);
   for i = 1:numTracks
       cost(i, :) = distance(tracks(i).filtreKalman, centroides);
   end

   % Resol el problema d'assignació. assignDetectionsToTracks funció
   % matlab.
   costDeNoAssignacio = 19;
   [assignacions, tracksAmbVehiclesInvisibles, ...
      novesDeteccionsSenseTrack] = ...
       assignDetectionsToTracks(cost, costDeNoAssignacio);
end

function actualitzaVehiclesVisiblesAmbTrack()
   numTracksAssignats = size(assignacions, 1);
   for i = 1:numTracksAssignats
       IndexsTracks = assignacions(i, 1);
       IndexDeteccions = assignacions(i, 2);
       centroide = centroides(IndexDeteccions, :);
       bbox = bboxes(IndexDeteccions, :);

       % Actualitza la localitació amb la nova detecció.
       correct(tracks(IndexsTracks).filtreKalman, centroide);
```

```
        % Canvia la capsa predita per la detectada i els centroides.
        tracks(IndexsTracks).bbox = bbox;
        tracks(IndexsTracks).centroid = centroide;

        % Actualitza l'edat del track.
        tracks(IndexsTracks).edat = tracks(IndexsTracks).edat + 1;

        % Actualitza la visibilitat.
        tracks(IndexsTracks).totalVisibleCount = ...
        tracks(IndexsTracks).totalVisibleCount + 1;
        tracks(IndexsTracks).consecutiveInvisibleCount = 0;
    end
end

function actualitzaVehiclesInvisiblesAmbTrack()
    for i = 1:length(tracksAmbVehiclesInvisibles)
        ind = tracksAmbVehiclesInvisibles(i);
        tracks(ind).edat = tracks(ind).edat + 1;
        tracks(ind).consecutiveInvisibleCount = ...
        tracks(ind).consecutiveInvisibleCount + 1;
    end
end

function borraTracksPerduts()
    if isempty(tracks)
        return;
    end

    % Valor que borra els tracks segons el numero de fotogrames que no
    % s'ha vist o segons la edat mínima.
    invisibleForTooLong = 10;
    edatMinima = 8;

    % Calcula la visibilitat en funció de les vegades que s'ha vist i
    % l'edat.
    edats = [tracks(:).edat];
    totalVisibleCounts = [tracks(:).totalVisibleCount];
    visibilitat = totalVisibleCounts ./ edats;

    % Busca els índex dels tracks perduts. Si la edat es menor que la
    % mínima i no s'ha vist en 1/3 de la seva edat, al palco.
    IndexsPerduts = (edats < edatMinima & visibilitat < 1/3) | ...
    [tracks(:).consecutiveInvisibleCount] >= invisibleForTooLong;

    % Borra els tracks perduts.
    tracks = tracks(~IndexsPerduts);
end

function creaNousTracks()
    centroides = centroides(novesDeteccionsSenseTrack, :);
    bboxes = bboxes(novesDeteccionsSenseTrack, :);
    for i = 1:size(centroides, 1)
        centroide = centroides(i,:);
        bbox = bboxes(i, :);
```

```
        % Crea un filtre de Kalman.
        filtreKalman = configureKalmanFilter('ConstantVelocity', ...
            centroide, [200, 50], [100, 25], 100);

        % Crea un nou track.
        nouTrack = struct(...
            'id', proximaId, ...
            'bbox', bbox, ...
            'centroid', centroide, ...
            'filtreKalman', filtreKalman, ...
            'edat', 1, ...
            'totalVisibleCount', 1, ...
            'consecutiveInvisibleCount', 0);

        % Afegeix a l'array de tracks.
        tracks(end + 1) = nouTrack;

        % Incrementa en 1 la seguent Id.
        proximaId = proximaId + 1;
    end
end

function Resultats()
    % Converteix el frame i la mask a uint8 RGB.
    frame = uint8(frameSencer);
    frame = insertShape(frame, 'Rectangle', ...
    [obj.x1 obj.y1 obj.x2-obj.x1 obj.y2-obj.y1], ...
        'Color', 'white','LineWidth',3);

    % Repeteix la matriu mask binaria 1 vegada en 1 columna per 3
    % vegades (3 colors) i els passa a color blanc. Cosa del format.
    mask = uint8(repmat(mask, [1 1 3])) * 255;

    %Valor mínim de fotogrames que s'ha vist el vehicle perquè sigui
    %considerat als resultats.
    minVisibleCount = 6;
    if ~isempty(tracks)

        % Filtra de tots els tracks, es queda amb els que superen el
        % minVisibleCount.
        IndexTracksPossibles = ...
        [tracks(:).totalVisibleCount] > minVisibleCount;
        TracksPossibles = tracks(IndexTracksPossibles);

        % Mostra els objectes. Si un objectes no s'ha detectat en el
        % frame catual, mostra la seva capsa predita.
        if ~isempty(TracksPossibles)

        % Obté les capses i centroides.
        bboxes = cat(1, TracksPossibles.bbox);
        centroides = cat(1, TracksPossibles.centroid);
        centroides(:,3) = 15;
        centroidsReals = obj.calibratge.distancePerPixel * ...
            centroides(:,1:2);
        punt = bboxes(:,1:2);
```

```
            puntReal =  obj.calibratge.distancePerPixel * punt;
            bbox1 = [bboxes(:,1)+obj.x1, bboxes(:,2)+obj.y1, bboxes(:,3), ...
                bboxes(:,4)];
            centroid1 = [centroides(:,1)+obj.x1, centroides(:,2)+obj.y1, ...
                centroides(:,3)];

            % Obté les ids.
            ids = int32([TracksPossibles(:).id]);

            % Afegeix una etiqueta als objectes predits.
            labels = cellstr(int2str(ids'));
            IndexTracksPredits = ...
                [TracksPossibles(:).consecutiveInvisibleCount] > 0;
            esPredit = cell(size(labels));
            esPredit(IndexTracksPredits) = {' predicted'};
            labels = strcat(labels, esPredit);

            % Mostra els resultats a la imatge sencera.
            frame = insertObjectAnnotation(frame, 'rectangle', ...
                bbox1, labels);
            frame = insertObjectAnnotation(frame, 'Circle', centroid1, ...
                labels);

            % Mostra els resultats a la mask.
            mask = insertObjectAnnotation(mask, 'rectangle', ...
                bboxes, labels);
            mask = insertObjectAnnotation(mask, 'Circle', centroides, ...
                labels);

            % Escriu els resultats
            T(end).bboxes = bboxes;
            T(end).id = labels;
            T(end).count = count;
            T(end).centroid = centroides;
            T(end).centroidsReals = centroidsReals;
            T(end).punt = punt;
            T(end).puntReal = puntReal;
            T(end+1) = T(end);
            end
        end

        % Mostra al reproductor el frame i la mask.
        obj.maskPlayer.step(mask);
        obj.videoPlayer.step(frame);
    end

% Funcions auxiliars per la selecció de l'àrea de treball i per la
% realització del calibratge de la distància.

    function calibra = Calibrate()
        global liniaDibuixada;
        try
            instructions = sprintf('Clic esquerra per ancorar el primer ...
                punt a la imatge.\nClic dret o doble clic esquerra per ...
                ancorar el segon punt de la linia.\n\nDesprés es podrà ...
```

```
            introduir la mesura real.');
        msgboxw(instructions);

        [cx, cy, rgbValues, xi, yi] = improfile(1000);
        % rgbValues is 1000x1x3.  Call Squeeze to get rid of the ...
            singleton dimension and make it 1000x3.
        rgbValues = squeeze(rgbValues);
        distanceInPixels = sqrt( (xi(2)-xi(1)).^2 + (yi(2)-yi(1)).^2);
        if length(xi) < 2
            return;
        end
        % Dibuixa la linia.
        hold on;
        liniaDibuixada = plot(xi, yi, 'y', 'LineWidth', 1);

        % Demana a l'usuari la distància real.
        userPrompt = {'Introdueix les unitats reals (per defecte: ...
            cm):','Introdueix la distancia en aquestes unitats'};
        dialogTitle = 'Eina de calibratge';
        nombreDeLinies = 1;
        def = {'cm', '200'};
        resposta = inputdlg(userPrompt, dialogTitle, nombreDeLinies, def);
        if isempty(resposta)
            return;
        end
        calibra.units = resposta{1};
        calibra.distanceInPixels = distanceInPixels;
        calibra.distanceInUnits = str2double(resposta{2});
        calibra.distancePerPixel = calibra.distanceInUnits / ...
            distanceInPixels;

    return;
    end
end

function msgboxw(message)
    uiwait(msgbox(message));
end

function [x1, x2, y1, y2] = Area(imatge)
    % Selecciona l'area de treball.
    grandaria = size(imatge);
    altura = grandaria(1);
    amplada = grandaria(2);

    [x,y]=ginput(2);

    x1 = round(x(1));
    x2 = round(x(2));
    y1 = round(y(1));
    y2 = round(y(2));

    if x1 < 0
        x1 = 1;
    end
```

```
    if x2 < 0
        x2 = 1;
    end

    if x1 > amplada
        x1 = amplada;
    end

    if x2 > amplada
        x2 = amplada;
    end

    if y1 < 0
        y1 = 1;
    end

    if y2 < 0
        y2 = 1;
    end

    if y1 > altura
        y1 = altura;
    end

    if y2 > altura
        y2 = altura;
    end
 end

end
```

A.3 Codi del programa de càlcul de velocitats i acceleracions de vehicles

```
% Programa de tractament de dades obtingudes a partir del
% programaPrincipal.m. Creat i desenvolupat per Josep Oriol
% Díaz Hernàndez com a projecte de final de màster en enginyeria
% industrial a l'ETSEIB.

% Per utilitzar el programa es necessita haver executat primer el
% programaPrincipal.m i emmagatzemar les dades en la variable T.
% Si no executar-lo primer a la consola de MATLAB amb aquesta comanda:
% T = programaPrincipal('C:\Users\Usuari\Desktop\FitxerVideo.avi')

clc

[a, b] = size(T); % Estructura a llegir procedent del programa de detecció.
Compt_V = 1; % Comptador referent al càlcul de la velocitat.
RRR = struct(); %Estructura de dades final.
Gran = 0; % Emmagatzema la mida del vector de propietats de cada vehicle.
Compt_A = 1; % Comptador referent al càlcul de l'acceleració.
FPS = 30; % Frames per segon del video.
ams = 1; % Comptador que serveix per indexar a la taula RRR les ...
    velocitats Vfit del polinomi.

for i = 1:b
[Fil, Col] = size(T(i).id);
if Fil > 1
    for k = 1:Fil
    Id(Compt_V,:) = regexprep(T(i).id(k),'[^\w'']','');
    Punts(Compt_V,:) = T(i).puntReal(k,:);
    Count(Compt_V,:) = T(i).count;
    Compt_V = Compt_V + 1;
    end
else
    Id(Compt_V,:) = regexprep(T(i).id,'[^\w'']','');
    Count(Compt_V,:) = T(i).count;
    Punts(Compt_V,:) = T(i).puntReal(:);
    Compt_V = Compt_V + 1;
end
end

Temps = Count/FPS;

% Busca el darrer cotxe detectat i la seva Id situada al final de tot.
Idmaxima = cell2mat(Id(length(Id)));
Idlength = strfind(Idmaxima,'predicted');
if Idlength > 0
    Idmaxima = Idmaxima(1:Idlength-1);
    Idmaxima = str2double(Idmaxima);
else
Idmaxima = str2double(Idmaxima);
end
```

```
% Bucle de tractament de les dades. Càlcul de V i A.
for i = 1:Idmaxima
id = i;
idtofind = int2str(id);
predicted = 'predicted';
stringtofind = strcat(idtofind,predicted);

index1 = find(strcmp(Id, idtofind));
index2 = find(strcmp(Id, stringtofind));

% index es la variable que ordena les propietats de cada vehicle segons
% aparició.

% Si es vol els predicted.
% index = sort(vertcat(index1,index2));

% Si no es volen els predicted.
index = sort(vertcat(index1));

Gran = numel(index);

% Reinicia els vectors a 0 de dades de velocitat i temps emprades per fer
% l'aproximació polinòmica.
Temps_polyfit = [];
Velocitat_polyfit = [];

for Compt_V = 1:Gran
    RRR(end).id = id;
    RRR(end).temps = Temps(index(Compt_V));
    Temps_polyfit(Compt_V,:) = Temps(index(Compt_V));

    if Compt_V < Gran
        % Càlcul de la velocitat fent la diferència de distàncies entre el
        % temps.
        Xx =  double((Punts(index(Compt_V+1),1) - ...
            Punts(index(Compt_V),1)))^2;
        Xy =  double((Punts(index(Compt_V+1),2) - ...
            Punts(index(Compt_V),2)))^2;
        X = sqrt(Xx + Xy);
        t = Temps(index(Compt_V+1)) - Temps(index(Compt_V));
        V = X / t;
        RRR(end).V = V;
        Velocitat_polyfit(Compt_V,:) = double(V);
        RRR(end).X = Punts(index(Compt_V),:);

        % Càlcul de l'acceleració
        if Compt_V >= 2
            RRR(end-1).A = (RRR(Compt_A).V - (RRR(Compt_A-1).V))/t;
        end

    else
        % Borra els dos punts buits d'acceleració i velocitat.
        RRR(end-1) = [];
        RRR(end) = [];
```

```
        Compt_A = Compt_A - 2;

        % Assegura que hi hagi suficients punts de temps i velocitat perquè
        % no reventi el programa.
        if  numel(Temps_polyfit) >= 2
            Temps_polyfit(end) = [];
            Temps_polyfit(end) = [];
        else
           Temps_polyfit(end) = 1000;
        end

        if  numel(Velocitat_polyfit) >= 1
            Velocitat_polyfit(end) = [];
        else
            Velocitat_polyfit = 1000;
        end

        % Part que realitza l'aproximació polinòmica segons el grau i
        % l'afegeix a la taula de resultats RRR.
        gp = 3; % Grau del polinomi.
        Vfit = polyfit(Temps_polyfit,Velocitat_polyfit,gp);

        % Càlcul del polinomi, si es vol. El programa no el mostra pero
        % fent debug es pot saber de cada vehicle.
        syms x
        for i = 1:(gp+1)
            VF(i) = Vfit(i)*x^((gp+1)-i);
        end
            v(x) = sum(VF);
            accel(x) = diff(v(x),x);

        for Compt_Vfit = 1:(Compt_A-ams+1)
            RRR(ams).Vfit = polyval(Vfit,Temps_polyfit(Compt_Vfit));
            RRR(ams).Afit = double(accel(Temps_polyfit(Compt_Vfit)));
            ams = ams + 1;
        end
    end
    Compt_A = Compt_A + 1;
    RRR(end+1).id = 'Emplena';

end
end

RRR(end) = [];
%writetable(struct2table(RRR), 'RRR.txt', 'Delimiter', 'tab')

% Programa que s'encarrega d'eliminar els 0 deguts als vehicles que just
% surten de l'àrea de treball a l'hora d'agafar el vèrtex superior
% esquerra.

[c, d] = size(RRR);
y = 1;
for z = 1:d
    VVV(y,:) = RRR(z).V;
    IDD(y,:) = RRR(z).id;
```

```
    AAA(y,:) = RRR(z).A;
    if isempty(RRR(z).Vfit) | isempty(RRR(z).Afit)
        VVVFit(y,:) = RRR(z+1).Vfit;
        AAAFit(y,:) = RRR(z+1).Afit;
    else
        VVVFit(y,:) = RRR(z).Vfit;
        AAAFit(y,:) = RRR(z).Afit;
    end
    TTT(y,:) = RRR(z).temps;
    y = y + 1;
end

% Aquí es podria posar un filtre sobre les velocitats. Per defecte
% elimina les velocitats que són 0 degudes al fenomen explicat. Per no
% esborrar les aturades s'hauria de modificar i per exemple posar
% find(VVV>=250) ara buscaria els índex de les velocitats superiors a
% 250 cm/s.

index3 = find(VVV);
index3grandaria = numel(index3);
clc

for g = 1:index3grandaria
    Resultats(g,:).Id = IDD(index3(g));
    Resultats(g,:).Temps = TTT(index3(g));
    Resultats(g,:).V = VVV(index3(g))*3.6/100; % Per passar de cm/s a km/h
    Resultats(g,:).A = AAA(index3(g))/100; % Per passar de cm/s2 a m/s2
    Resultats(g,:).VFit = VVVFit(index3(g))*3.6/100;
    Resultats(g,:).AFit = AAAFit(index3(g))/100;
end

writetable(struct2table(Resultats), 'Resultats.txt', 'Delimiter', 'tab')

% Part que s'encarrega de fer una fulla excel amb els resultats. Primer
% desglossa els vehicles i els introdueix per separat en una fulla ...
   excel amb
% el número d'acord a la id del vehicle. Finalment crea una darrer fulla
% "Resultats" on inclou tots els punts de tots els vehicles en una única
% llista.

% Pot ser que alguna fulla quedi buida perquè no hi havia suficients punts
% del vehicle per calcular les acceleracions o velocitats.

for g = 1:index3grandaria
    IdExcel(g,:) = Resultats(g).Id;
    TempsExcel(g,:) = Resultats(g).Temps;
    VExcel(g,:) = Resultats(g).V;
    AExcel(g,:) = Resultats(g).A;
    VFitExcel(g,:) = Resultats(g).VFit;
    AFitExcel(g,:) = Resultats(g).AFit;
end

sheet = 1;
i = 1;
filename = 'Resultats.xlsx';
```

```
Cap = {'Id', 'Temps [s]', 'V [km/h]', 'A [m/s2]', 'VFit [km/h]', 'AFit ...
    [m/s2]'};

for id = 1:Idmaxima
    indexExcel = find(IdExcel==id);
    granExcel = size(indexExcel);
    if ~granExcel(1) == 0
    for i = 1:granExcel
      IdE(i,:) = IdExcel(indexExcel(i));
      TempsE(i,:) = TempsExcel(indexExcel(i));
      VE(i,:) = VExcel(indexExcel(i));
      AE(i,:) = AExcel(indexExcel(i));
      VFitE(i,:) = VFitExcel(indexExcel(i));
      AFitE(i,:) = AFitExcel(indexExcel(i));
      i = i + 1;
    end
    filename = 'Resultats.xlsx';
    Valors = [IdE TempsE VE AE VFitE AFitE];
        xlswrite(filename,Cap,sheet,'A1')
        xlswrite(filename,Valors,sheet,'A2')
    end
    sheet = sheet + 1;
    IdE = [];
    TempsE = [];
    VE = [];
    AE = [];
    VFitE = [];
    AFitE = [];
end

sheet = 'Resultats';
Valors = [IdExcel TempsExcel VExcel AExcel VFitExcel AFitExcel];
xlswrite(filename,Cap,sheet,'A1')
xlswrite(filename,Valors,sheet,'A2')

clc
clear a granExcel VE AE VFitE AFitE IdE indexExcel TempsE Valors AAA ...
    AAAFit accel ams b c Col Compt_A Compt_V Compt_Vfit Count d Fil FPS ...
    g gp Gran i id Id IDD Idlength Idmaxima idtofind index index1 index2 ...
    index3 index3grandaria k predicted Punts RRR stringtofind t Temps ...
    Temps_polyfit TTT v V Velocitat_polyfit VF Vfit VVV VVVFit x X Xx Xy ...
    y z A AExcel IdExcel AFitExcel Cap filename EdExcel sheet TempsExcel ...
    VExcel VFitExcel
```