Projekts:

Valsts pētījumu programmas SOPHIS Projekts Nr.4. „Tehnoloģijas drošai un uzticamai gudrajai pilsētai”

Ultraplatjoslas radaru izmantošana automašīnu skaitīšanai

Tehnoloģijas lietošanas apraksts

Autori:

Rūdolfs Cīrulis, Ģirts Kaģis, Eduards Lobanovs

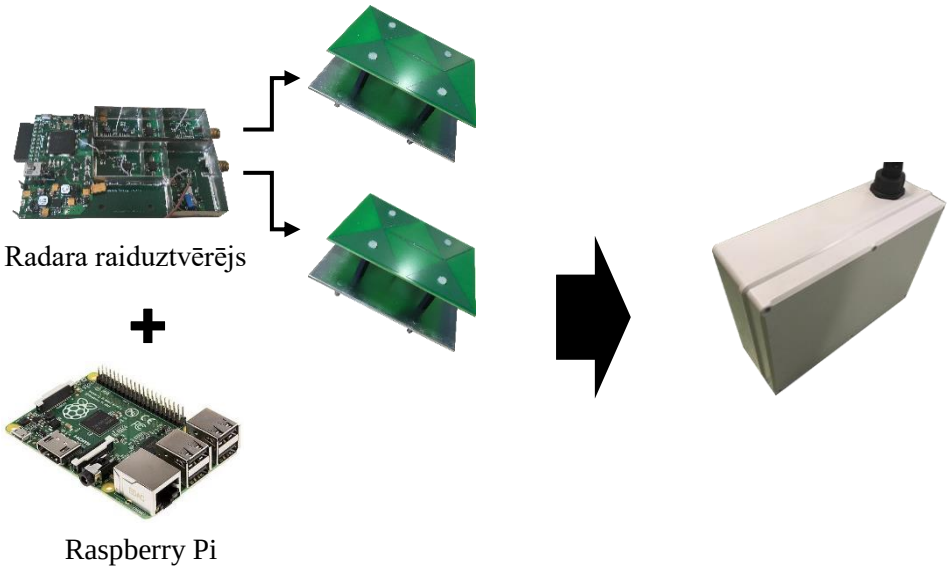
2017.g.

Metodika ultraplatjoslas radaru izmantošanai automašīnu skaitīšanai

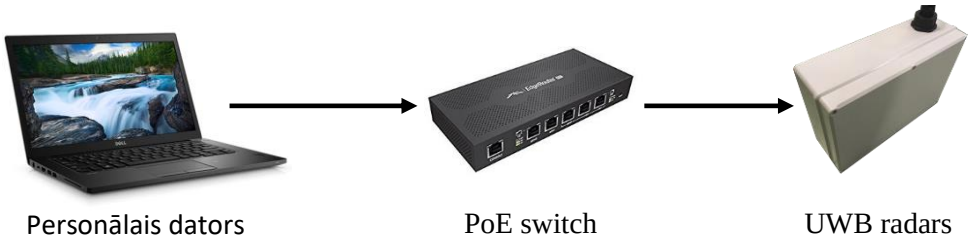
Problēma: Precīza automašīnu skaitīšana un skaita klasificēšana pa joslām.

Mērķis: Izstrādāt metodiku, kas sniedz iespēju personai bez specifiskas tehniskās kvalifikācijas, uzstādīt ultraplatjoslas radaru un tā programmas nodrošinājumu automašīnu skaitīšanai.

Apraksts: Sistēma sastāv no UWB radara, tīkla komutatora (*switch*) ar PoE iespēju un personālā datora, skat. 2.attēlu. UWB radara darbības attālums ir $12\,m$. Iekārtā tiek izmantots EDI (Elektronikas un Datorzinātņu institūtā) izstrādātais ultraplatjoslas radara prototips 1. att. Izmantotā programmatūra (*GUI_v03.exe*) veic radara signāla attēlošanu, filtrāciju un iekļauj algoritmus automašīnu skaitīšanai līdz 2 joslām (5.att.). Elektriskās barošanas un komunikācijas interfeiss ir PoE (*Power over Ethernet*).



1. att. UWB radara iekārta.



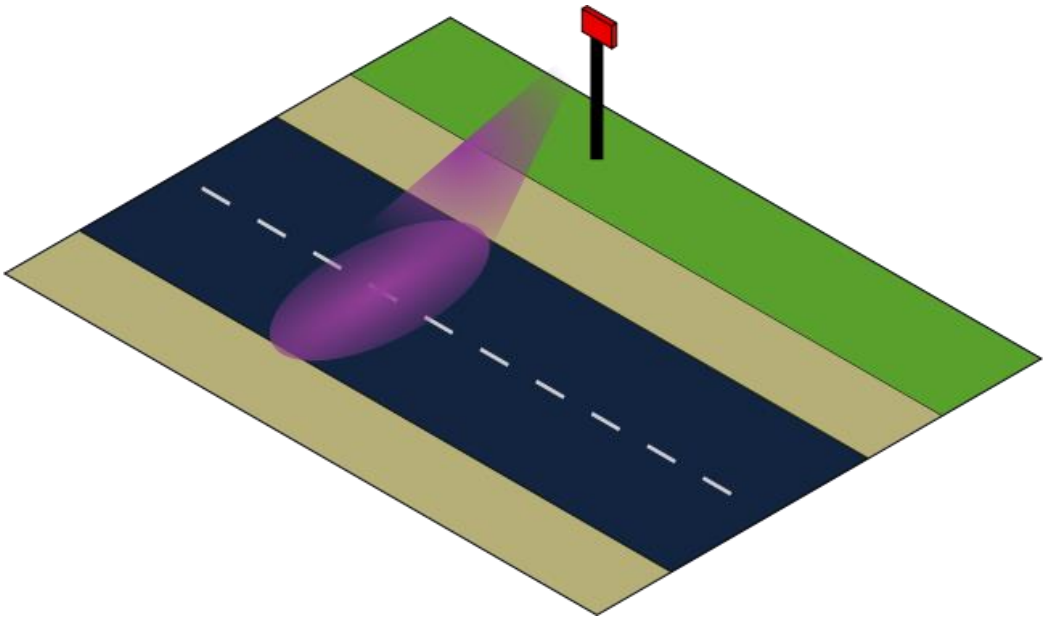
2. att. Sistēmas pieslēgšanas konfigurācija.

Uzdevumi:

Precīzai automašīnu skaitīšanai ultraplatjoslas radaru nepieciešams uzstādīt pareizā novietojumā attiecībā pret brauktuvi un precīzi atzīmēt signālā katras brauktuves joslas robežas. Grafiska soļu reprezentācija redzama 3. attēlā.

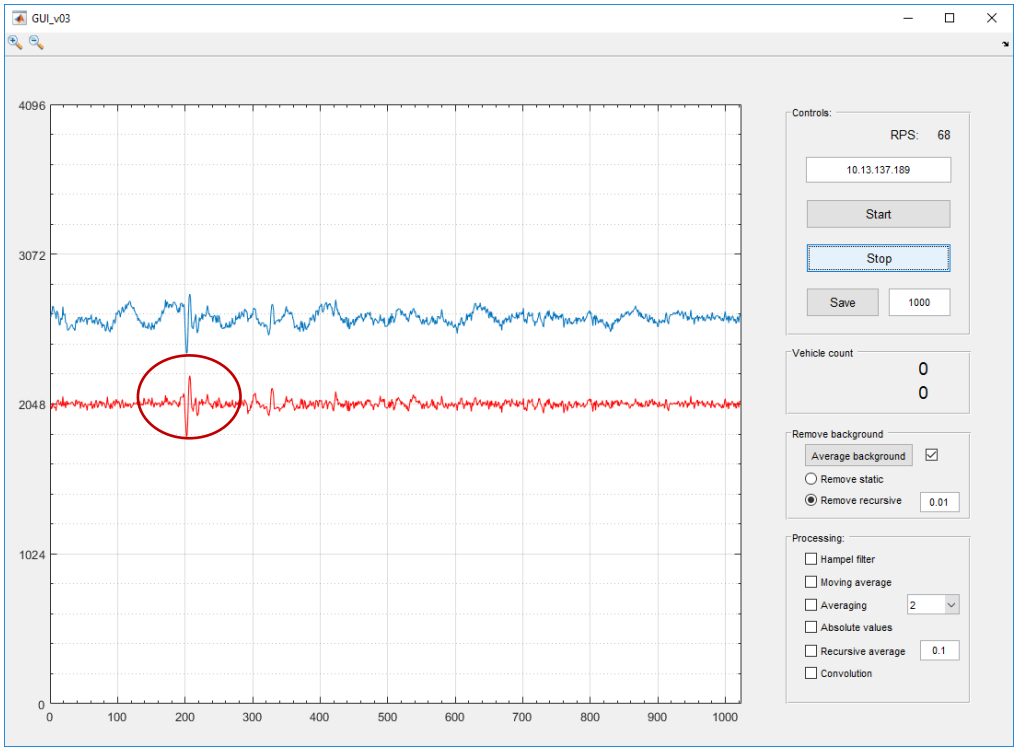
3. att. Radara sistēmas uzstādīšanas gaita auto skaitīšanai.

1. Radara uzstādīšana:
   Ultraplatjoslas radara iekārtu nepieciešams uzstādīt tā, lai signāli tiktu raidīti un uztverti perpendikulāri brauktuvei, uz kuras nepieciešams veikt automašīnu skaitīšanu. Jāievēro, lai iekārtas antenas virziena diagramma ir notēmēta uz brauktuves vidus daļu.



4. att. Radara sistēmas pozīcijas uzstādīšana.

1.1. Radara darbības zonas atrašana:
   Lai sekmīgi atrastu radara iekārtas darbības zonu, nepieciešams veikt mērījumu ar imitētu objektu, izmantojot programmu *GUI_v03.exe*. Nepieciešams palaist programmu *GUI_v03.exe*, ievadīt radaram piešķirto IP adresi un nospiest *Start* spiedpogu. Auto skaitīšanas gadījumā iespējams novietot automašīnu intereses zonā un novērtēt labāko radara pozīciju pret interesējošo zonu pēc iegūtā atstarotā signāla amplitūdas, kura jānovērtē no iegūtās radara oscilogrammas (5.att). Mašīnu iespējams aizstāt arī ar metālisku plāksni apmēram $50\ x\ 50cm$ izmērā. Programma *GUI_v03.exe* nodrošina atstarotā signāla nolašu attēlošanu. Piemērs ar atstaroto signālu, kas noregulēts precīzi uz automašīnu, kas atrodas pirmajā joslā, redzams 5. att. Radara pozīcija jānoregulē tā, lai atstarojuma amplitūda būtu pēc iespējas lielāka.



5. att. Atstarojums no automašīnas, kas novietota pirmajā joslā.

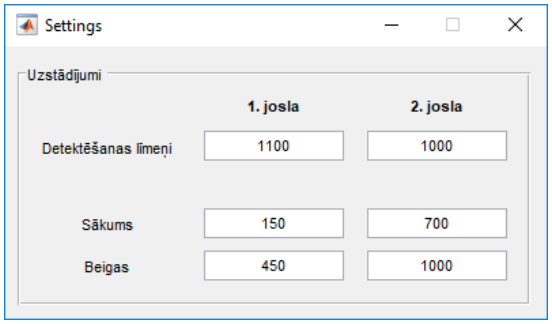
2. Brauktuves joslas robežas iestādīšana, izmantojot programmas *GUI_v03.exe* izvēlnes funkciju *Files -> Settings*

2.1. Automašīnu skaitīšanai neatkarīgi no joslām:

Lai skaitītu auto un neizdalītu transporta blīvumu starp joslām, nepieciešams iestatīt pirmās joslas robežas visā uztveršanas diapazonā (nolases 0 .. 1023), izmantojot programmatūras uzstādījumu paneli. Joslai jāiestata *Sākuma* un *Beigu* robežas (6. att..) nolašu vienībās. Lai nodrošinātu sistēmas precīzāku darbību un mazāku kļūdainas automašīnu pieskaitīšanas varbūtību, joslas robežas var iestatīt pēc atstarojuma no brauktuves sākuma daļas līdz tās beigu daļai. Tas samazinās iespēju, ka gājējs var tikt ieskaitīts kā automašīna.

2.2. Automašīnu skaitīšanai 2 joslās:

Divu joslu gadījumā nepieciešams iestatīt katras joslas nolašu vērtības atsevišķi. Piemērs ar joslu uzstādījumu vērtībām redzams 6. attēlā. Ieteicams starp atsevišķām joslām (1. joslas beigu vērtības un 2. joslas sākuma vērtības) atstāt vismaz 100 nolašu starpību. Tas izslēgs gadījumu, kad automašīna varētu tikt ieskaitīta abās joslās vienlaicīgi.



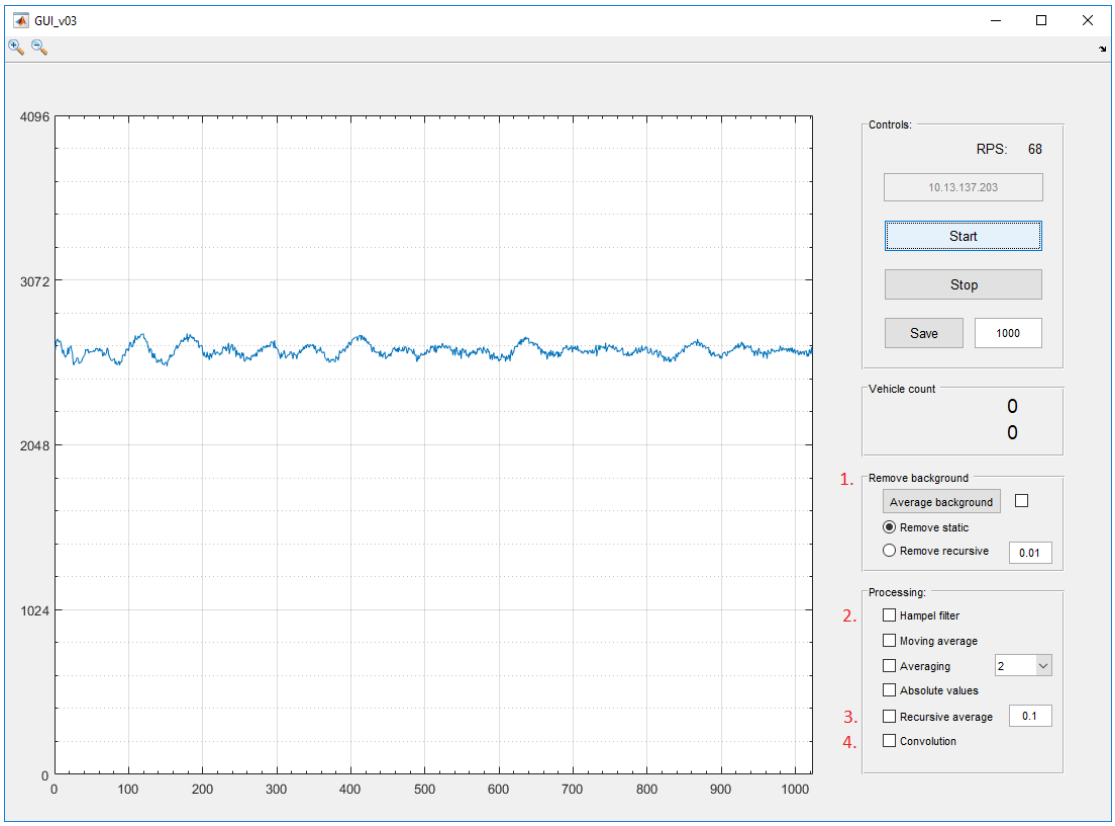
6. att. Programmatūras uzstādījumu panelis.

3. Papildus parametru iestatīšana:

   Automašīna pareizai detektēšanai nepieciešams novērtēt aptuveno signāla un trokšņa attiecību pēc radara iegūtā signāla. Ja signāla amplitūda nav vismaz *5* reizes lielāka par trokšņa amplitūdu, tad jāveic secīgas, zemāk aprakstītās darbības, līdz tiek iegūts vēlamais rezultāts.

   3.1. Fona signāla noņemšana:

   Radara iekārta iegūst atstarojumu no visiem objektiem, kas ir tās redzamības zonā. Tāpēc nepieciešams atņemt signālu, kurš ir nemainīgs (statisks). To iespējams panākt, programmas *GUI_v03.exe* ieslēdzot 1. bloka (7. att.) darbību ar izvēles rūtiņu (*checkbox*), kas atrodas blakus spiedpogai *Average background* sadaļā *Remove background*. Fona atņemšanai iespējamas 2 opcijas: noņemt statisku ierakstītu signālu (*Remove static*) vai fona signālu atjaunināt ar rekursīvu vidējošanu (*Remove recursive*). Tā kā laika apstākļu dēļ un dažādu ārēju faktoru dēļ fona signāls var mainīties laika gaitā, ieteicams izmantot rekursīvu fona ierakstīšanas un atņemšanas iespēju.



7. att. Papildus parametru izvēļu logs.

   3.2. Hampeļa filtrs:

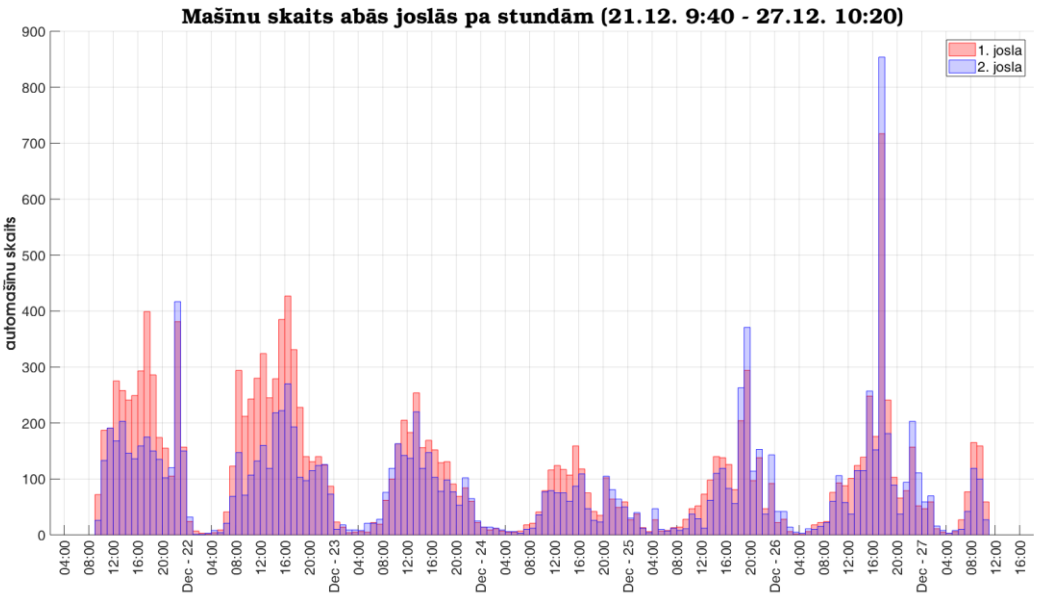
   Filtrs, kas noņem atsevišķu izlecošu vērtību pīķus. Ārēju radio signālu iespaidā radara ieejā var rasties signāla pīķi, kas nav saistīti ar atstarojumu no objektiem. Ieslēdzot izvēles rūtiņas 2. opciju (7. att.), tiks veikta šādu trokšņaina signāla filtrēšana.

3.3. Rekursīvā vidējošana:
Lai samazinātu trokšņa signāla amplitūdu, iespējams ieslēgt signāla vidējošanas iespēju - 3. opcija (7. att.). Tā kā tiek izmantota rekursīvā vidējošana, tad nepieciešams iestatīt vidējošanas koeficienta vērtību. Automašīnu detektēšanai, kas nekustas ātrāk par *60 km/h*, ieteicamā koeficienta vērtība ir *0,1*.

3.4. Konvolūcija ar ierakstītā signāla paraugu:
Ja augstāk minētie soļi nepalīdz iegūt vēlamo signāla/trokšņa attiecību, tad iespējams veikt arī signāla konvolūciju ar iebūvēto saglabāto atstarojuma signālu – 4. opcija.

4. Pārliedzamā sliekšņa vērtības iestatīšana:
Pēdējais solis automašīnu detektēšanai ir sliekšņa vērtības iestatīšana. Tā ir relatīva vērtība, kura jāpārsniedz pēc programmatūras iekšējo algoritmu izpildes, lai atstarojums tiktu ieskaitīta kā automašīna. Pēc noklusējuma šīs vērtības ir *1100* (1. joslai) un *1000* (2. joslai). Esošās vērtības iespējams palielināt, lai samazinātu kļūdainu automašīnas ieskaitīšanas iespēju, vai samazināt, ja automašīnas netiek ieskaitītas. Ieteicams vērtības mainīt maksimāli 600 vienību robežās ap noklusētajām.

Rezultāti: Skaitīšanas rezultāti tiek saglabāti atsevišķos ārējos *MATLAB* failos ar nosaukumiem *Timestamps1.mat* (1. josla) un *Timestamps2.mat* (2. josla). Rezultātu piemērs 8. att.



8. att. Piemērs rezultātu attēlošanai *MATLAB* vidē.

Programmatūras kods:

```
function varargout = GUI_v03(varargin)
% GUI_V03 MATLAB code for GUI_v03.fig
%      GUI_V03, by itself, creates a new GUI_V03 or raises the existing
%      singleton*.
%
%      H = GUI_V03 returns the handle to a new GUI_V03 or the handle to
%      the existing singleton*.
%
%      GUI_V03('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in GUI_V03.M with the given input arguments.
%
%      GUI_V03('Property','Value',...) creates a new GUI_V03 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before GUI_v03_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to GUI_v03_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help GUI_v03
% Last Modified by GUIDE v2.5 05-Jan-2018 11:37:00

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @GUI_v03_OpeningFcn, ...
                   'gui_OutputFcn',  @GUI_v03_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before GUI_v03 is made visible.
function GUI_v03_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to GUI_v03 (see VARARGIN)

set(handles.Start_Button,'UserData', 0);
handles.TimerVal = zeros(1,100);
handles.TimerVal(end) = tic;

handles.Background = zeros(1023,50);
handles.Background_mean = zeros(1023,1);
```

```
handles.Frame_smooth_matrix = zeros(1023,1);

handles.Average_matrix = zeros(1023,50);
axes(handles.A_Scan_axes);
hold on;
handles.A_Scan = plot(zeros(1023,1));
handles.Processed = plot(handles.Background_mean+2048, 'r');
hold off;
set(handles.Processed,'Visible','off');

handles.k = 0;

% Choose default command line output for GUI_v03
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% --- Outputs from this function are returned to the command line.
function varargout = GUI_v03_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in Start_Button.
function Start_Button_Callback(hObject, eventdata, handles)
% hObject    handle to Start_Button (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if (get(handles.Start_Button,'UserData') ~= 1)
%     try
        handles.k = 1;
        n = 1;

        %====== TCP Client
        t = tcpclient(get(handles.Edit_IP,'String'), 22000);
        d = double(read(t,1024,'uint16'));
        delay = find(~d);
        d = double(read(t,delay-1,'uint16'));
        d = double(read(t,1024,'uint16'));


        % A-scan plot
        axes(handles.A_Scan_axes);
        handles.A_Scan.YData = d(2:end);
        xlim([0, 1024]);
        yticks([0, 1024, 2048, 3072, 4096]);
        ylim([0, 4096]);
        grid on

        if (get(handles.Status_message,'String') ~= "")
            set(handles.Status_message,'String',"")
        end
```

```
set(handles.Edit_IP,'Enable','Off')

%===================
tic;

set(handles.Start_Button,'UserData', 1);
Save_matrix = [];
Save_counter = [];

%===================
%Variables for car detection

car_detected1 = false;
car_detected2 = false;

salsA = 150; %Pirmās joslas sākuma robeža
salbA = 450; %Pirmās joslas beigu robeža
salsB = 700; %Otrās joslas sākuma robeža
salbB = 1000; %Otrās joslas beigu robeža

winlen = 45;
winstep = 10;
winA = floor((salbA - salsA-winlen)/winstep);
winB = floor((salbB - salsB-winlen)/winstep);

veclen = 5;
sumlvecA = false([winA,veclen]);
sumlvecB = false([winB,veclen]);

levelA = 1100;
levelB = 950;

load('Fragment.mat');
fragment = fragment/sqrt(sum(fragment.^2));

VehicleCount1 = 0;
VehicleCount2 = 0;
set(handles.VC1,'String',num2str(VehicleCount1));
set(handles.VC2,'String',num2str(VehicleCount2));

ind_timer1 = 0;
ind_timer2 = 0;
%======== MAIN WHILE LOOP ========
%=================================
while(get(handles.Start_Button,'UserData') ~= 0)

    %Reading data and refresh A_Scan_axes
    d = double(read(t,1024,'uint16'));


    handles.A_Scan.YData = d(2:end);

    %Data save
    if get(handles.Save,'UserData') == 2
        if isnan(str2double(get(handles.Save_count,'String')))
            Save_counter = 100;
        else
```

```
                    Save_counter = str2double(get(handles.Save_count,'String'))-
1;
                end
                set(handles.Save,'UserData',1)
                Save_matrix = [];
            elseif get(handles.Save,'UserData') == 1
                Save_matrix = [Save_matrix; d];
                if Save_counter ~= 0
                    Save_counter = Save_counter - 1;
                else
                    save('Data.mat','Save_matrix');
                    set(handles.Save,'UserData',0);
                    set(handles.Status_message,'String','Save done!')
                end
            end

            %Determins RPS and displays it
            handles.TimerVal = circshift(handles.TimerVal,[0,-1]);
            handles.TimerVal(end) = toc;
            set(handles.RPS,'String',round(100/sum(handles.TimerVal)));
            tic;

            %Transpose data for further processing and remove mean value
            %Also removes first value (0)
            D = d(2:end)'-mean(d(2:end));

            %Acquiring Bacground
            if(get(handles.Average_bcg,'UserData') == 1)
                if n <= 50
                    handles.Background(:,n) = D;
                    n = n + 1;
                else
                    set(handles.Average_bcg,'UserData', 0)
                    n = 1;
                    set(handles.Background_Status,'String','Done!');
                    handles.Background_mean = mean(handles.Background,2);
                end
            end

            %==============================
            %Processing part
            if (get(handles.Remove_bcg,'Value'))
                if (get(handles.Remove_bcg_rec,'Value'))
                    koef = str2double(get(handles.Background_koef,'String'));
                    if isnan(koef)
                        handles.Background_mean = 0.9 * handles.Background_mean
+ 0.1 * D;
                    else
                        handles.Background_mean = (1-koef) *
handles.Background_mean + koef * D;
                    end
                end
                D = D - handles.Background_mean;
            end

            if (get(handles.Hampel,'Value'))
                D = hampel(D,3,2);
            end

            if (get(handles.Conv1_checkbox,'Value'))
```

```
                D = smooth(D,10);
            end

            if (get(handles.Averaging,'Value'))
                handles.Average_matrix = circshift(handles.Average_matrix,[0,-
1]);
                handles.Average_matrix(:,end) = D;
                content = get(handles.Average_numb,'String');
                value = get(handles.Average_numb,'Value');
                avg = str2double(content(value));
                D = mean(handles.Average_matrix(:,51-avg:end),2);
            end

            if (get(handles.Abs_val,'Value'))
                D = 10*abs(D);
            end

            if (get(handles.Recursive_avg,'Value'))
                koef = str2double(get(handles.Avg_koef,'String'));
                if isnan(koef)
                    koef = 0.3;
                end
                D = (1 - koef) * handles.Frame_smooth_matrix + koef * D;
                handles.Frame_smooth_matrix = D;
            end

            if (get(handles.convolution,'Value'))
                G = conv(D,fragment);
                D = G(length(fragment):end);
            end

            %Vehicle detection
            %==============================

            prev_car_det1 = car_detected1;
            prev_car_det2 = car_detected2;

            sumlvecA = circshift(sumlvecA,1,2);
            sumlvecB = circshift(sumlvecB,1,2);

            for m = 1:winA
                sumlvecA(m,1) = sum(abs(D(salsA+winstep*(m-1):salsA+winstep*(m-
1)+winlen))) > levelA;
            end

            for m = 1:winB
                sumlvecB(m,1) = sum(abs(D(salsB+winstep*(m-1):salsB+winstep*(m-
1)+winlen))) > levelB;
            end

            % Detects car if at least 2 zones are valid
            if (sum(sum(sumlvecA,2) == 5) > 0)  %Zone count
                 car_detected1 = true;
            elseif (sum(sum(sumlvecA,2) == 5) == 0)
                car_detected1 = false;
            end

            % Detects car if at least 2 zones are valid
            if (sum(sum(sumlvecB,2) == 5) > 0)  %Zone count
```

```
                car_detected2 = true;
            elseif (sum(sum(sumlvecB,2) == 5) == 0)
                car_detected2 = false;
            end

            if ((car_detected1 == true) && (prev_car_det1 == false))
                VehicleCount1 = VehicleCount1 + 1;
                set(handles.VC1,'String',num2str(VehicleCount1));
                set(handles.VC1,'BackgroundColor','green');
                ind_timer1 = 35;
                Timestamp(1)
            end

            if (ind_timer1 > 0)
                ind_timer1 = ind_timer1 - 1;
                if (ind_timer1 == 1)
                    set(handles.VC1,'BackgroundColor','default');
                end
            end

            if ((car_detected2 == true) && (prev_car_det2 == false))
                VehicleCount2 = VehicleCount2 + 1;
                set(handles.VC2,'String',num2str(VehicleCount2));
                set(handles.VC2,'BackgroundColor','green');
                ind_timer2 = 35;
                Timestamp(2)
            end

            if (ind_timer2 > 0)
                ind_timer2 = ind_timer2 - 1;
                if (ind_timer2 == 1)
                    set(handles.VC2,'BackgroundColor','default');
                end
            end

            %==============================
            %Visibility of processed data (Enables Red line)
            if(get(handles.Remove_bcg,'Value') ||...
                get(handles.Conv1_checkbox,'Value') ||...
                get(handles.Hampel,'Value') || ...
                get(handles.Averaging,'Value') || ...
                get(handles.Abs_val,'Value') || ...
                get(handles.Recursive_avg,'Value') || ...
                get(handles.convolution,'Value'))

                set(handles.Processed,'Visible','on');
                handles.Processed.YData = D*5 + 2048;
            else
                set(handles.Processed,'Visible','off');
            end

            %==============================
            guidata(hObject,handles);
        end
        clear t;

%     catch
%         set(handles.Status_message,'String','Cannot access as TCP Client!')
%     end
```

```
end

% --- Executes on button press in Stop_Button.
function Stop_Button_Callback(hObject, eventdata, handles)
% hObject    handle to Stop_Button (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.Start_Button,'UserData', 0);
set(handles.Edit_IP,'Enable','On')
clear t;

function Edit_IP_Callback(hObject, eventdata, handles)
% hObject    handle to Edit_IP (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Edit_IP as text
%        str2double(get(hObject,'String')) returns contents of Edit_IP as a
double

% --- Executes during object creation, after setting all properties.
function Edit_IP_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Edit_IP (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in Remove_bcg_static.
function Remove_bcg_static_Callback(hObject, eventdata, handles)
% hObject    handle to Remove_bcg_static (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of Remove_bcg_static

% --- Executes on button press in Average_bcg.
function Average_bcg_Callback(hObject, eventdata, handles)
% hObject    handle to Average_bcg (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(hObject,'UserData', 1);
set(handles.Background_Status,'String','');

% Hint: get(hObject,'Value') returns toggle state of Average_bcg



% --- Executes on button press in Conv1_checkbox.
function Conv1_checkbox_Callback(hObject, eventdata, handles)
% hObject    handle to Conv1_checkbox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if get(handles.Start_Button,'UserData') ~= 1
    if (hObject.Value == 1)
```

```
        handles.Processed.YData = handles.Background_mean+2048;
        set(handles.Processed,'Visible','on');
    else
        set(handles.Processed,'Visible','off');
    end
end

% Hint: get(hObject,'Value') returns toggle state of Conv1_checkbox

% --- Executes on button press in Save.
function Save_Callback(hObject, eventdata, handles)
% hObject    handle to Save (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.Save, 'UserData', 2)
set(handles.Status_message,'String','Saving data...')

% --- Executes on button press in Hampel.
function Hampel_Callback(hObject, eventdata, handles)
% hObject    handle to Hampel (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if get(handles.Start_Button,'UserData') ~= 1
    if (hObject.Value == 1)
        handles.Processed.YData = handles.Background_mean+2048;
        set(handles.Processed,'Visible','on');
    else
        set(handles.Processed,'Visible','off');
    end
end
% Hint: get(hObject,'Value') returns toggle state of Hampel

% --- Executes on button press in Averaging.
function Averaging_Callback(hObject, eventdata, handles)
% hObject    handle to Averaging (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if get(handles.Start_Button,'UserData') ~= 1
    if (hObject.Value == 1)
        handles.Processed.YData = handles.Background_mean+2048;
        set(handles.Processed,'Visible','on');
    else
        set(handles.Processed,'Visible','off');
    end
end
% Hint: get(hObject,'Value') returns toggle state of Averaging

function Average_numb_Callback(hObject, eventdata, handles)
% hObject    handle to Average_numb (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Average_numb as text
%        str2double(get(hObject,'String')) returns contents of Average_numb as a
double

% --- Executes during object creation, after setting all properties.
function Average_numb_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Average_numb (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in Abs_val.
function Abs_val_Callback(hObject, eventdata, handles)
% hObject    handle to Abs_val (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if get(handles.Start_Button,'UserData') ~= 1
    if (hObject.Value == 1)
        handles.Processed.YData = handles.Background_mean+2048;
        set(handles.Processed,'Visible','on');
    else
        set(handles.Processed,'Visible','off');
    end
end
% Hint: get(hObject,'Value') returns toggle state of Abs_val

% --- Executes on button press in Recursive_avg.
function Recursive_avg_Callback(hObject, eventdata, handles)
% hObject    handle to Recursive_avg (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if get(handles.Start_Button,'UserData') ~= 1
    if (hObject.Value == 1)
        handles.Processed.YData = handles.Background_mean+2048;
        set(handles.Processed,'Visible','on');
    else
        set(handles.Processed,'Visible','off');
    end
end
% Hint: get(hObject,'Value') returns toggle state of Recursive_avg

function Avg_koef_Callback(hObject, eventdata, handles)
% hObject    handle to Avg_koef (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Avg_koef as text
%        str2double(get(hObject,'String')) returns contents of Avg_koef as a
double

% --- Executes during object creation, after setting all properties.
function Avg_koef_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Avg_koef (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
```

```
    set(hObject,'BackgroundColor','white');
end

function Save_count_Callback(hObject, eventdata, handles)
% hObject    handle to Save_count (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Save_count as text
%        str2double(get(hObject,'String')) returns contents of Save_count as a
double

% --- Executes during object creation, after setting all properties.
function Save_count_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Save_count (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in Remove_bcg_rec.
function Remove_bcg_rec_Callback(hObject, eventdata, handles)
% hObject    handle to Remove_bcg_rec (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of Remove_bcg_rec

% --- Executes on button press in Remove_bcg.
function Remove_bcg_Callback(hObject, eventdata, handles)
% hObject    handle to Remove_bcg (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of Remove_bcg
if get(handles.Start_Button,'UserData') ~= 1
    if (hObject.Value == 1)
        handles.Processed.YData = handles.Background_mean+2048;
        set(handles.Processed,'Visible','on');
    else
        set(handles.Processed,'Visible','off');
    end
end

function Background_koef_Callback(hObject, eventdata, handles)
% hObject    handle to Background_koef (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Background_koef as text
%        str2double(get(hObject,'String')) returns contents of Background_koef
as a double

% --- Executes during object creation, after setting all properties.
```

```
function Background_koef_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Background_koef (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in convolution.
function convolution_Callback(hObject, eventdata, handles)
% hObject    handle to convolution (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of convolution
if get(handles.Start_Button,'UserData') ~= 1
    if (hObject.Value == 1)
        handles.Processed.YData = handles.Background_mean+2048;
        set(handles.Processed,'Visible','on');
    else
        set(handles.Processed,'Visible','off');
    end
end

% --------------------------------------------------------------------
function Files_Callback(hObject, eventdata, handles)
% hObject    handle to Files (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --------------------------------------------------------------------
function Settings_Callback(hObject, eventdata, handles)
% hObject    handle to Settings (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
pos_size = get(handles.figure1,'Position');

user_response = Settings('Title','Settings');
switch user_response
    case "Ok"

    case "Cancel"
end

% --------------------------------------------------------------------
function Exit_Callback(hObject, eventdata, handles)
% hObject    handle to Exit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```