ივანე ჯავახიშვილის სახელობის თბილისის

სახელმწიფო უნივერსიტეტი

ლუკა წულაია

საავტომობილო ნომრების დისკრიმინირების სისტემა პარკირების სივრცის ავტომატიზაციისთვის

ზუსტი და საბუნებისმეტყველო მეცნიერებათა ფაკულტეტის ელექტრული და ელექტრონული ინჟინერიის მიმართულება

სამაგისტრო ნაშრომი

ხელმძღვანელი: დავით კაკულია

ფიზიკის და მათემატიკის მეცნიერებათა კანდიდატი

თბილისი, 2017

ანოტაცია

წინამდებარე ნაშრომში განვიხილავთ მანქანურ სწავლებაზე დაფუძნებულ საავტომობილო სანიშნე ნომრების ამომცნობ სისტემას და მის ბაზაზე აგებულ პარკირების სივრცის ავტომატიზაციის სერვისს. აღნიშნული სერვისით შესაძლებელია პარკირების სივრცეში გამავალი მანქანების სიის შედგენა და შლაგბაუმების გაღების ავტომატიზაცია წინასწარ შედგენილი სიის მიხედვით. ასევე არის რეალურ დროში ადმინისტრირების შესაძლებლობა მობილური აპლიკაციის საშუალებით, მათ შორის უცნობი ნომრების ბაზაში მარტივად ჩამატების შესაძლებლობა.

აღსანიშნავია, რომ ოპტიკური ამოცნობისთვის გამოყენებული ღია ბიბლიოთეკა არ იყო გათვლილი ქართულ სანომრე ნიშნებზე და შრიფტზე. პროექტის ფარგლებში შევადგინეთ ქართული სანომრე ნიშნების ბაზა და მისი საშუალებით თავიდან გავწვრთენით არსებული სისტემა, რათა გაგვეზარდა ნომრების ამოცნობის სიზუსტე.

Abstract

In this Paper, we review parking space automation service which uses Automatic License Plate Recognition System based on machine learning. This service can be used to control traffic flow through the parking space and automate entrance gates based on predefined database. Service can be administered using mobile App, which can also be used to quickly add new license plates.

Used open-source library for Optical Character Recognition (OCR) was not intended to be used on Georgian license plates and fonts. During the project, we collected new database of Georgian license plates and retrained default system to increase detection confidence.

შინაარსი

ანოტაცია 2

Abstract 2

შესავალი 4

სერვისის სტრუქტურა 4

ჩაშენებული სისტემა 6

სანომრე ნიშნების ოპტიკური ამოცნობის სისტემის სტრუქტურა 7

ცენტრალური სერვერი 8

კლიენტი აპლიკაცია 9

დასკვნა 9

გამოყენებული ლიტერატურა 10

დანართები 11

შესავალი

სანომრე ნიშნების ამოცნობის ავტომატური სისტემები, დღესდღეობით, მრავალი მიმართულებით გამოიყენება: საგზაო მოძრაობის მონიტორინგისთვის, მოძრაობის წესების დარღვევის აღრიცხვისთვის, პარკირების კონტროლისთვის, სხვადასხვა ტერიტორიაზე დაშვების შეზღუდვისთვის და ა. შ. ასეთი სისტემები უმეტესწილად მოიცავს ინფრაწითელ ან ფერად კამერებს, რადარებს (ზოგჯერ), ციფრულ შესავალსა და გამოსავალს, ასოების ოპტიკური ამოცნობის პროგრამულ უზრუნველყოფას (OCR – Optical Character Recognition), მონაცემთა ბაზებსა და სამართავ სამომხმარებლო ინტერფეისს.

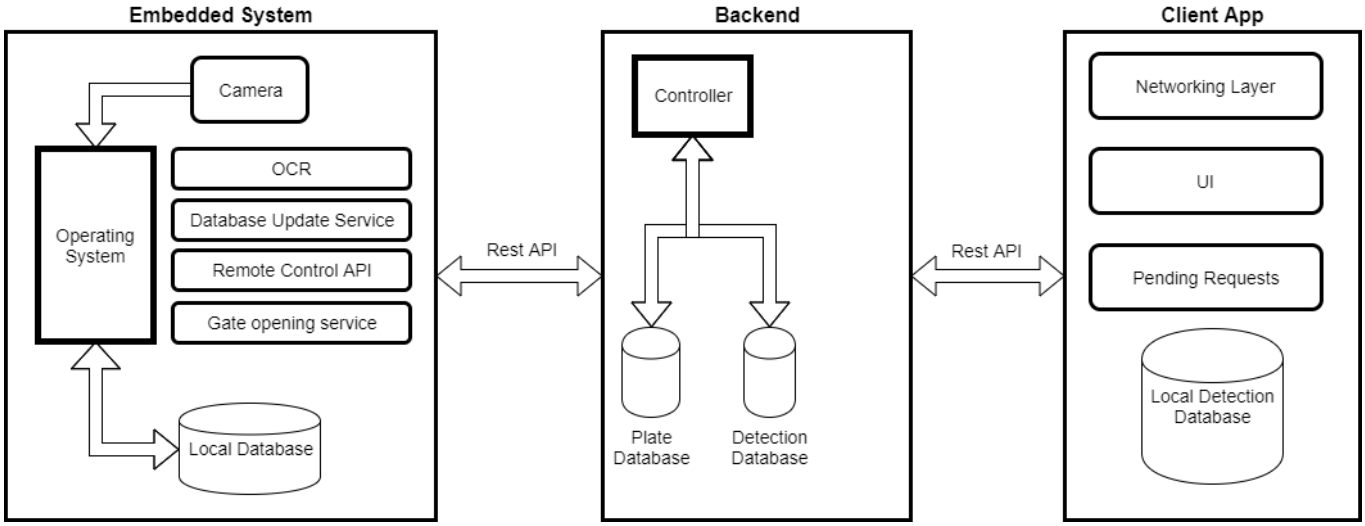
წინამდებარე ნაშრომის მთავარი მიზანია პარკირების სივრცის ავტომატიზაციის და მართვის სერვისის შექმნა. განსახილველი სერვისი გამოიყენებს დაბალი რეზოლუციის ფერად კამერას და იქნება განკუთვნილი სტაციონარული სამანქანო ნომრების ამოსაცნობად. იმის მიუხედავად, რომ არსებობს მრავალი საერთაშორისო კომერციული ოპტიკური ამომცნობი სისტემა, ღია ბიბლიოთეკების რაოდენობა არც ისე მრავალია, ხოლო ქართულ სანიშნე ნომრებზე გათვლილი ღია კოდის მქონე სისტემები ამჟამად საერთოდ არ არსებობს. ჩვენ გამოვიყენებთ სამანქანო ნომრების ამოცნობის ერთ-ერთ ღია ბიბლიოთეკას, OpenALRP-ს, ასევე ავაგებთ ცენტრალურ სერვერს ASP.Net Core ბაზაზე და მართვის მობილურ აპლიკაციას. პირველ რიგში მიმოვიხილავთ სერვისის მთლიან არქიტექტურას, ხოლო შემდგომ გადავალთ მისი კომპონენტების მუშაობის დაწვრილებით აღწერაზე.

OpenALPR-ის ბაზისური ამოცნობის სისტემა არ იყო მორგებული ქართული სანიშნე ნომრების ამოცნობაზე. ჩვენ ავაგეთ ქართული სანომრე ნიშნების ბაზა და მოვახდინეთ სისტემის გაწვრთნა ახალი მონაცემების მიხედვით რათა გაგვეზარდა ნიშნების ამოცნობის სიზუსტე.

სერვისის სტრუქტურა

სერვისის აგების მთავარი პრიორიტეტები იყო დეველოპმენტის/მომსახურების სიმარტივე და მინიმალური საჭირო ფუნქციონალის რეალიზაცია. სამუშაო გაიყო ოთხ ძირითად ნაწილად:

1) ჩაშენებული სისტემის აგება თავისი კამერით, რომელიც პასუხისმგებელი იქნება ნომრების ამოცნობაზე და ამ ინფორმაციის სერვერისთვის გადაცემაზე ინტერნეტის საშუალებით
2) გაუწვრთნელი ამოცნობის სისტემის სიზუსტის დადგენა ქართულ ნომრებთან მიმართებაში და საჭიროების შემთხვევაში მისი გაუმჯობესება
3) ცენტრალური სერვერის (Backend) და მონაცემთა ბაზის აგება, სადაც ინახება ინფორმაცია ყველა დაფიქსირებული ნომრის შესახებ.
4) მობილური კლიენტი აპლიკაცია, რომლითაც შესაძლებელია სისტემის მართვა და საჭიროების შემთხვევაში ნომრების ჩამატება



(ნახ. 1) სერვისის არქიტექტურა

მთლიანი სერვისის არქიტექტურა შეგიძლიათ იხილოთ ნახ. 1-ზე. მარცხენა ნაწილში მოცემულია ჩაშენებული სისტემა, შუაში ცენტრალური სერვერი ხოლო მარჯვნივ კლიენტის აპლიკაცია. ისინი ერთმანეთს REST API-ს საშუალებით უცვლიან ინფორმაციას.

ამ ეტაპზე ჩვენ მიზნად არ ვისახავდით სერვისის შემდგომ სკალირებას და მრავალი კამერის მხარდაჭერას. ასევე მობილური აპლიკაციისთვის არ ვიყენებთ ე.წ. “Push Notifications”-ს, რათა არ გავართულოთ მთლიანი არქიტექტურა ან არ გამოვიყენოთ დამატებითი გარე სერვისები.

სერვისის სასიცოცხლო ციკლი ასე გამოიყურება:

- ჩაშენებული სისტემა კამერის საშუალებით ახდენს უწყვეტ მონიტორინგს

- ყოველ წინასწარ განსაზღვრულ ფიქსირებულ დროის ერთეულში ჩაშენებული სისტემა ამოწმებს ბაზის განახლებებს ცენტრალურ სერვერთან.
- ყოველ წინასწარ განსაზღვრულ ფიქსირებულ დროის შუალედში სისტემა ეძებს კადრში არსებულ სანომრე ნიშანს
- თუ ამოცნობილი სანომრე ნიშანი არის ჩაშენებული სისტემის ლოკალურ მონაცემთა ბაზაში, მაშინ ხდება შლაგბაუმის გაღება
- ამოცნობილი ნიშნის შესახებ ინფორმაცია (ამოცნობილი სიმბოლოები; სანდოობის კოეფიციენტი; შეესაბამება თუ არა საქართველოს სანომრე შაბლონს; სავარაუდო გამცემი ქვეყანა; იყო თუ არა ნომერი ლოკალურ ბაზაში), იგზავნება ცენტრალურ სერვერზე.
- თუ დაფიქსირებული ნომერი არაა ლოკალურ მონაცემთა ბაზაში მაშინ სერვერზე კამერის ბოლო კადრიც იგზავნება.
- ცენტრალური სერვერი, როგორც კი მისდის ინფორმაცია ახალი ნომრის შესახებ, რომელიც ბაზაში არაა შეყვანილი, შეტყობინებას უგზავნის მობილურ კლიენტს (სავარაუდო დაფიქსირებული ნომერი, კადრი კამერიდან, დრო). მობილური კლიენტიდან შესაძლებელია ამ შეტყობინების იგნორირება ან ახალი ნომრის მონაცემთა ბაზაში ჩამატება.
- ახალი ნომრის ჩამატების დასტურის შემთხვევაში მობილური აპლიკაცია აგზავნის ინფორმაციას სერვერზე, სადაც ხდება მისი ბაზაში ჩამატება. ბაზის ცვლილებები ასევე იგზავნება ჩაშენებულ სისტემაში.

ჩაშენებული სისტემა

ჩაშენებული სისტემა მოიცავს სანომრე ნიშნების ამოცნობის ოპტიკურ სისტემას, ლოკალურ მონაცემთა ბაზას ნომრების სიით და ცენტრალურ სერვერთან კავშირის ნაწილს. თუ ცენტრალურ სერვერთან კავშირი გაწყვეტილია, სისტემას დამოუკიდებლად შეუძლია ფუნქციონირება. ასევე სატესტო მონაცემების დასამუშავებლად ჩამატებულია დამატებითი ფუნქციონალი - შესაძლებელია ვებკამერის ნაცვლად სურათების სიის მითითება შედეგების სწრაფად დასამუშავებლად. ამოცნობის შედეგების გატანა ცალკე ფაილად არის შესაძლებელი.

ჩაშენებული სისტემის აგება მოხდა WinForms პლატფორმაზე. აპლიკაციის ძირითადი ნაწილის კოდი მოცემულია დანართში.

სანომრე ნიშნების ოპტიკური ამოცნობის სისტემის სტრუქტურა

სანომრე ნიშნების ამოსაცნობად, როგორც უკვე ვახსენეთ ვიყენებთ ღია ბიბლიოთეკას OpenALPR-ს. OpenALPR დაფუძნებულია ორ ბიბლიოთეკაზე, OpenCV-სა და Tesseract-ზე. OpenCV გამოიყენება სურათების დამუშავებისთვის, სანომრე ნიშნების არეების დეტექტირების და ასოების სეგმენტაციისთვის, ხოლო Tesseract მანქანური სწავლების საშუალებით ასოების ამოცნობაზეა პასუხისმგებელი.

მიწოდებულ კადრებს OpenALPR სისტემა რამდენიმე ეტაპად ამუშავებს, ესენია:

- ნომრების პოტენციური რეგიონების პოვნა სურათზე;
- სურათის შავ-თეთრად გარდაქმნა;
- ასოების არეების დეტექტირება - ასოების ზომის „ლაქების“ პოვნა ნომრების რეგიონებში;
- სანომრე ნიშნების ზუსტი რეგიონის ზომის/ფორმის პოვნა;
- მოტრიალება - პერსპექტივის გარდაქმნა ისე, რომ ნომერს პირდაპირ ვუყურებდეთ (Deskewing);
- ასოების სეგმენტაცია - ასოების რეგიონების განაწილება და გასუფთავება მათი შემდგომი დეტექტირებისთვის;
- ასოების ოპტიკური ამოცნობის სისტემის (OCR) საშუალებით თითოეული ასოს ცალ-ცალკე განსაზღვრა. ამ ეტაპზე ვაბრუნებთ რამდენიმე შესაძლო ვარიანტს შესაბამისი სანდოობის კოეფიციენტებით;
- ყველაზე მაღალი სანდოობის კოეფიციენტების მქონე ნომრების სიის შექმნა. ასევე ხდება ნომრების შესაბამისობის შემოწმება არჩეულ რეგიონულ შაბლონთან. სხვადასხვა ქვეყნის შაბლონები პროგრამაში წინასწარაა განსაზღვრული (მაგ.: AA000AA, პირველი და ბოლო ორი სიმბოლო უნდა იყოს ინგლისური ასო, შუა სამი სიმბოლო კი ციფრი)

OpenALRP-ს მრავალი ქვეყნის ნომრის ამოცნობის მხარდაჭერა აქვს, თუმცა თუ ამოცნობის სისტემა ცუდად მუშაობს, შესაძლებელია მისი ოპტიმიზაცია. ამისთვის

სხვადასხვა გზები არსებობს. ერთ-ერთი მეთოდია მანქანური სწავლების საშუალებით Tesseract-ის ასოების ამოცნობის სისტემის მორგება საჭირო ქვეყნის სანომრე ნიშნების შრიფტზე.

როგორც პირველადმა ტესტირებამ აჩვენა ქართული ნომრების ამოცნობა დაბალი სიზუსტით ხდებოდა. ამას ორი მიზეზი ჰქონდა:

- საქართველოში არის ძირითადი ორი ტიპის ნომრის შაბლონია, ახალი და ძველი სტილით ჩაწერილი, მათ ასევე სხვადასხვა შრიფტი აქვთ
- დაშვებულია სხვადასხვა ფორმის სანომრე ნიშნები

იმისათვის რომ სისტემამ სანდოდ შეძლოს ნომრების ამოცნობა საჭიროა, რომ ის ცალ-ცალკე მოვარგოთ ძველ და ახალ სანომრე შრიფტებს. ასევე დავასწავლოთ მას სხვადასხვა ფორმის სანომრე ნიშნების რეგიონების ამოცნობა. პირველ შემთხვევაში საჭიროა ახალი და ძველი შრიფტის მქონე მინიმუმ ასი ფოტოს მოძიება და დამუშავება. მეორე შემთხვევაში საჭიროა ყველაზე ცოტა რამდენიმე ათასი კადრის მოგროვება, რომ სანომრე ნიშნის რეგიონის ამოცნობის სიზუსტე საგრძნობლად გაიზარდოს. სამუშაოს მასშტაბებიდან გამომდინარე ნომრების ამოცნობის ოპტიმიზაცია მხოლოდ პირველი მეთოდით მოხდა. ამისთვის შეგროვდა 200-ზე მეტი ახალის სტილის სანომრე ნიშნის სურათი და 100-ზე მეტი ძველის. ამ სურათებიდან მოხდა ასოების სიმბოლოების ამოღება და OpenALRP-ის მანქანური სწავლების ნაწილის თავიდან გაწვრთნა. ამით საგრძნობლად გაიზარდა ქართული სანომრე ნიშნების ამოცნობის სიზუსტე (იხ. დანართი სურათების ნიმუშების სანახავად)

ცენტრალური სერვერი

ცენტრალური სერვერის მთავარი დანიშნულებაა ნომრების ძირითადი სიის შენახვა, ამ სიის საჭიროების შემთხვევაში ჩაშენებული სისტემისთვის მიწოდება და ნომრების დეტექტირების შედეგების შენახვა.

ცენტრალური სერვერი ავაგეთ ASP.Net Core ტექნოლოგიაზე. ბაზებისთვის გამოვიყენეთ Entity Framework.

კლიენტი აპლიკაცია

მობილურ მართვის აპლიკაციას აქვს ორი ძირითადი მიზანი: დაფიქსირებული ნომრების ბაზის დათვალიერება და საჭიროების შემთხვევაში „სტუმრის“ ნომრის ჩამატება. კლიენტ აპლიკაციას არ აქვს პირდაპირი კონტაქტი ჩაშენებულ სისტემასთან. კავშირი ხდება ბექენდის საშუალებით. აპლიკაციის სერვერთან ურთიერთქმედების კოდის ნიმუში შეგიძლიათ იხილოთ დანართში.

მობილური აპლიკაცია აგებულია თამაშის ძრავა Unity3D-ში და დაწერილია C#-ზე. Unity3D გვაძლევს საშუალებას სწრაფად ავაგოთ პროტოტიპი და ამასთანავე გავუშვათ მრავალ პლატფორმაზე (Windows/Mac/Linux, Android/iOS/WinPhone)

დასკვნა

განხორციელდა პარკირების სივრცის ავტომატიზაციის სერვისის აგება. ეს სისტემა მოიცავს ჩაშენებულ სისტემას, სერვერის ნაწილსა და სამართავ მობილურ აპლიკაციას. თავიდან ნომრების ამოცნობა ცუდი სიზუსტით ხდებოდა, თუმცა დადგინდა ამის მიზეზი, მოგროვდა სანომრე ნიშნების ბაზა და სისტემა თავიდან გაიწვრთნა. შედეგად ამოცნობის სიზუსტე საკმაოდ მაღალია იმისთვის, რომ მოხდეს სერვისის მომავალი კომერციალიზაცია.

აღსანიშნავია, რომ სერვისის ინტეგრაცია იაფი და მარტივია არსებულ სისტემებში. მისი გამოყენება შესაძლებელია ასევე ე.წ. USB-Stick ტიპის კომპიუტერებშიც კი (იხ. ნახ. 2).

სერვისის გამოყენების პოტენციალიდან გამომდინარე შემდგომში სერვისის გაუმჯობესება შესაძლებელია OpenALPR-ის სანომრე ნიშნების რეგიონების დეტექტირების გაწვრთნით. დიდი მნიშვნელობა ექნება სერვისის ფუნქციონალის გაფართოებასაც, მრავალი კამერის მხარდაჭერის ჩამატებასა და სამომხმარებლო ვებ-ინტერფეისის შექმნას. მთლიანი პროექტის კოდი ღიადაა ხელმისაწვდომი ყველასთვის.

https://bitbucket.org/luchianno/master-thesis-2017

https://bitbucket.org/luchianno/master-thesis-2017-client-app

ნახ. 2 პორტატული კომპიუტერი

გამოყენებული ლიტერატურა

Atanassov, A. (2012). Advanced software architecture of an automatic vehicle number plate recognition system. *Journal of the University of Chemical Technology and Metallurgy*, 77-82.

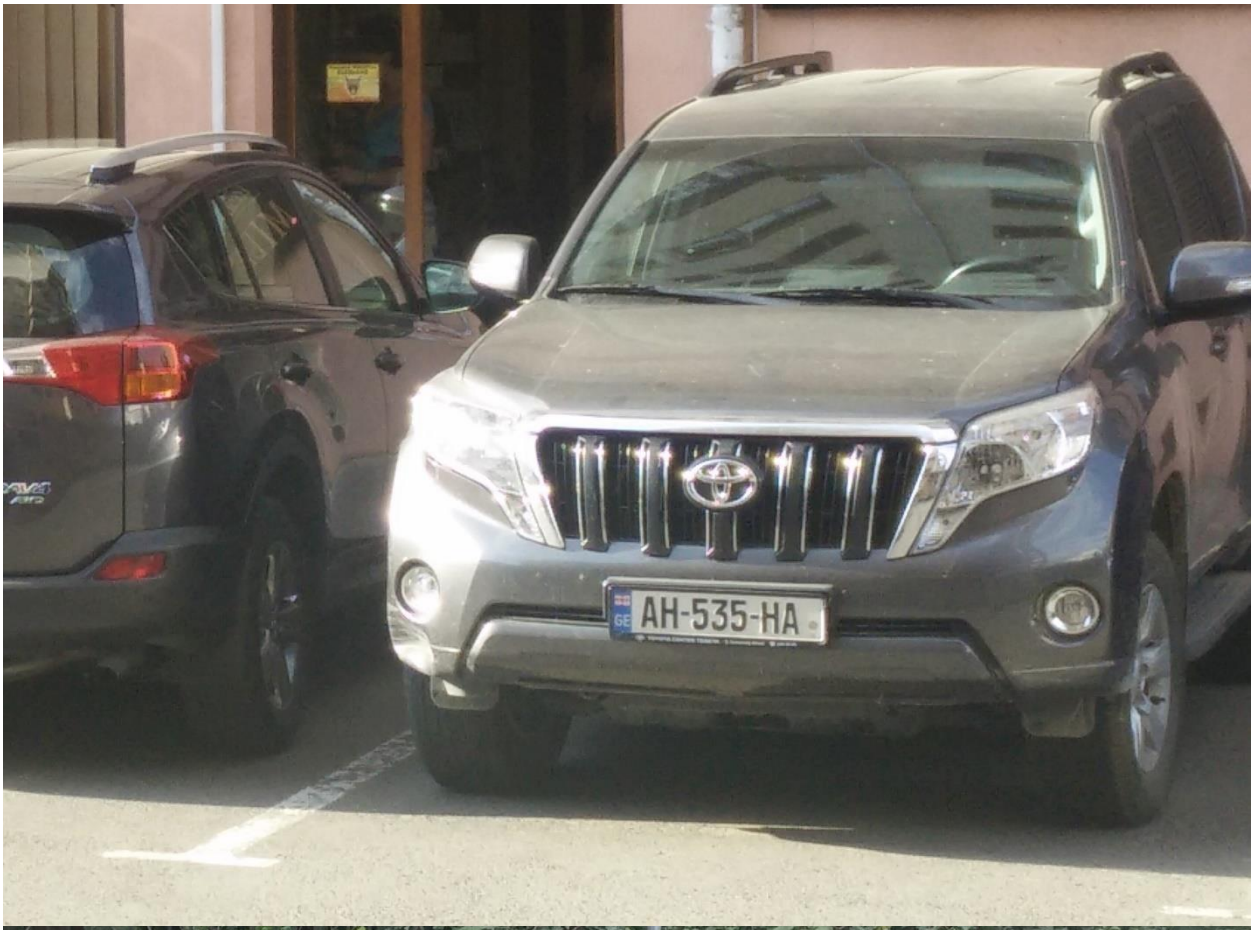
OpenARPR Documentation. (2017 წლის 20 6). მოპოვებული http://doc.openalpr.com-დან

დანართები

სისტემის გაწვრთვნისთვის გამოყენებული ნომრების ნიმუშები (ახალი სტილის ნომრები):


SUBARU
TI-868-KO


ZZ-990-GG


GE
AH-535-HA


GE
BE-
099-NZ

ძველი სტილით ნომრების ნიმუშები:


KALE Bergstrom
SIBEL
FIRAT
TRANSIT
T·MC·937


VIA·600

ჩაშენებული სისტემა

WinForms აპლიკაციის ძირითადი ნაწილი კოდი რომელიც პასუხისმგებელია ვებკამერიდან ან ფოტოებიდან ნომრების ამოცნობაზე

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using openalprnet;
using Emgu.CV;
using System.Net.Http;

namespace Client_Forms
{
    public partial class Form1 : Form
    {
        Capture webcam;
        Mat lastFrame = new Mat();
        AlprNet alpr;
        readonly HttpClient client = new HttpClient();
        string sendPath;
        Dictionary<string, float> lastResults = new Dictionary<string,
float>();

        public Form1()
```

```
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            webcam = new Emgu.CV.Capture();
            webcam.ImageGrabbed += Webcam_ImageGrabbed;
            webcam.Start();

            alpr = new AlprNet("eu", "openalpr.conf", "/runtime_data");
            if (!alpr.IsLoaded())
            {
                throw new InvalidOperationException("OpenAlpr failed to
load");
            }
            alpr.DefaultRegion = "ge";
        }

        private void Webcam_ImageGrabbed(object sender, EventArgs e)
        {
            webcam.Retrieve(lastFrame);
            pictureBox1.Image = lastFrame.Bitmap;
        }

        public Image cachedImage;

        private void Path_Button_Click(object sender, EventArgs e)
        {
            Stream stream = null;

            openFileDialog1.InitialDirectory = "c:\\";
            openFileDialog1.Filter = "Image files |*.jpeg;*.jpg;*.png;";
```

```
            openFileDialog1.RestoreDirectory = true;
            openFileDialog1.InitialDirectory =
@"C:\Users\lucho\OneDrive\Projects (Drive)\Master_Tsu\Plates";

            if (openFileDialog1.ShowDialog() == DialogResult.OK)
            {
                try
                {
                    if ((stream = openFileDialog1.OpenFile()) != null)
                    {
                        textBox1.AppendText("Selected: " +
openFileDialog1.FileName);
                        textBox1.AppendText(Environment.NewLine);

                        using (stream)
                        {
                            cachedImage = Image.FromStream(stream);
                            pictureBox1.Image = cachedImage;
                        }
                    }
                }
                catch (Exception ex)
                {
                    MessageBox.Show("Error: Could not read file from disk.
Original error: " + ex.Message);
                }
            }
        }

        public void RecognizeImage(string path)
        {
            if (!File.Exists(path))
            {
```

```
                textBox1.AppendText("Path does not exist");
                return;
            }

            var results = alpr.Recognize(path);
            int i = 0;
            foreach (var result in results.Plates)
            {
                textBox1.AppendText(String.Format("Plate {0}: {1}
result(s)\n", i++, result.TopNPlates.Count));
                textBox1.AppendText(String.Format("  Processing Time: {0}
msec(s)\n", result.ProcessingTimeMs));
                foreach (var plate in result.TopNPlates)
                {
                    textBox1.AppendText(String.Format("  - {0}\t Confidence:
{1}\tMatches Template: {2}\n", plate.Characters,
                                        plate.OverallConfidence,
plate.MatchesTemplate));
                }
            }
        }

        public string RecognizeImage(Bitmap image)
        {
            StringBuilder result = new StringBuilder();
            alpr.TopN = (int)numericUpDown1.Value;
            if (!alpr.IsLoaded())
            {
                result.AppendLine("OpenAlpr failed to load!");
            }
            else
            {
                var results = alpr.Recognize(image);
```

```
                if (results.Plates.Count != 0)
                {
                    int i = 0;
                    foreach (var item in results.Plates)
                    {
                        result.AppendFormat("Plate {0}: {1} result(s)" +
Environment.NewLine, i++, item.TopNPlates.Count);
                        result.AppendFormat("  Processing Time: {0} msec(s)"
+ Environment.NewLine, item.ProcessingTimeMs);
                        foreach (var plate in item.TopNPlates)
                        {

                            result.AppendFormat("- {0}\t Confidence:
{1}\tMatches Template: {2}" + Environment.NewLine,
                                                plate.Characters,
                                                plate.OverallConfidence,
                                                plate.MatchesTemplate);
                        }
                    }
                }
            }
            return result.ToString();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            RecognizeImage(openFileDialog1.FileName);
        }

        private void button2_Click(object sender, EventArgs e)
        {
            backgroundWorker1.RunWorkerAsync(pictureBox1.Image.Clone());
```

```
            button2.Enabled = false;
        }

        private void backgroundWorker1_DoWork(object sender, DoWorkEventArgs
e)
        {
            e.Result = RecognizeImage((Bitmap)e.Argument);
        }

        private void backgroundWorker1_RunWorkerCompleted(object sender,
RunWorkerCompletedEventArgs e)
        {
            button2.Enabled = true;
            textBox1.AppendText(((string)e.Result) + Environment.NewLine);
        }

        private void timer1_Tick(object sender, EventArgs e)
        {
            if (!backgroundWorker1.IsBusy)
                backgroundWorker1.RunWorkerAsync(pictureBox1.Image.Clone());
        }

        private void checkBox1_CheckedChanged(object sender, EventArgs e)
        {
            if (checkBox1.Checked)
            {
                timer1.Interval = (int)intervalUpDown.Value;
                timer1.Start();
                button2.Enabled = false;
                intervalUpDown.Enabled = false;
            }
            else
            {
```

```
            timer1.Stop();
            button2.Enabled = true;
            intervalUpDown.Enabled = true;
        }
    }

    private void detectImagesButt_Click(object sender, EventArgs e)
    {
        autoDetectionPanel.Enabled = false;
        testingPanel.Enabled = false;
        textBox1.AppendText("Starting to detect images");
        imageScanWorker.RunWorkerAsync();
    }

    private void browseFolderButt_Click(object sender, EventArgs e)
    {
        if (folderBrowserDialog1.ShowDialog() == DialogResult.OK)
        {
            folderBox.Text = folderBrowserDialog1.SelectedPath;
        }
    }

    private void imageScanWorker_ProgressChanged(object sender,
ProgressChangedEventArgs e)
    {
        progressBar1.Value = e.ProgressPercentage;
    }

    private void imageScanWorker_RunWorkerCompleted(object sender,
RunWorkerCompletedEventArgs e)
    {
        autoDetectionPanel.Enabled = true;
        testingPanel.Enabled = true;
```

```
            textBox1.AppendText("Done" + Environment.NewLine);
        }

        private void imageScanWorker_DoWork(object sender, DoWorkEventArgs e)
        {
            var lines = File.ReadAllLines(Path.Combine(folderBox.Text,
"real.txt"));
            var real = new Dictionary<string, string>(lines.Length);

            foreach (var item in lines)
            {
                var pair = item.Split(' ', '\t');
                real.Add(pair[0], pair[1]);
            }

            try
            {
                var path = folderBox.Text;
                var files = new List<string>(Directory.GetFiles(path, "*.*",
SearchOption.TopDirectoryOnly)
                      //.OrderByDescending<FileInfo>(x=>x.Name),
                      .Where(s => s.EndsWith(".png",
StringComparison.OrdinalIgnoreCase) ||
                      s.EndsWith(".jpg",
StringComparison.OrdinalIgnoreCase)));

                StringBuilder results = new StringBuilder("file  result
confidence" + Environment.NewLine);

                List<AlprPlateResultNet> plates = new
List<AlprPlateResultNet>();
                int i = 0;
                foreach (var file in files)
```

```
                {
                    plates.Clear();
                    plates = alpr.Recognize(file).Plates;
                    var detectionResult = "NaN";
                    float confidence = 0;
                    if (plates.Count != 0)
                    {
                        var plateResult = plates[0].TopNPlates[0];
                        detectionResult = plateResult.Characters;
                        confidence = plateResult.OverallConfidence;
                    }

                    var fileName = Path.GetFileNameWithoutExtension(file);
                    results.AppendFormat("{0}\t{1}\t{2}\t{3}{4}",
                        fileName,
                        detectionResult,
                        confidence,
                        real[fileName] == detectionResult,
                        Environment.NewLine);
                    (sender as
BackgroundWorker).ReportProgress((int)(((float)i / files.Count) * 100f),
null);
                    i++;
                }

                File.WriteAllText(Path.Combine(path, "results.txt"),
results.ToString());
            }
            catch (Exception ex)
            {
                // throw ex;
                MessageBox.Show(ex.ToString());
```

```
                //  MessageBox.Show("Error: Could not read file from disk.
Original error: " + ex.Message);
            }
        }
    }
}
```

მობილური კლიენტი

ნომრის დეტექტირების მოვლენის მოდელი:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System;

[Serializable]
public class PlateDetection
{
    public int id;
    public DateTime time;
    public string plateNumber;
    public float confidence;

    public string region;

    public float imagePath;

    public class PlateDetectionComparer : IEqualityComparer<PlateDetection>
    {
        public bool Equals(PlateDetection x, PlateDetection y)
        {
            return x.id == y.id;
        }
```

```
        public int GetHashCode(PlateDetection obj)
        {
            return obj.id;
        }
    }
}
```

დამხმარე კლასი სიების სერიალიზაციისთვის:

```
using System;

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public static class JsonHelper
{
    public static T[] FromJson<T>(string json)
    {
        Wrapper<T> wrapper = JsonUtility.FromJson<Wrapper<T>>(json);
        return wrapper.Items;
    }

    public static string ToJson<T>(T[] array)
    {
        Wrapper<T> wrapper = new Wrapper<T>();
        wrapper.Items = array;
        return JsonUtility.ToJson(wrapper);
    }

    public static string ToJson<T>(T[] array, bool prettyPrint)
    {
        Wrapper<T> wrapper = new Wrapper<T>();
        wrapper.Items = array;
```

```
        return JsonUtility.ToJson(wrapper, prettyPrint);
    }

    [Serializable]
    private class Wrapper<T>
    {
        public T[] Items;
    }
}
```

სერვერიდან ნომრების ამოცნობის შესახებ ინფორმაციის მიღებასა და შენახვაზე პასუხისმგებელი კლასი:

```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Networking;
using UnityEngine.Events;

public class ListController : MonoBehaviour
{
    public HashSet<PlateDetection> List;
    public UnityEvent ReloadCompleted;

    public string Path;

    void Awake()
    {
        List = new HashSet<PlateDetection>(new
PlateDetection.PlateDetectionComparer());
    }

    void Start()
    {
```

```
        TestSerialize();
    }

    // try to load information from server
    public void Reload()
    {
        StartCoroutine(SendRequest());
    }

    IEnumerator SendRequest()
    {
        UnityWebRequest www = UnityWebRequest.Get(Path);
        yield return www.Send();

        if (www.isError)
        {
            Debug.Log(www.error);
        }
        else
        {
            Debug.Log(www.downloadHandler.text);
            // deserialize information from server
            var temp =
JsonUtility.FromJson<PlateDetection[]>(www.downloadHandler.text);
            foreach (var item in temp)
            {
                // this is hashset, we cant create any dublicates
                List.Add(item);
            }
            ReloadCompleted.Invoke();
        }
    }

    // serialize/deserialize test
    public void TestSerialize()
```

```
    {
        var a = new PlateDetection { id = 0, plateNumber = "AA982HH", confidence
= 90f, time = DateTime.Now, region = "ge" };
        var b = new PlateDetection { id = 1, plateNumber = "BB000BB", confidence
= 96f, time = DateTime.Now, region = "ge" };
        var list = new List<PlateDetection>();
        var json = JsonHelper.ToJson(new PlateDetection[2] { a, b }, true);
        Debug.Log(json);

        var temp = JsonHelper.FromJson<PlateDetection>(json);
        Debug.Log(temp);
        Debug.Log(temp.Length);

        foreach (var item in temp)
        {
            Debug.Log(item.region);
        }
    }

}
```

დამხმარე აპლიკაცია სურათების დამუშავებისთვის და სორტირებისთვის

მცირე დამხმარე პროგრამა რომელიც გარდაქმნის ფოტოებს საჭირო ფორმატში და უცვლის მათ სახელებს

```
using System;
using System.IO;
using System.Linq;
using System.Drawing;

namespace RenameTool
{
    class Program
```

```
    {
        static void Main(string[] args)
        {
            var path = AppDomain.CurrentDomain.BaseDirectory;
            var filename = @"converted\plate{0}.jpg";

            Console.WriteLine(path);
            Console.WriteLine("Template is: '{0}'", filename);
            Console.WriteLine("Enter starting index", filename);

            if (!int.TryParse(Console.ReadLine().Trim(), out int startIndex))
            {
                Console.WriteLine("Not integer. Bye");
                Console.ReadKey();
                return;
            }
            Console.WriteLine("Press any key to start");
            Console.ReadKey();

            var files = Directory.GetFiles(path, "*.*",
SearchOption.TopDirectoryOnly)
                  //.OrderByDescending<FileInfo>(x=>x.Name),
                  .Where(s => s.EndsWith(".png",
StringComparison.OrdinalIgnoreCase) ||
                  s.EndsWith(".jpg", StringComparison.OrdinalIgnoreCase));

            foreach (var item in files)
            {
                Console.WriteLine("Found: {0}", item);
            }

            Directory.CreateDirectory(Path.Combine(path, "converted"));
            var i = 0;
```

```
            foreach (var item in files)
            {
                var image = Image.FromFile(item);
                var tempName = Path.Combine(path, String.Format(filename, (i
+ startIndex).ToString("00000")));
                Console.WriteLine(tempName);

                using (FileStream fs = new FileStream(tempName,
FileMode.Create, FileAccess.ReadWrite))
                {
                    image.Save(fs, System.Drawing.Imaging.ImageFormat.Jpeg);
                }

                i++;
            }
            Console.WriteLine("Converted and renamed {0} images", i);
            Console.ReadKey();
        }
    }
      }
```