ДОДАТОК А

(довідковий)

Фрагменти похідного коду програми

```
#PoseRecognition_code
import numpy as np
from enum import Enum
import cv2
import parse_openpose_json
import matplotlib.pyplot as plt


def test():
    json_data_path = 'data/json_data/'
    images_data_path = 'data/image_data/'
    model = "foto1"
    input = "kleuter2"
    model_json = json_data_path + model + '.json'
    input_json = json_data_path + input + '_keypoints.json'

    model_image = images_data_path + model + '.jpg'
    input_image = images_data_path + input + '.jpg'

    model_features = parse_openpose_json.parse_JSON_multi_person(model_json)
    assert type(model_features) is list
    img = draw_humans(model_image, model_features, True)
    plt.figure()
    plt.imshow(img)
    plt.show()



class CocoPart(Enum):
    Nose = 0
    Neck = 1
    RShoulder = 2
    RElbow = 3
    RWrist = 4
    LShoulder = 5
    LElbow = 6
    LWrist = 7
    RHip = 8
    RKnee = 9
    RAnkle = 10
    LHip = 11
    LKnee = 12
```

```
    LAnkle = 13
    REye = 14
    LEye = 15
    REar = 16
    LEar = 17
    Background = 18

CocoColors = [[255, 0, 0], [255, 85, 0], [255, 170, 0], [255, 255, 0], [170, 255, 0], [85, 255, 0], [0, 255, 0],
              [0, 255, 85], [0, 255, 170], [0, 255, 255], [0, 170, 255], [0, 85, 255], [0, 0, 255], [85, 0, 255],
              [170, 0, 255], [255, 0, 255], [255, 0, 170], [255, 0, 85]]

CocoPairs = [
    (1, 2), (1, 5), (2, 3), (3, 4), (5, 6), (6, 7), (1, 8), (8, 9), (9, 10), (1, 11),
    (11, 12), (12, 13), (1, 0), (0, 14), (14, 16), (0, 15), (15, 17), (2, 16), (5, 17)
]   # = 19
CocoPairsRender = CocoPairs[:-2]

thickness= 4

# source: https://github.com/ildoonet/tf-pose-estimation/blob/master/src/estimator.py
def draw_humans(npimg, humans, imgcopy=False):
    #npimg = plt.imread(img_path)
    if imgcopy:
        npimg = np.copy(npimg)
    #image_h, image_w = npimg.shape[:2]
    centers = {}

    # if is type list => openpose.json van multiple persons
    if type(humans) is list:
        for human in humans:

            # draw point
            for i in range(CocoPart.Background.value):

                body_part = human[i]

                if body_part[0] == 0 and body_part[1] ==0:
                    continue
                print("laaaaaaaaaaaaaaaaa")
                center = (int(body_part[0]), int(body_part[1]))
                centers[i] = center
                cv2.circle(npimg, center, thickness, CocoColors[i], thickness=thickness, lineType=8, shift=0)

            #draw line
            for pair_order, pair in enumerate(CocoPairsRender):
                # if pair[0] not in human.body_parts.keys() or pair[1] not in human.body_parts.keys():
                #     continue

                # npimg = cv2.line(npimg, centers[pair[0]], centers[pair[1]], common.CocoColors[pair_order], 3)
                cv2.line(npimg, centers[pair[0]], centers[pair[1]], CocoColors[pair_order], thickness=thickness)


        return npimg
```

```
    else: # single-person pose  (no list, but plain np.array)
        # draw point
        for i in range(CocoPart.Background.value):

            body_part = humans[i]

            if body_part[0] == 0 and body_part[1] == 0:
                continue

            center = (int(body_part[0]), int(body_part[1]))
            centers[i] = center

            vers = 7
            #cv2.rectangle(npimg, (center[0]-vers,center[1]-vers), (center[0]+vers,center[1]+vers) ,CocoColors[i],
thickness=cv2.FILLED)
            cv2.circle(npimg, center, thickness + 1, CocoColors[i], thickness=thickness + 2, lineType=8, shift=0)

        # draw line
        for pair_order, pair in enumerate(CocoPairsRender):
            # if pair[0] not in human.body_parts.keys() or pair[1] not in human.body_parts.keys():
            #     continue

            # npimg = cv2.line(npimg, centers[pair[0]], centers[pair[1]], common.CocoColors[pair_order], 3)
            cv2.line(npimg, centers[pair[0]], centers[pair[1]], CocoColors[pair_order], thickness=thickness-1)


    return npimg

def draw_square(npimg, features):
    centers = {}

    # draw point
    for i in range(CocoPart.Background.value):

        body_part = features[i]

        if body_part[0] == 0 and body_part[1] == 0:
            continue

        center = (int(body_part[0]), int(body_part[1]))
        centers[i] = center

        vers = 7


        cv2.rectangle(npimg, (center[0] - vers, center[1] - vers), (center[0] + vers, center[1] + vers),
                      color=[abs(CocoColors[i][0]-255),abs(CocoColors[i][1]-255),abs(CocoColors[i][2]-255)],
thickness=3)


        # cv2.circle(npimg, center, thickness + 1, color=[255, 0, 170], thickness=thickness + 2, lineType=8, shift=0)

        # cv2.rectangle(npimg, (center[0] - vers, center[1] - vers), (center[0] + vers, center[1] + vers),
```

```
        #            color=[255, 0, 170], thickness=3)


    return npimg


//SERVER CODE
using System;
using System.Linq;
using PoseRecognition;
using Microsoft.AspNetCore;
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.Logging;
using Serilog;
using Serilog.Events;
using Serilog.Sinks.SystemConsole.Themes;

namespace PoseRecognition
{
    public class Program
    {
        public static void Main(string[] args)
        {
            Console.Title = "Pose Recognition";

            //Log.Logger = new LoggerConfiguration()
            //    .MinimumLevel.Debug()
            //    .MinimumLevel.Override("Microsoft", LogEventLevel.Warning)
            //    .MinimumLevel.Override("System", LogEventLevel.Warning)
            //    .MinimumLevel.Override("Microsoft.AspNetCore.Authentication", LogEventLevel.Information)
            //    .Enrich.FromLogContext()
            //    .WriteTo.File(@"PoseRecognition_log_new_logs.txt")
            //    .WriteTo.Console(outputTemplate: "[{Timestamp:HH:mm:ss} {Level}]
{SourceContext}{NewLine}{Message:lj}{NewLine}{Exception}{NewLine}", theme: AnsiConsoleTheme.Literate)
            //    .CreateLogger();

            try
            {
                                        var seed = args.Contains("/seed");
                if (seed)
                {
                    args = args.Except(new[] { "/seed" }).ToArray();
                }


                var host = BuildWebHost(args);

                if (seed)
                {
                    SeedData.EnsureSeedData(host.Services);
                }
```

```
            host.Run();
        }
        catch (Exception ex)
        {
            Log.Logger.Error("Error occured in console", ex);
        }
    }

    public static IWebHost BuildWebHost(string[] args) =>
        WebHost.CreateDefaultBuilder(args)
            .ConfigureLogging(builder =>
            {
                builder.ClearProviders();
                builder.AddSerilog();
            })
            .UseStartup<Startup>()
            .Build();
}


using System;
using System.Linq;
using System.Security.Claims;
using IdentityModel;
using IdentityServer4.EntityFramework.DbContexts;
using IdentityServer4.EntityFramework.Mappers;
using PoseRecognition;
using PoseRecognition.Data;
using PoseRecognition.Models;
using Microsoft.AspNetCore.Identity;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.DependencyInjection;
using System.Collections.Generic;
using IdentityServer4.Models;

namespace PoseRecognition
{
    public class SeedData
    {
        public static void EnsureSeedData(IServiceProvider serviceProvider)
        {
            Console.WriteLine("Seeding database...");

            using (var scope = serviceProvider.GetRequiredService<IServiceScopeFactory>().CreateScope())
            {
                scope.ServiceProvider.GetRequiredService<PersistedGrantDbContext>().Database.Migrate();

                {
                    var context = scope.ServiceProvider.GetRequiredService<ConfigurationDbContext>();
                    context.Database.Migrate();
                    EnsureSeedData(context);
                }
```

```
        {
            var context = scope.ServiceProvider.GetService<ApplicationDbContext>();
            context.Database.Migrate();

            var userMgr = scope.ServiceProvider.GetRequiredService<UserManager<ApplicationUser>>();
            var alice = userMgr.FindByNameAsync("alice").Result;
            if (alice == null)
            {
                alice = new ApplicationUser
                {
                    UserName = "alice"
                };
                var result = userMgr.CreateAsync(alice, "Pass123$").Result;
                if (!result.Succeeded)
                {
                    throw new Exception(result.Errors.First().Description);
                }

                result = userMgr.AddClaimsAsync(alice, new Claim[]{
                    new Claim(JwtClaimTypes.Name, "Alice Smith"),
                    new Claim(JwtClaimTypes.GivenName, "Alice"),
                    new Claim(JwtClaimTypes.FamilyName, "Smith"),
                    new Claim(JwtClaimTypes.Email, "AliceSmith@email.com"),
                    new Claim(JwtClaimTypes.EmailVerified, "true", ClaimValueTypes.Boolean),
                    new Claim(JwtClaimTypes.WebSite, "http://alice.com"),
                                                        new Claim("Admin", "MainAdmin"),
                                                        new Claim(JwtClaimTypes.Address, @"{
'street_address': 'One Hacker Way', 'locality': 'Heidelberg', 'postal_code': 69118, 'country': 'Germany' }",
IdentityServer4.IdentityServerConstants.ClaimValueTypes.Json)
                }).Result;
                if (!result.Succeeded)
                {
                    throw new Exception(result.Errors.First().Description);
                }
                Console.WriteLine("alice created");
            }
            else
            {
                Console.WriteLine("alice already exists");
            }

            var bob = userMgr.FindByNameAsync("bob").Result;
            if (bob == null)
            {
                bob = new ApplicationUser
                {
                    UserName = "bob"
                };
                var result = userMgr.CreateAsync(bob, "Pass123$").Result;
                if (!result.Succeeded)
                {
                    throw new Exception(result.Errors.First().Description);
                }
```

```
                result = userMgr.AddClaimsAsync(bob, new Claim[]{
                new Claim(JwtClaimTypes.Name, "Bob Smith"),
                new Claim(JwtClaimTypes.GivenName, "Bob"),
                new Claim(JwtClaimTypes.FamilyName, "Smith"),
                new Claim(JwtClaimTypes.Email, "BobSmith@email.com"),
                new Claim(JwtClaimTypes.EmailVerified, "true", ClaimValueTypes.Boolean),
                new Claim(JwtClaimTypes.WebSite, "http://bob.com"),
                new Claim(JwtClaimTypes.Address, @"{ 'street_address': 'One Hacker Way', 'locality':
'Heidelberg', 'postal_code': 69118, 'country': 'Germany' }",
IdentityServer4.IdentityServerConstants.ClaimValueTypes.Json),
                new Claim("location", "somewhere")
            }).Result;
                if (!result.Succeeded)
                {
                    throw new Exception(result.Errors.First().Description);
                }
                Console.WriteLine("bob created");
            }
            else
            {
                Console.WriteLine("bob already exists");
            }
        }

        Console.WriteLine("Done seeding database.");
        Console.WriteLine();
    }

    private static void EnsureSeedData(ConfigurationDbContext context)
    {
        if (!context.Clients.Any())
        {
            Console.WriteLine("Clients being populated");
            foreach (var client in Config.GetClients().ToList())
            {
                context.Clients.Add(client.ToEntity());
            }
            context.SaveChanges();
        }
        else
        {
            Console.WriteLine("Clients already populated");
        }

        if (!context.IdentityResources.Any())
        {
            Console.WriteLine("IdentityResources being populated");
            foreach (var resource in Config.GetIdentityResources().ToList())
            {
                context.IdentityResources.Add(resource.ToEntity());
            }
```

```
            context.SaveChanges();
        }
        else
        {
            Console.WriteLine("IdentityResources already populated");
        }

        if (!context.ApiResources.Any())
        {
            Console.WriteLine("ApiResources being populated");
            foreach (var resource in Config.GetApiResources().ToList())
            {
                context.ApiResources.Add(resource.ToEntity());
            }
            context.SaveChanges();
        }
        else
        {
            Console.WriteLine("ApiResources already populated");
        }
    }

                public static void SeedGyms(ApplicationDbContext context, ConfigurationDbContext
configurationContext)
                {
                        var gym = new Gym
                        {
                                Name = "Харьков, Салтовка, м. Героев Труда",
                                Address = "г. Харьков, ул. Ак.Павлова, 144б, ТЦ «Космос», 5 этаж ",
                                PhoneNumber = "+380577616407,+380937616407",
                                Description = "Пн - Пт : 07:00 - 23:00; Сб - Вс : 09:00 - 23:00",
                                Areas = new List<CameraArea>()
                        };
                        var gym2 = new Gym
                        {
                                Name = "Харьков, Бавария",
                                Address = "г. Харьков, ул. Свинаренко, 9 ",
                                PhoneNumber = "+380577614419,+380937614419",
                                Description = "Пн - Пт : 07:00 - 23:00; Сб - Вс : 09:00 - 23:00",
                                Areas = new List<CameraArea>()
                        };
                        var gym3 = new Gym
                        {
                                Name = "Харьков, м. Дворец Спорта",
                                Address = "г. Харьков, пр.Московский 257, НИИ «Кондиционер», 2 этаж ",
                                PhoneNumber = "+380577172520,+380937614490",
                                Description = "Пн - Пт : 07:00 - 23:00; Сб - Вс : 09:00 - 23:00",
                                Areas = new List<CameraArea>()
                        };
                        var gym4 = new Gym
                        {
                                Name = "Харьков, пос. Жуковского",
                                Address = "г. Харьков, ул. Чкалова 1 (кафе молодость)",
```

```
        PhoneNumber = "+380577616420, +380937616420",
        Description = "В настоящее время нет грядущих событий. ",
        Areas = new List<CameraArea>()
};
var gym5 = new Gym
{
        Name = "Харьков, Холодная Гора",
        Address = "г. Харьков, Полтавский шлях,184 ",
        PhoneNumber = "+380577616412,+380937616412",
        Description = "Множество групповых тренировок, помимо спортзала.",
        Areas = new List<CameraArea>()
};
var gym6 = new Gym
{
        Name = "Харьков, Салтовка",
        Address = "НИИ ПТИмаш: пр. Юбилейный 56 (пр.50 лет ВЛКСМ 56) ",
        PhoneNumber = "+380577616406, +380937616406",
        Description = "",
        Areas = new List<CameraArea>()
};
var gym7 = new Gym
{
        Name = " Харьков, Серповая",
        Address = "г. Харьков, ул. Серповая, 4а",
        PhoneNumber = "+380937617772, +380577617772",
        Description = "Есть бассейн.",
        Areas = new List<CameraArea>()
};

var camera1 = new CameraArea
{
        Name = "Camera 1 canon",
        Description = "Правый квадрат 9.6",
        Apparatuses = new List<TrainingApparatus>
        {
                new TrainingApparatus
                {
                        Name = "Orbitreck 789SED",
                        Code = "OB-789SED"
                },
                new TrainingApparatus
                {
                        Name = "Orbitreck 989SED",
                        Code = "OB-989SED"
                }
        }
};
GenerateAuthCode(camera1, configurationContext);
gym.Areas.Add(camera1);

var camera2 = new CameraArea
{
        Name = "Camera 2 vision",
```

```
        Description = "Правый квадрат 1.6",
        Apparatuses = new List<TrainingApparatus>
        {
                new TrainingApparatus
                {
                        Name = "Ручной жим",
                        Code = "RG-4"
                },
                new TrainingApparatus
                {
                        Name = "Жим лёжа",
                        Code = "GL-7"
                }
        }
};
GenerateAuthCode(camera2, configurationContext);
gym.Areas.Add(camera2);

var camera3 = new CameraArea
{
        Name = "Camera 3 vision",
        Description = "Правый квадрат 3.6",
        Apparatuses = new List<TrainingApparatus>
        {
                new TrainingApparatus
                {
                        Name = "Ручной жим",
                        Code = "RG-4"
                },
                new TrainingApparatus
                {
                        Name = "Жим лёжа",
                        Code = "GL-7"
                }
        }
};
GenerateAuthCode(camera3, configurationContext);
gym2.Areas.Add(camera3);


var camera4 = new CameraArea
{
        Name = "Camera 4 vision",
        Description = "Правый квадрат 1.6",
        Apparatuses = new List<TrainingApparatus>
        {
                new TrainingApparatus
                {
                        Name = "Ручной жим",
                        Code = "RG-4"
                },
                new TrainingApparatus
                {
```

```
                        Name = "Жим лёжа",
                        Code = "GL-7"
                }
        }
};
GenerateAuthCode(camera4, configurationContext);
gym3.Areas.Add(camera4);


var camera5 = new CameraArea
{
        Name = "Camera 5 vision",
        Description = "Правый квадрат 5.6",
        Apparatuses = new List<TrainingApparatus>
        {
                new TrainingApparatus
                {
                        Name = "Ручной жим",
                        Code = "RG-4"
                },
                new TrainingApparatus
                {
                        Name = "Жим лёжа",
                        Code = "GL-7"
                }
        }
};
GenerateAuthCode(camera5, configurationContext);
gym4.Areas.Add(camera5);


var camera6 = new CameraArea
{
        Name = "Camera 6 vision",
        Description = "Правый квадрат 6.6",
        Apparatuses = new List<TrainingApparatus>
        {
                new TrainingApparatus
                {
                        Name = "Ручной жим",
                        Code = "RG-4"
                },
                new TrainingApparatus
                {
                        Name = "Жим лёжа",
                        Code = "GL-7"
                }
        }
};
GenerateAuthCode(camera6, configurationContext);
gym5.Areas.Add(camera6);
```

```
        var camera7 = new CameraArea
        {
            Name = "Camera 7 vision",
            Description = "Правый квадрат 7.6",
            Apparatuses = new List<TrainingApparatus>
            {
                new TrainingApparatus
                {
                    Name = "Ручной жим",
                    Code = "RG-74"
                },
                new TrainingApparatus
                {
                    Name = "Жим лёжа",
                    Code = "GL-77"
                }
            }
        };
        GenerateAuthCode(camera7, configurationContext);
        gym6.Areas.Add(camera7);


        var camera8 = new CameraArea
        {
            Name = "Camera 8 vision",
            Description = "Правый квадрат 8.6",
            Apparatuses = new List<TrainingApparatus>
            {
                new TrainingApparatus
                {
                    Name = "Ручной жим",
                    Code = "RG-84"
                },
                new TrainingApparatus
                {
                    Name = "Жим лёжа",
                    Code = "GL-87"
                }
            }
        };
        GenerateAuthCode(camera8, configurationContext);
        gym7.Areas.Add(camera8);

        context.Gym.Add(gym);
        context.Gym.Add(gym2);
        context.Gym.Add(gym3);
        context.Gym.Add(gym4);
        context.Gym.Add(gym5);
        context.Gym.Add(gym6);
        context.Gym.Add(gym7);

        context.SaveChanges();
}
```

```
        private static void GenerateAuthCode(CameraArea area, ConfigurationDbContext
configurationContext)
        {
            var clientKey = area.Name.Trim().Replace(" ", "_");
            var clientSecret = GenerateSecret();
            clientKey = $"{clientKey}-{clientSecret}";

            var clientEntity = new IdentityServer4.Models.Client
            {
                ClientId = clientKey,
                AllowedGrantTypes = IdentityServer4.Models.GrantTypes.ClientCredentials,

                ClientSecrets =
                    {
                        new IdentityServer4.Models.Secret(clientSecret.Sha256())
                    },
                AllowedScopes = { "api1" }
            }.ToEntity();

            configurationContext.Clients.Add(clientEntity);

            configurationContext.SaveChanges();

            area.ClientKey = clientEntity.ClientId;
            area.AuthCode = clientSecret;
        }

        private static string GenerateSecret()
        {
            var random = new Random();

            const string chars = "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789";

            return new string(Enumerable.Repeat(chars, 8)
              .Select(s => s[random.Next(s.Length)]).ToArray());
        }
    }
}


//Mobile App
//  ExerciseCoordinator.swift
//  PoseRecognition
//
//  Created by Ihor Biletskiy on 20/05/2020.
//  Copyright © 2020 Ihor Biletskiy. All rights reserved.
//

import Foundation
import UIKit

class ExerciseCoordinator: Coordinatorable {
```

```
    // MARK: - Instance Variables

    var viewController: UIViewController?

    var navigationController: UINavigationController?

    weak var delegate: CoordinatorEventControllable?

    var uuid: String = UUID().uuidString

    var childCoordinators: [Coordinatorable] = []

    // MARK: - Life Cycle

    init() {
        configure()
    }

    func start() {
        delegate?.willStart(from: self)

        delegate?.didStart(from: self)
    }

    func stop() {
        delegate?.willStop(from: self)

        // Put your code here...

        delegate?.didStop(from: self)
    }

    // MARK: - Configuration

    private func configure() {
        guard let exerciseViewController = Storyboard.exercise.viewController(viewControllerClass:
ExerciseViewController.self) else { return }

        viewController = exerciseViewController

        let viewModel: ExerciseViewModel = ExerciseViewModel()
        viewModel.coordinatorDelegate = self

        exerciseViewController.setViewModel(viewModel)

        if navigationController == nil {
            navigationController = UINavigationController(rootViewController: exerciseViewController)
        }

        navigationController?.tabBarItem = UITabBarItem(title: "Exercise".localized(), image:
#imageLiteral(resourceName: "exercise_inactive"), selectedImage: #imageLiteral(resourceName:
"exercise_active"))
```

```
    }

}

extension ExerciseCoordinator: CoordinatorEventControllable {

    func willStart(from: Coordinatorable) {

    }

    func didStart(from: Coordinatorable) {

    }

    func willStop(from: Coordinatorable) {

    }

    func didStop(from: Coordinatorable) {
        removeChildCoordinator(from)
    }

}

extension ExerciseCoordinator: CoordinatorChildControllable {

}

extension ExerciseCoordinator: CoordinatorEquatable {

}

extension ExerciseCoordinator: ExerciseViewModelCoordinatorDelegate {

    func processToApparatusDetails(_ apparatus: Apparatus) {
        guard let apparatusDetailsViewController = Storyboard.exercise.viewController(viewControllerClass:
ApparatusViewController.self) else { return }

        viewController = apparatusDetailsViewController

        let viewModel: ApparatusViewModel = ApparatusViewModel(apparatus: apparatus)
        viewModel.coordinatorDelegate = self

        apparatusDetailsViewController.setViewModel(viewModel)

        navigationController?.pushViewController(apparatusDetailsViewController, animated: true)
    }
//    func processChangePassword() {
//        guard let changePasswordViewController = Storyboard.exercise.viewController(viewControllerClass:
ChangePasswordViewController.self) else { return }
//
//        viewController = changePasswordViewController
//
```

```
//      let viewModel: ChangePasswordViewModel = ChangePasswordViewModel()
//      viewModel.coordinatorDelegate = self
//
//      changePasswordViewController.setViewModel(viewModel)
//
//      navigationController?.pushViewController(changePasswordViewController, animated: true)
//  }

}

extension ExerciseCoordinator: ApparatusViewModelCoordinatorDelegate {
```

ДОДАТОК Б

(довідковий)

Слайди презентації

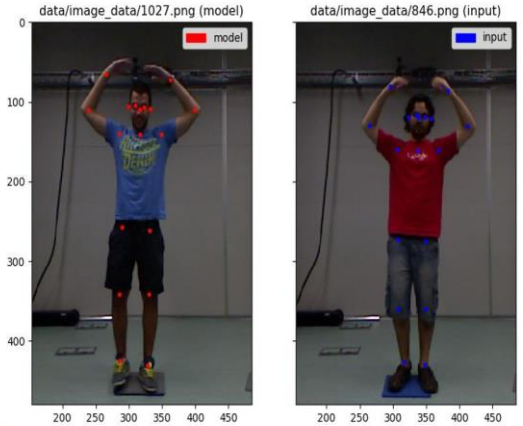
Атестаційна робота магістра

Дослідження методів розпізнавання зображень для розробки програмної системи визначення тіла людини з різних ракурсів

Керівник:
проф. Білоус Н. В.

Виконав:
студент групи
ІПЗм-18-4
Білецький І. С.

Визначення поз



Задача визначення тіла людини на зображенні стосовно її пози полягає у знаходженні та ідентифікації ключових точок - суглобів. Знаходження їхніх координат дозволяє визначити їх на зображенні, а ідентифікація, яка з цих точок якому суглобу відповідає, дозволяє поєднати їх належним чином для побудови “скелета” людини.

2

Актуальність

- Існує безліч систем тренінгу, але в принципі, не існує єдиного типу тренінгу для всіх - для кожного потрібен індивідуальний підхід. Єдиний тип тренінгу, який враховує всі ваші особливості фігури, який в залежності від вашого типу запропонує вам комплекс вправ, що включає в себе заняття з вагами і аеробіку - це фітнес. поняття. Фітнес» вже міцно увійшло в наше життя, створені фітнес-клуби, видаються журнали, проводяться фітнес турніри. Фітнес став способом життя, який веде до фізичного і ментального здоров'я людини. Однак, важливо правильне виконання всіх фізичних вправ.
- Для цього потрібен або контроль тренера, професіонала, або самоконтроль за допомогою інтелектуального додатку, який перевіряє правильність виконання, порівнюючи з шаблоном вправи.
- Актуальним на даний момент є визначення правильності виконання фізичних вправ та рухів людиною для подальшого порівняння з правильним шаблоном їх виконання.

3

Існуючи рішення

- Microsoft Kinect
- Фітнес-трекери / смарт-годинник
- Сенсорні датчики на тренажерах
- Розумний одяг

4

Постановка задачі

- провести аналіз існуючих алгоритмів та бібліотек у предметній області;
- розробити систему розпізнавання пози людини у режимі реального часу;
- розробити серверну частину та адміністративну панель системи, за допомогою яких є можливість проводити визначення тіла людини з різних ракурсів;
- розробити API для мобільного додатку клієнтів камери та полегшення використання бібліотеки OpenPose;
- провести дослідження ефективності розроблених алгоритмів.

5

Вимоги до системи

Система повинна надавати користувачу наступний функціонал:

1. Можливість додавати камеру, або використовувати стаціонарну
2. Можливість вибіру залу, тренажеру або пристрою
3. Можливість реєстрації та авторизації в системі
4. Можливість отримувати інформацію про виділення ключових точок з різних ракурсів

6

Аналіз існуючих бібліотек для визначення поз

Було проаналізовано дві бібліотеки з відкритим початковим кодом, що надають можливість визначення пози на зображенні:

1. OpenPose
2. DeepCut

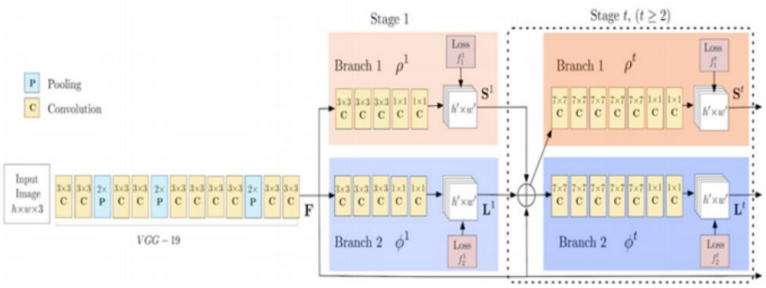
7

Методи порівняння поз

1. Метод, що базуються на підході «знизу вгору» (OpenPose, DeepCut)
2. Метод, що базуються на підході «зверху вниз», т.н. Pose estimation (RMPE, Alpha Pose)
3. Нейромережеві методи (R-CNN, Mask R-CNN)

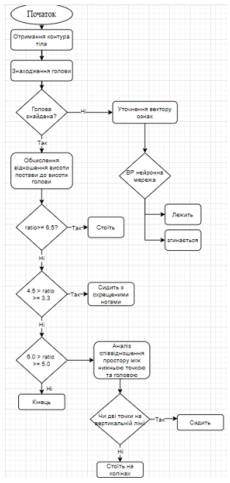
8

Архітектура моделі OpenPose



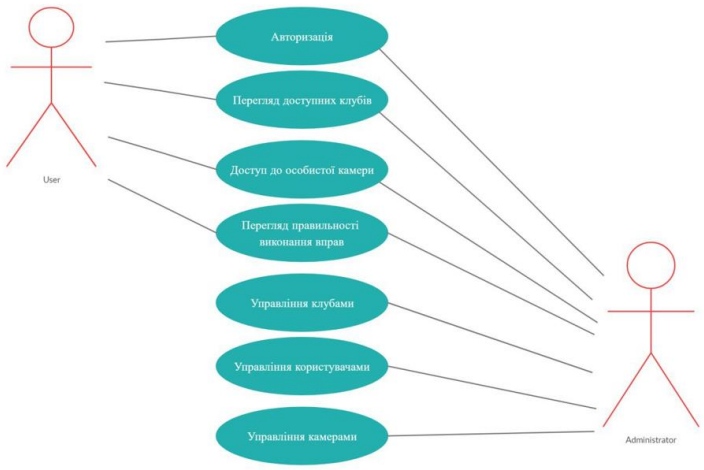
9

Гібридний підхід з використанням Kinect



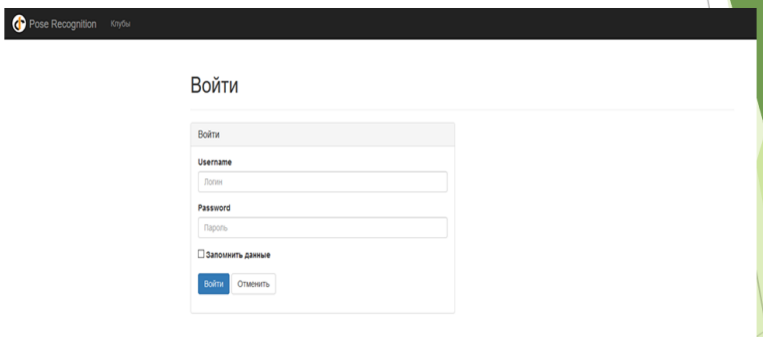
10

UseCase діаграма



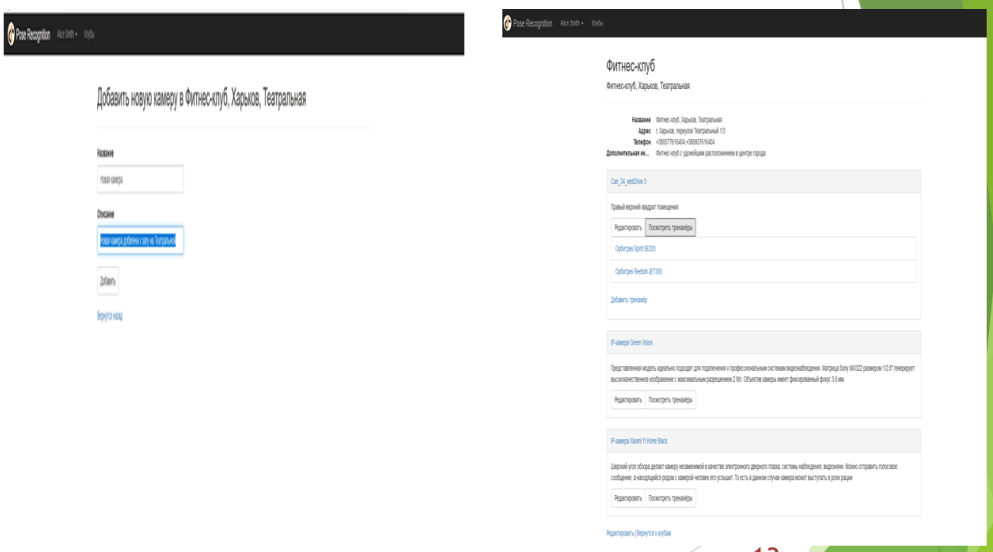
11

Адміністративна панель. Вікно авторизації



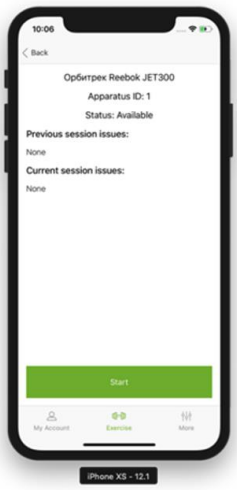
12

Адміністративна панель. Додавання камери.



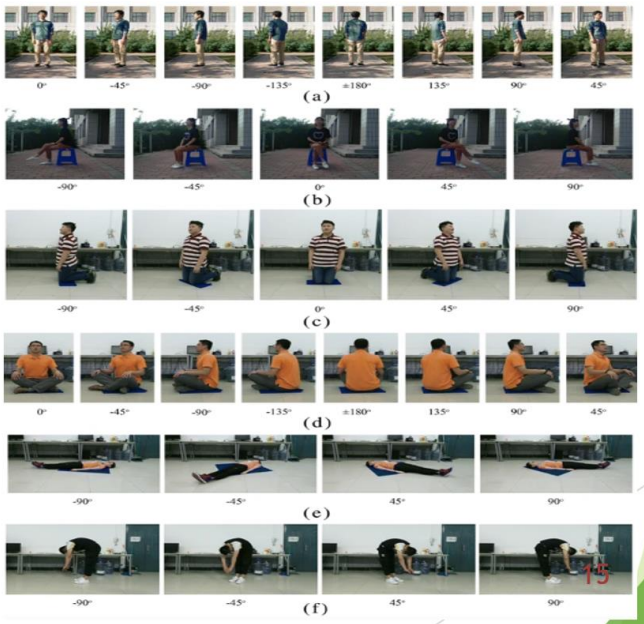
13

Мобільний додаток



14

Постава людей



15

Висновки

Результатами виконання атестаційної роботи є:

1. Дослідження методів визначення тіла людини стосовно пози в контексті аналізу потокового відео.
2. Дослідження ефективності досліджених методів.
3. Реалізація та аналіз працездатності системи визначення пози людини PoseRecognition

16