OpenCV AI Competition

PROJECT TITLE: PERSONAL YOGA TRAINER
TEAM: ULYANOVSK PIONEERS

PROJECT RESULTS:

The project was aimed at developing a small OAK-D-based device prototype that would act as a personal fitness/yoga trainer. The functions declared for the device were:

- Showing an original yoga video course demo via HDMI containing both video instructions and a detailed audio description for visually impaired persons;
- Human position recognition and assessment of its correctness;
- Voice prompts to move body parts in the desired direction until the position becomes correct;
- Overall assessment of the success of the yoga class.

During the project, it became clear that, for the reasons of user's convenience, it was more preferable to display a static pose image during the lesson instead of a video. The other results of the project correspond highly to the above tasks.

Technically, the Personal Yoga Trainer consists of two main parts: hardware and software. Below is a consistent description of both.

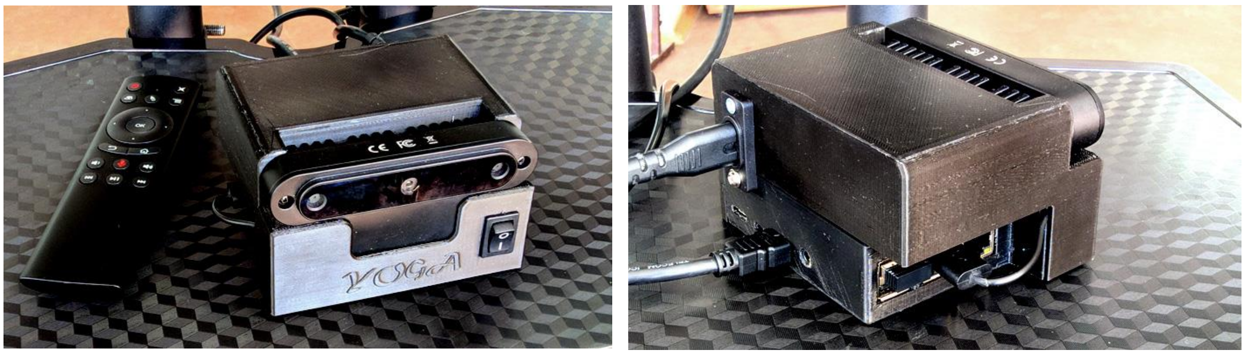
Figure 1. The 'Personal Yoga Trainer' device. The front and rear views.

1. The hardware part comprises the following components (see Fig. 1):
   a) an OAK-D camera;
   b) a Raspberry Pi 4 computer with 4 GB RAM;
   c) a 25W power unit with a power cord;
   d) a 110x90x65 mm case with a cooler and a power button. The fan sucks in air from the side of the case and cools the CPU, the power source and the camera sequentially. Eventually, air is blown up along the camera's heatsink;
   e) a remote control. The remote connects to the Raspberry Pi computer via the Bluetooth protocol.

The case was designed in Kompas-3D (see Fig. 2), then 3D-printed on an Anycubic Linear Plus 3D-printer using PETG and PLA plastics. The case consists of two parts. In order to disassemble the device, one needs to unscrew four screws.

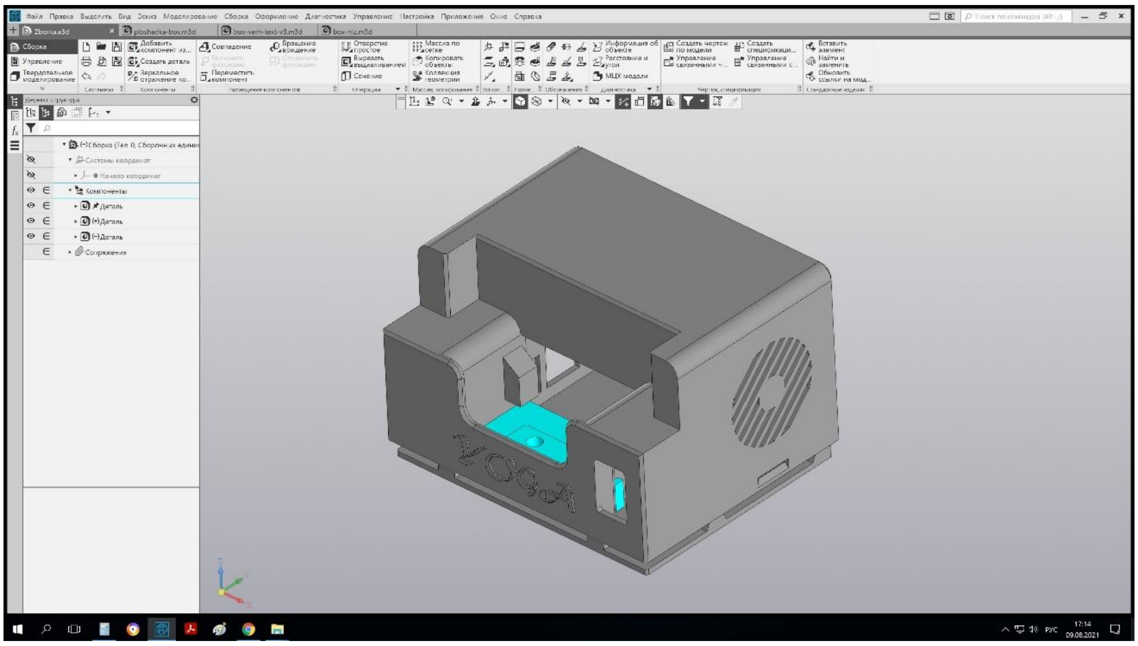
Figure 2. The design of the case in Kompas-3D.

One of the design features is a quick release connector for an OAK-D camera which could be helpful if necessary to use the system unit and the camera separately. With the power off, the camera can be pulled up to detach.

The case has openings for all necessary connectors of the Raspberry Pi computer, including a slit for the microSD card.

2. The software part was implemented in Python and C++.

   The main functions of the C++ software are launching at startup, processing the data from the remote control, and providing a GUI (implemented using Qt5) with the following screens:

   a) The Home screen containing the icons of several yoga poses (see Fig. 3). A user may choose the necessary pose to start training using a remote control.

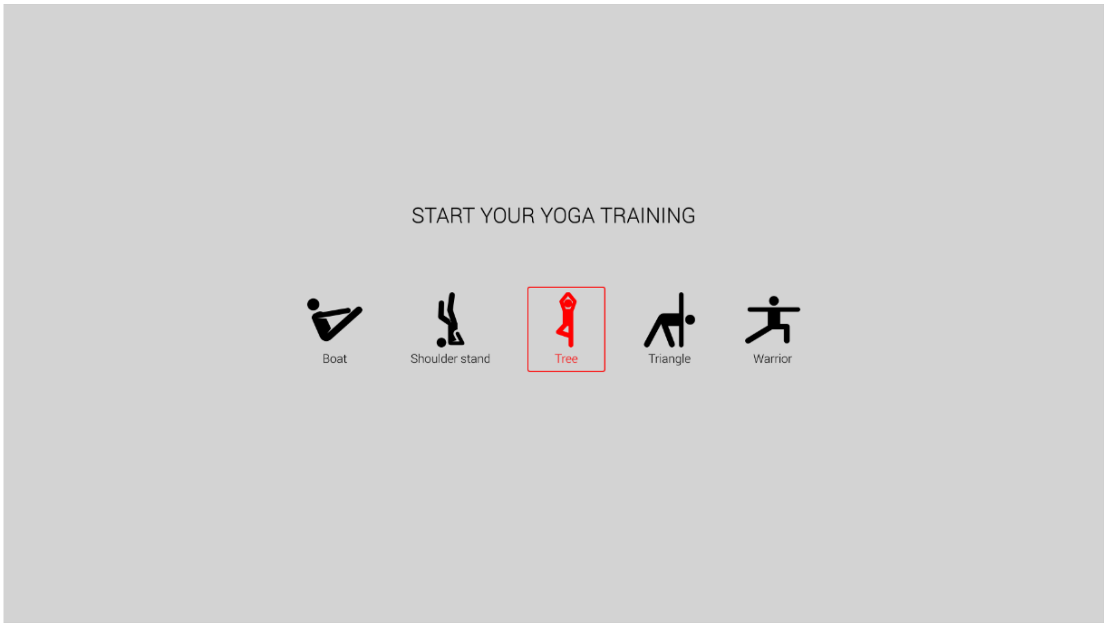

Figure 3. The Home screen.

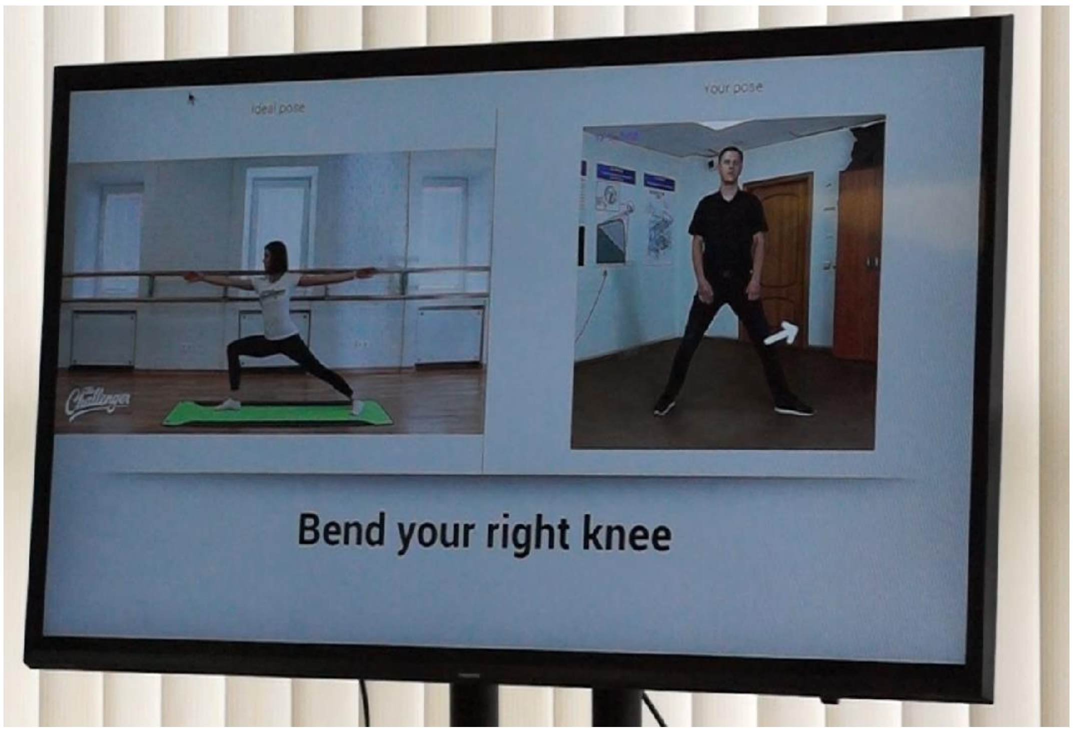

Figure 4. The Pose screen.

b) The Pose screen containing three windows (see Fig. 4):

- a window illustrating the correct pose (a photo);
- a window displaying the current pose of the user from the camera;
- a window displaying information about the necessary actions in text form.

The python script runs the inference model and contains the following:

a) an inference model call which yields the key points for the current pose. The BlazePose model adapted for DepthAI (https://github.com/philipjhc/depthai_blazepose) is at the heart of pose recognition;
b) an algorithm to calculate angles between the key points;
c) a list of correct angles between human limbs for each pose;
d) a list of angle tolerances for each angle and each pose. The tolerances are different for each angle, and they have been chosen as a result of measurements of a series of correct poses;
e) a list of corrective phrases to be displayed for each pose;
f) a vocalization procedure call.

The python script controls the body position through the key angles between links. When several angles are beyond the required tolerance, the system consistently advises actions to be taken.

Let’s consider the sequence of actions using an example of the Warrior Pose:

1) Setting the feet. The required width is approximately two shins;
2) Turning the right foot outward;
3) Bending the right knee;
4) Tilting the body forward;

5) Raising the right arm;

6) Raising the left arm;

7) Control of straightness of arms.

The suggested actions are displayed in a separate window in text form. Besides, the system vocalizes them by means of RHVoice, a free and open-source speech synthesizer (https://github.com/RHVoice/RHVoice). The benefit of using RHVoice is that it supports both Russian and English.

The system would return to the previous item(s) when it notices that, while, say, raising the right arm, the user has lost his control over the right knee.

Eventually, when all the angles between limbs are within the given tolerances, the system announces the overall success. The user may then press the Home button on the remote and choose the next lesson.

THE TEAM:

Victor Prikhodko – Team Leader
Aleksey Tregubov – Lead Developer
Pavel Guskov – Design Developer
Evgeny Chavkin – Modeler
Ivan Guschin – Programmer
Anton Leschinskiy – Electronics engineer
Vladimir Levschanov – Engineer
Dmitry Lavygin – Programmer
Pavel Kapustin – AI Developer
Valeriy Kozhevnikov – AI Developer

Этап 1
Разработка / оцифровка КД
SIEMENS NX
Этап 2
Разработка управляющего устройства
Этап 3
Подготовка URDF-моделей объектов
Этап 4
Разработка управляющего программного обеспечения
ROS
MERRELL

Figure 5. Part of our team: Evgeny Chavkin, Vladimir Levschanov, Ivan Guschin, Aleksey Tregubov, Victor Prikhodko, Anton Leschinskiy.

SPECIAL THANKS TO:

Artyom Muralyov
Kseniya Nuzhina
Valeriy Sapunov