Computer Vision Competition WS2018/2019

David Scherfgen
Katharina Stollenwerk
Christoph Pomrehn
Anton Sigitov
Rainer Herpers

October 24, 2018

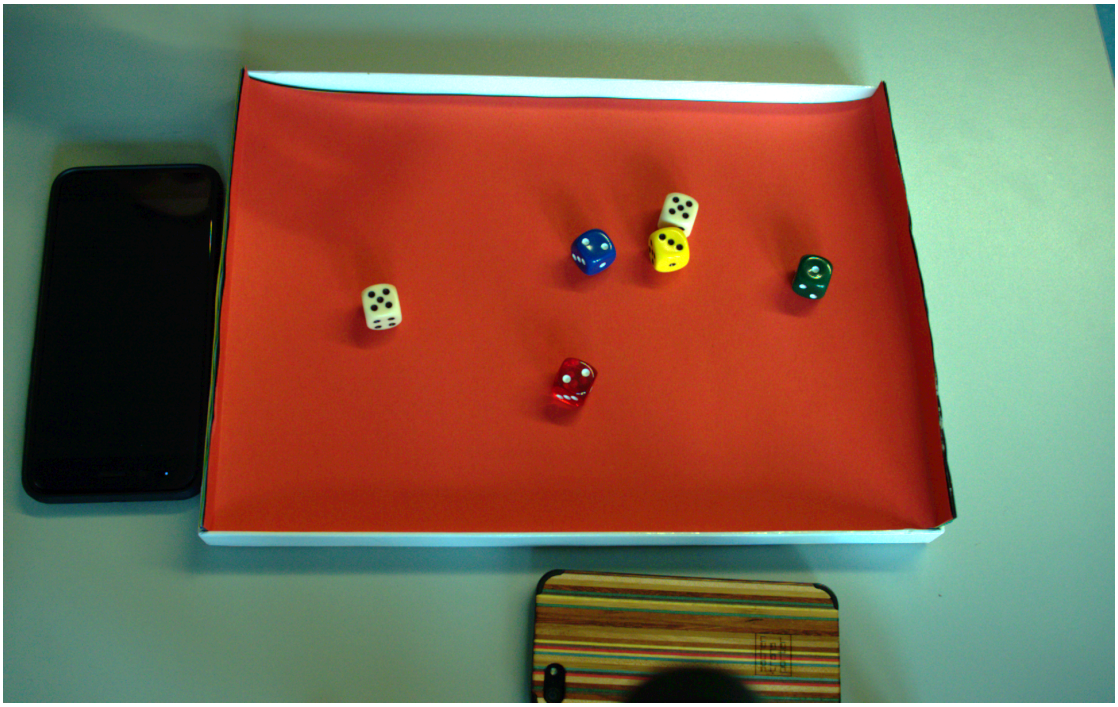

Figure 1: Example video frame. Dice of different materials and colors have been rolled on a uniformly colored background. It is your task to write a program that detects the dice and reports the position and value of each.

1 Introduction

In previous semesters, students could define their own Computer Vision projects. This time, we will try something new: Everybody will have to solve the same task, and there will be a competition between the students.

You will be given video data showing dice being rolled. Your task is to detect the dice and report the position and value (the number of dots shown by the top face) of each. See Figure 1 for an example video frame.

Your program will compete with the other students' programs. Depending on your performance, you will receive additional bonus points that will help you achieve a better grade in the exam.

2 Downloadable packages

> **Important:**
> Most instructions in this document were written targeting Windows users. Unix users should keep this in mind and make sure to also read section 11.

We provide downloadable packages that contain "everything" you need to get started. You find them on the course's LEA page at `https://lea.hochschule-bonn-rhein-sieg.de/goto.php?target=crs_507471&client_id=db_040811`.

The **software package** contains:

- OpenCV 3.4.3 compiled for Visual C++ 2017 (64-bit) and Python on Windows (2.7 and 3.6, 64-bit each)
- basic example programs for OpenCV in C++ and Python
- self-explanatory templates that you can use to start developing your dice detection program in C++ or Python
- the "Evaluation" program to test your dice detection program (see section 8)
- the "VideoRecorder" program to record your own videos (see section 10) on the lab PC (see section 9)

Visual Studio 2017, which is required to open the `CVC1819.sln` file, can be downloaded for free from `https://visualstudio.microsoft.com/thank-you-downloading-visual-studio/?sku=Community&rel=15`.

> **Important:**
> After extracting the software package, first run the batch file `CopyDLLs.bat`, otherwise the programs will not run due to missing DLL files.

The first **video package**, provided as a separate download, contains a data set of 20 example videos with ground truth (see section 7). After downloading and extracting the software package, extract the files within the video package into the directory `Data\Videos\Evaluation`.

To get started, open the Visual Studio 2017 solution file `CVC1819.sln` and run the "Evaluation" or the "Example" (a basic OpenCV example) project. If you're new to Visual Studio: Right-click a project in the "Solution Explorer" window and click "Select as StartUp Project", then press "F5" to start with debugging or "Ctrl+F5" to start without debugging.

3 Input data

Your task is to implement (using programming language and libraries of your choice, but we provide program templates for C++ and Python only) a program that accepts, via command line argument, a path to a video file showing a dice roll. For the exact command line interface that your program has to implement, please refer to the program templates (in C++ and Python) that are provided in the software package.

Some useful information regarding the input data:

- Video resolution is 1936×1216 px.
- The number of dice in a single video is between 1 and 6.
- Dice are regular 6-sided dice that vary in material and color, but not in size.
- Camera perspective and lens vary between videos, but the view is always more or less top-down (i. e. the camera won't be upside down).
- Lighting and camera settings (aperture, focus, exposure time, sensor gain) vary between videos.

- Dice are rolled inside a box that has a uniform background color, which however varies between videos.

- Dice may not all be rolled at the same time. However, after the roll is complete, there is a reasonable timespan during which the dice don't move, before they are taken away at the end of the video.

- Additional objects may be present in the box or rolled together with the dice, however they are distinguishable from them.

- Dice or other objects will not be on top of each other, however partial occlusion of side faces due to perspective can occur.

4 Expected output data

For each die that your program detects, it has to output:

- its value (the number of dots shown by the top face)

- an (x, y) pixel position anywhere within the **visible** part of the die after it has stopped moving – the safest choice is the center of its top face, since it is usually unoccluded and most certainly correctly labeled in the ground truth

Additionally, your program has to output the zero-based index of the video frame that should be displayed along with the detection results during evaluation, i. e. (one of) the frame(s) that your program has chosen for the final dice detection.

Once again, see the program templates in the software package for technical details on how the result is expected to be output.

5 Time constraint

In order to prevent programs from taking too much time during evaluation, we impose a maximum running time of 15 s per video. If your program is still running after that time, it will be terminated, and if it hasn't output anything by then, you will receive 0 points for that video.

Note that your program is not allowed to keep running between processing videos. A

new process will be launched for each video to be processed (we know that's unrealistic...). Therefore, try to avoid time-expensive initialization tasks.

> **Important:**
> Make sure to compile your project in "Release" mode (not "Debug") when submitting the executable for the competition. In "Release" mode, your program will run much faster. Use "Debug" mode during development.

6 Scoring

For each video, scores are calculated according to the following scheme (see Figure 2 for an example), and then added together to produce your final score:

- You "get" -1 (negative!) point for **every** detection output by your program, no matter if it is correct or not, i. e. if your program outputs n detected dice, you start with $-n$ points.
- Each real die in the video that is hit by any of your detections (in terms of pixel position) gives you $+2$ points, regardless of its value.
- If the value of the die is also correctly detected, you receive an extra $+1$ point.

This has the following implications:

- For a video with n dice, you can receive a maximum of $2n$ points if you detect all dice with their correct values.
- If you don't output any detections, you will receive 0 points for that video.
- Negative scores are possible, so you should only output a detection when you're certain enough that it's correct.

The programs will be ranked based on the final score.

7 Data sets

At first, you are provided with 20 videos including ground truth so that you can start developing and testing. These videos are similar to those that will be used for the

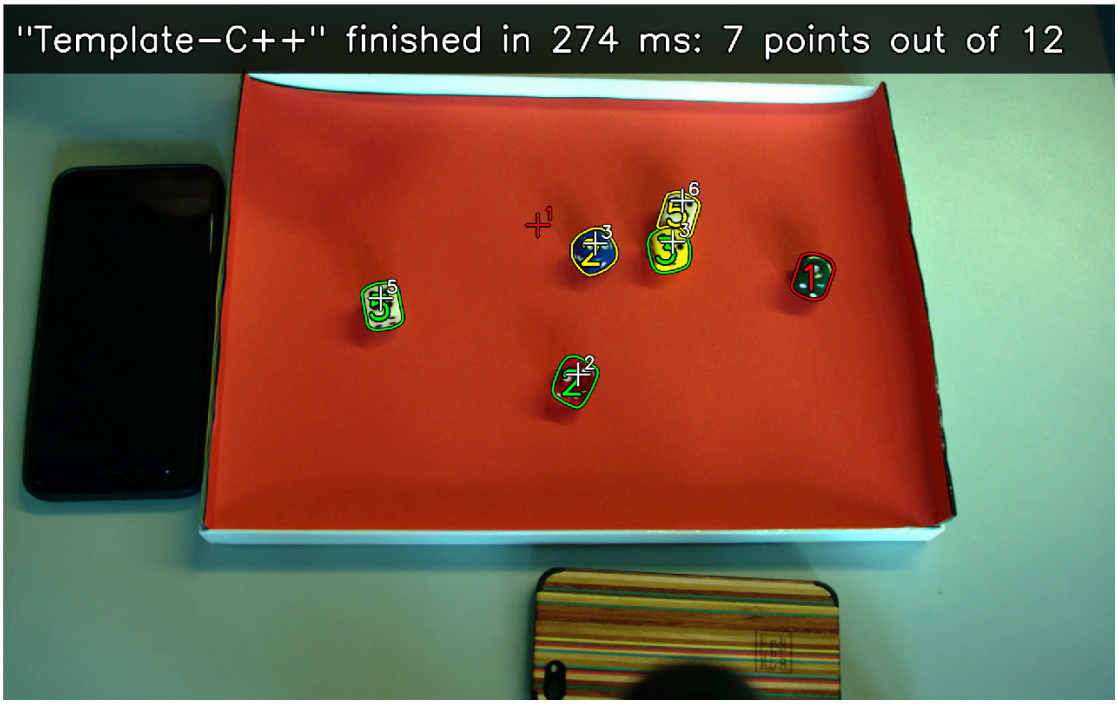

Figure 2: Example screenshot from the evaluation program. The program has output 6 detections (one of them being completely off), so the score starts at -6 points. 5 of the real dice have been detected, yielding $+10$ points. 3 of them were assigned the correct value, resulting in an extra $+3$ points. The total score is therefore $+7$ points. The maximum score for this video is 12 points.

final evaluation. Ground truth has been manually created by the Computer Vision staff by labeling each die with a polygon and annotating its value (see Figure 3). The C++ program that will be used for evaluation is included in the software package (see section 8).

Later during the course, you will be given another 20 videos with ground truth for performing a self-evaluation and further improving your program.

For the final evaluation, we will use a "secret" data set of at least 60 new videos.

8 Evaluation program

You can use the evaluation program contained in the software package to test your program and see what score it would achieve. This is also useful for comparing different variations of your program, e. g. different algorithms or parameters.

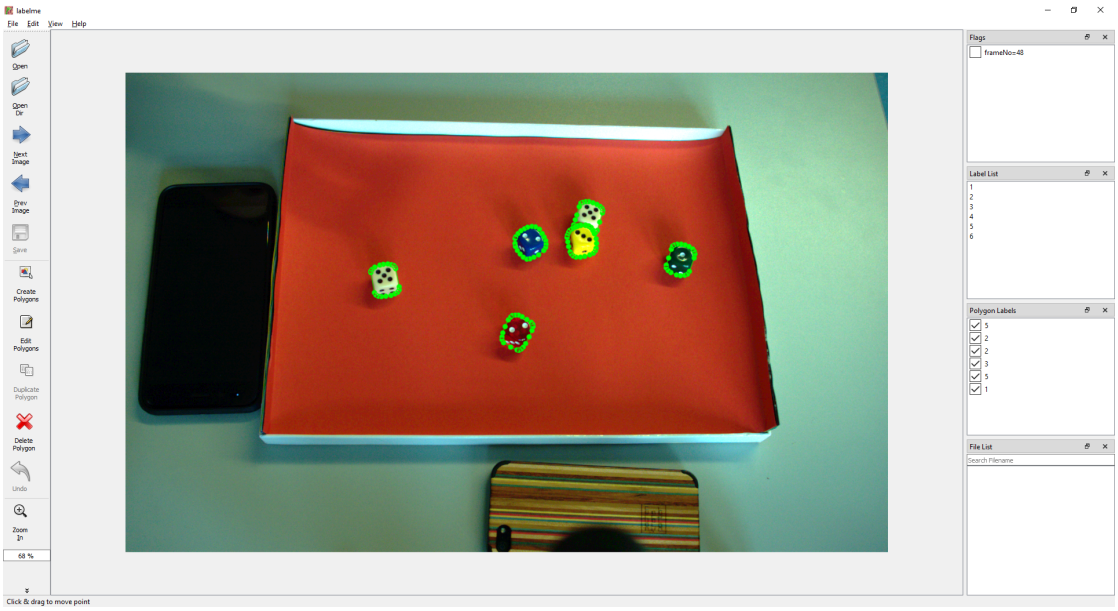

Figure 3: Example ground truth from the video package. Dice have been labeled/annotated with polygons using the "labelme" program that is included in the software package.

The evaluation program expects the following command line arguments (make sure to place `"quotation marks"` around arguments containing spaces):

- For each program to be tested:
  - the name to be displayed during evaluation, e. g. your name (avoid characters that aren't allowed to appear inside filenames)
  - the path to its executable file
- the directory that will be searched for video and ground truth data
- the directory where the evaluation results will be stored

When launched from within Visual Studio, the two program templates (C++ and Python) are tested using the data in `Data\Videos\Evaluation`, and the evaluation results will be written into the `Results` directory. The command line then is, assuming the current directory is the one where the `CVC1819.sln` file is located:

```
x64\Release\Evaluation.exe Template-C++ x64\Release\Template.exe
                           Template-Python Projects\Template-Python\Template.py
                           Data\Videos\Evaluation
                           Results
```

The same command is also executed by the included batch file `RunEvaluation.bat`.

> **Important:**
> When your program is launched by the evaluation program, the current working directory might not be the one that you expect. This is important to be aware of when your program loads additional files (e. g. a trained CNN model).

See Figure 2 for an example screenshot from the program. The current ranking will be shown in a separate window.

9 Using the lab PC

The final evaluation will be performed on our lab PC (labeled "Computer Vision") in the Computer Vision Lab (room C065, see Figure 4). It runs Windows 10 (64-bit) and has Visual Studio 2017, Python 3.6 and the software/video package installed. It is equipped with 16 GiB of RAM.

> **Important:**
> It is your own responsibility to make sure that your program runs properly on the lab PC and when being called from within the evaluation program! A program that crashes constantly or always exceeds the time limit will result in 0 points.

Your names will be added to the list of people that are allowed to borrow a key for C065 at the front desk. Since we have only one lab PC for this project, please make use of the time slot reservation list in order to avoid conflicts.

A local user account with limited privileges will be created for each of you. Your username will be the same as the one from your university account, e. g. "jdoe2s". The initial passwords will be sent to your university e-mail address.

If you need to install additional software for your program to run, please try to find a way to install it locally under your own user account. If administrator rights are absolutely necessary to install something, then you can ask one of the staff members to assist you.

Figure 4: The lab PC in room C065 where the final evaluation will be done. Make sure that your program runs properly on this machine.

The software/video packages are installed for everyone to be accessed (read-only) in `V:\Packages`. You can copy the software from there into your own user directory, **but please don't copy the video files (in order to save disk space)**. Instead, test your program against the video files in `V:\Packages\VideoPackage1` by adjusting the path in the call to the evaluation program.

10 Creating additional data

An industrial grade camera is attached to the lab PC. This is the camera that has been used to record the videos. The dice seen in the videos are also provided[1] as well as various background paper sheets and light sources. If you wish to record your own

[1] **WARNING!** Not suitable for children under 36 months. Risk of choking due to small parts that may be swallowed!

videos in order to have more data to work on (which we highly recommend), you can do so using the program "VideoRecorder" from the software package. Launch it from within Visual Studio. This is how you record a video:

- Press "R" to start recording.
- Roll the dice. Once they don't move any more, press "S" to save a screenshot.
- Press "R" to stop recording.
- The program "labelme" will be started so that you can create the ground truth based on the screenshot. We recommend to do the labeling at a later time. In order to be able to save the labeling file (JSON file format), you have to create at least one "dummy" polygon. If you want to label the data later, you find the software in `Software\labelme-3.3.6`, where a `.bat` file is provided to start the program with useful default parameters.
- The video, the saved screenshot and the labeling file will end up in the directory `Data\Videos\Recorded`.
- Repeat the procedure to record the next video (you don't have to restart the program).

11 Unix users

Important:
After extracting the software package, you might have to run the `SetExecutableBits.sh` shell script by opening a terminal and typing "`bash SetExecutableBits.sh`". This will make all shell scripts and the Python scripts executable.

Important:
You're free to develop your program under a non-Windows operating system, however please keep in mind that the final evaluation of all programs will be done under Windows. Make sure that your program runs on the lab PC.

Unless you intend to compile the latest OpenCV version yourself, you should first check if your distribution's package repository provides some version of OpenCV (although it may not be the most current one). This makes installation much easier. For example, installing OpenCV and its Python 3 module under Ubuntu can be done by opening a terminal and typing:

```
sudo apt-get install libopencv-dev python3-opencv
```

The software package provides a shell script named `Compile.sh` that compiles the OpenCV example program, the competition C++ template and the evaluation program. It relies on a recent version of GCC and "pkg-config" being installed, also the evaluation program makes use of the `timeout` utility to limit execution time. Under Ubuntu, GCC and "pkg-config" can be installed as follows:

```
sudo apt-get install build-essential pkg-config
```

The shell script `RunEvaluation.sh` runs the compiled C++ template and the Python template against the evaluation videos found in `Data/Videos/Evaluation`, writing the evaluation results into the `Results` directory.

Good luck, and may the dice be with you!