Numlog Guide

Author: Kim Miikki
Date: 20.8.2021

1 Introduction

Numlog is an optical character recognition (OCR) program which can be used to digitize seven segment digits to numbers. Typical applications are digitization of LCD or LED display numbers. The program has four different modes: single image OCR, capture image, real-time logger and batch OCR for several images.

2 System Requirements

Operating System: Raspberry Pi OS or Linux

Python 3 with Matplotlib, Numpy and OpenCV.

RPI camera module or an OpenCV video capture class object

3 OCR Method

This program is built on *ssocr.py* (https://github.com/jiweibo/SSOCR) and licensed under GPLv3. The idea is to detect LCD numbers when they are displayed in a seven-segment style. Several image operations are done in order to get clean numbers for the OCR. Levels and inverse options has been added to the program in order to make OCR more reliable. This program is ROI aware, and it is strongly recommend to use this functionality.

The workflow is summarized in the following list:

1. Compose and capture an initial image of the LCD digits
2. Select ROI with roi-picture.py
3. Do the preliminary OCR in capture mode by adjusting
    1. lighting and capture angle
    2. focusing ↔ de-focusing
    3. control arguments (inverse mode, threshold, levels etc.)
4. Capture digits in real-time or from already captured images in batch mode

4 Program Usage

Numlog has several arguments which can be listed with the following command:

```
$ numlog.py -h
usage: numlog.py [-h] [-b] [-c] [-r] [-s] [-d] [-i] [-hw HW_RATIO]
                 [-a TILT_ANGLE] [-t THRESHOLD] [-n DIGITS] [-p DECIMALS]
                 [-bp BP] [-wp WP] [-gamma GAMMA]
                 [image_path]

positional arguments:
  image_path            path to image

optional arguments:
  -h, --help            show this help message and exit
  -b, --batch_ocr       batch mode OCR
  -c, --capture_image   capture an image
  -r, --real_time_logger
                        real-time data logger
  -s, --show_image      whether to show image
  -d, --is_debug        True or False
  -i, --inverse_display
                        inverse display mode (bright digits)
  -hw HW_RATIO, --hw_ratio HW_RATIO

                        H/W ratio
  -a TILT_ANGLE, --tilt_angle TILT_ANGLE
                        tilt angle
  -t THRESHOLD, --threshold THRESHOLD
                        threshold (0-255)
  -n DIGITS, --digits DIGITS
                        digits
  -p DECIMALS, --decimals DECIMALS
                        decimal point position (digits >= pos >= 0)
  -bp BP                black point (0...255) as integer
  -wp WP                white point (0...255) as integer
  -gamma GAMMA          gamma (default=1.0) as float
```

A positional argument (path or filename in current directory) is used for single image OCR. Arguments marked with blue color (-b, -c and -r) executes the script in its main operation modes. Only one of these arguments can be used simultaneously. Arguments -s and -d are useful for debugging and for OCR parameters selection. Pink colored arguments (-i, -hw, -a and -t) are OCR parameters. If digits or decimals are specified (green arguments), then OCR is forced to be in selected format. If the length of detected digits differs from the specified digits option, the value will be discarded. The last group of arguments (red color) adjusts the levels operation.

Normal display is here considered as dark digits on light LCD. Hence, inverse color scheme is light digits on dark background.

4.1 Basic Operations

The program usage is demonstrated in this chapter by digitizing temperatures on a multimeter LCD display.

First task is to set up a digitization environment (Fig. 1). The LCD display and the camera should be firmly on place. The lightning should be adjusted, and optimized with the *numlog.py*.

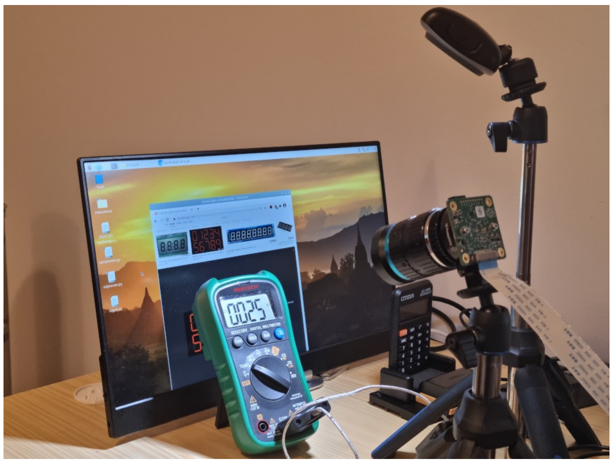


Figure 1. A composition of a digitization environment.

In the second phase the OCR program is launched:

```
$ numlog.py -c
Seven segment LCD number logger

Current directory:
/home/pi/python/20210819-numlog

ROI file not found!
Capture an image:        SPACE
Toggle processed image: T
OCR digits:              O
Exit preview:            ESC
```

A live OCR session is started when pressing T button. The OCR can be turned off by pressing O or T. Both keys are used in toggle principle. To exit the program ESC key has to be pressed. In Figure 2 the effect of adding more light to the system can be easily seen, especially on the last threshold window. A snapshot of a live OCR can be seen in Figure 3.

An important step is to capture an image from the LCD. The image is needed for a region of interest (ROI) selection. When pressing SPACE bar, an image is captured and the execution of the program is ended.

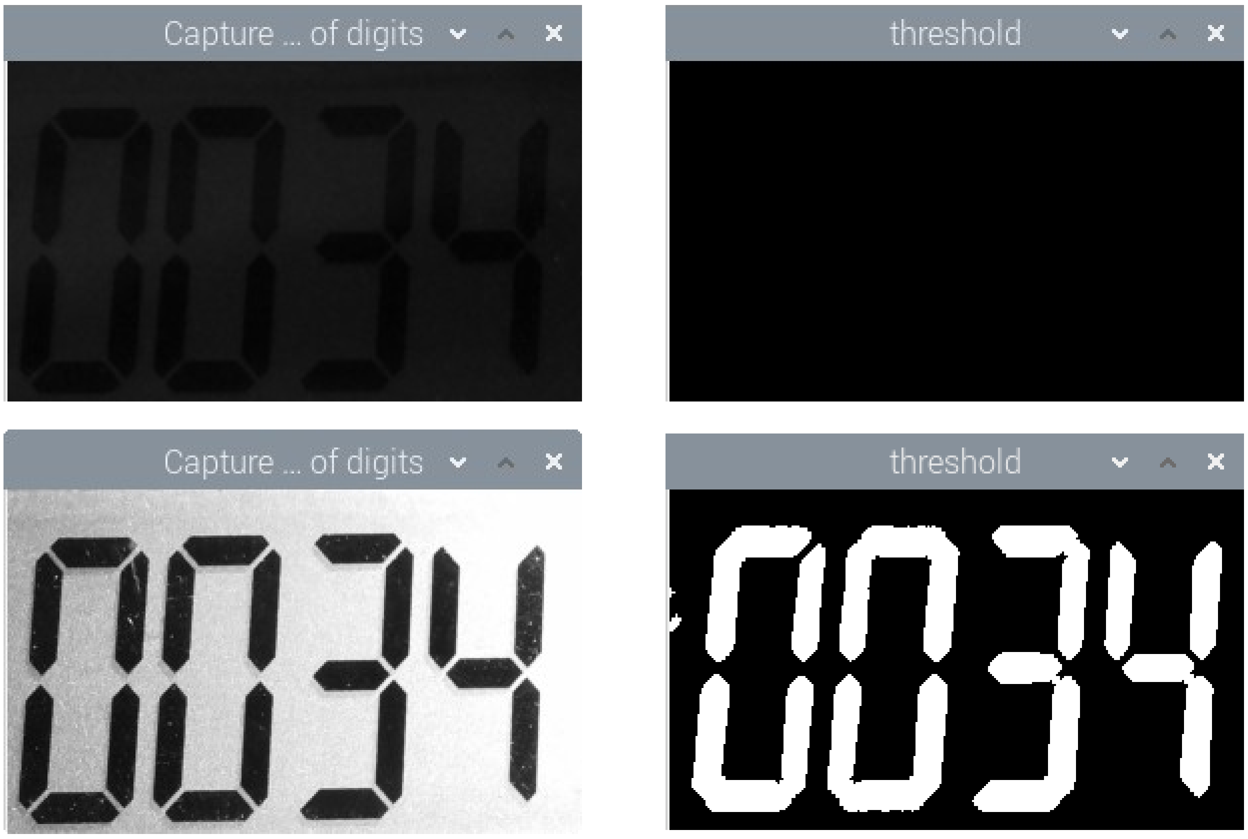
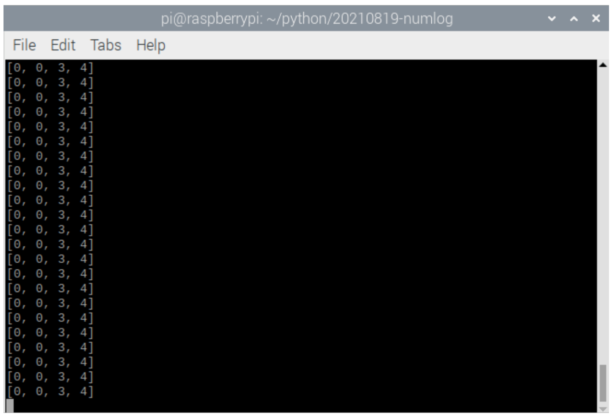


Figure 2. The first row of pictures is an indication of poor lightning environment. The threshold window is lacking digits, and the program cannot perform OCR without them. The situation is corrected by adding an external LED light source. Now the digits are visible on the threshold window.



```
[0, 0, 3, 4]
[0, 0, 3, 4]
[0, 0, 3, 4]
[0, 0, 3, 4]
[0, 0, 3, 4]
[0, 0, 3, 4]
[0, 0, 3, 4]
[0, 0, 3, 4]
[0, 0, 3, 4]
[0, 0, 3, 4]
[0, 0, 3, 4]
[0, 0, 3, 4]
[0, 0, 3, 4]
[0, 0, 3, 4]
[0, 0, 3, 4]
[0, 0, 3, 4]
[0, 0, 3, 4]
[0, 0, 3, 4]
[0, 0, 3, 4]
[0, 0, 3, 4]
[0, 0, 3, 4]
[0, 0, 3, 4]
[0, 0, 3, 4]
```

Figure 3. Numlog OCR function in live mode.

When passing the name of the image to *numlog.py* an OCR operation is executed:

```
$ numlog.py lcd-with-light.png
Seven segment LCD number logger

Current directory:
/home/pi/python/20210819-numlog

ROI file found in current directory: 0.3469,0.5125,0.4484,0.3583
Digits: [0, 0, 3, 4]
Value:  0034
```

When the initial image has been captured and ROI selected, a real-time or batch OCR can be done. These steps are demonstrated in sections 4.2 and 4.3.

4.2 Real-Time OCR From a Multimeter Display

Numlog.py can be started in real-time mode with the argument -r. The default y axes text can be changed by entering a new text to the first question. Default text is ‘Value’. This mode requires a camera.

The data can be collected in two modes: interval or data change mode. Minimum interval is 1 s, and if 0 is selected, the OCR is continuous without the interval delay. Interval is not needed in the change mode, which acts as a stopwatch. OCR starts at 0 s and ends when pressing ESC. The data will be recorded only when the number changes, unless logging of all values has been selected.

In the following example, the temperature readings of the multimeter are digitized with *numlog.py*:

```
$ numlog.py -r -t 35
Seven segment LCD number logger

Current directory:
/home/pi/python/20210819-numlog

ROI file found in current directory: 0.3469,0.5125,0.4484,0.3583
Enter y axes text (Default=Value: <Enter>): Temperature (°C)
Use interval (Y/N, Default y: <Enter>):
Default selected: Interval enabled
Select interval: (0...2592000 s, Default=1: <Enter>):
Default value selected: 1
000001: 35
000002: 35
000003: 35
000004: 35
000005: 35
000006: 35
000007: 35
000008: 35
000009: 34
000010: 33
000011: 32
000012: 31
...
```

The experiment was ended by pressing ESC. Three files were created at the end: numlog20210819_171955.csv, numlog20210819_171955.log and numlog20210819_171955.png. The data is stored in a CSV file, OCR parameters in a LOG file and the data is plotted as function of time in a PNG file. The graph for this experiment is shown in Figure 4.

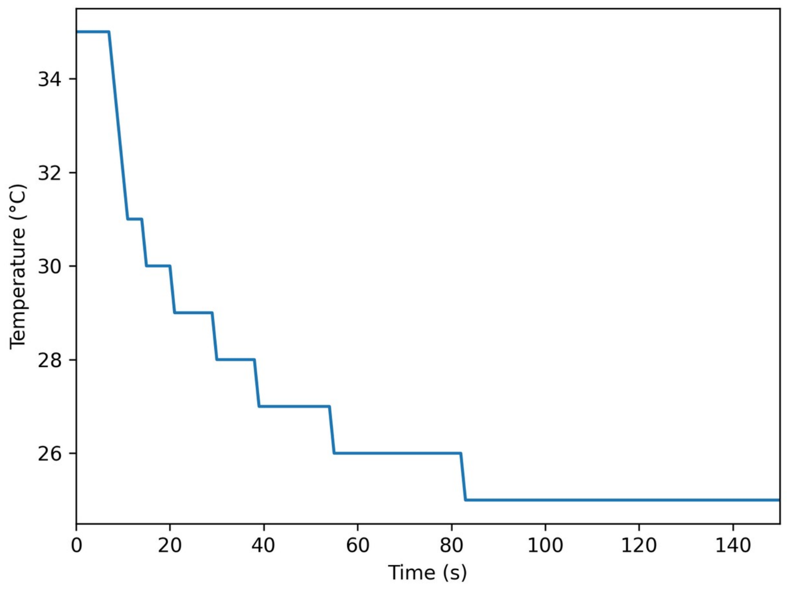


Figure 4. Temperature readings on a multimeter LCD display as function of time. The readings were recorded in real-time logger mode with threshold 35 and 1 s interval.

4.3 Batch Digitization of Numbers in Inverse Mode

For fast changes of display numbers, it is recommended to record a video of the screen or digits, and then to perform OCR in batch mode. This will also give a better control for frame timing (= interval time).

A simple experiment was done from a seven digits displayed on an ordinary monitor. A very short video was recorded with a Raspberry Pi Camera. Thereafter the frames were extracted with *vid2pic.py*. The first frame was used when selecting ROI manually with *roi-piture.py*.

The digits are in this case in inverse mode, therefore -i argument must be used. A ROI picture representing digits are shown in Figure 5.

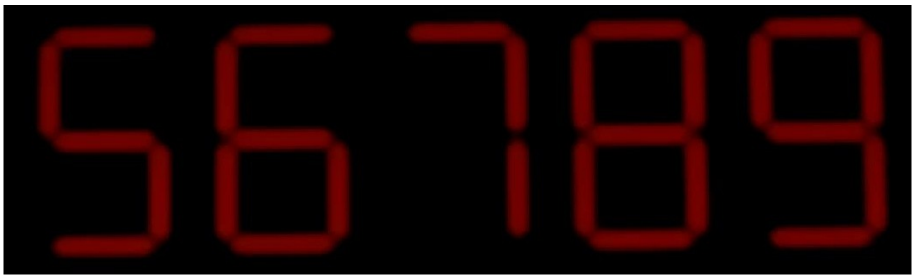


Figure 5. ROI area from a video frame.

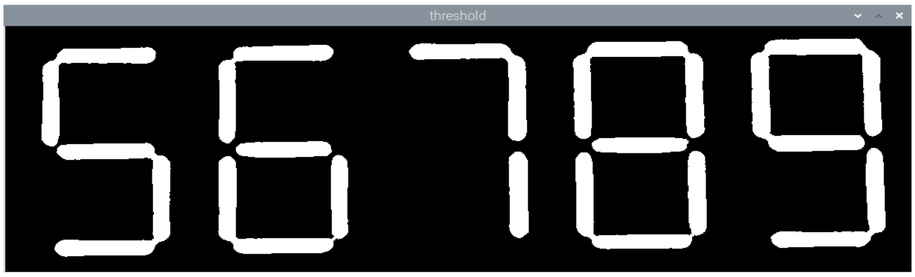
OCR can be performed now by using a threshold value of 35 and gamma 1.5. Some argument bracketing are usually needed for good OCR results. Argument -s is used to create preprocessed images, and it is very useful when searching for optimal OCR parameters. The threshold and output images are most helpful in this situation, as shown in Figure 6.

Here is the terminal output from a single image OCR operation:

```
$ numlog.py frame01.jpg -i -t 32 -gamma 1.5 -s
Seven segment LCD number logger

Current directory:
/home/pi/python/20210819-numlog

ROI file found in current directory: 0.3005,0.3963,0.5464,0.3046
Digits: [5, 6, 7, 8, 9]
Value:  56789
```



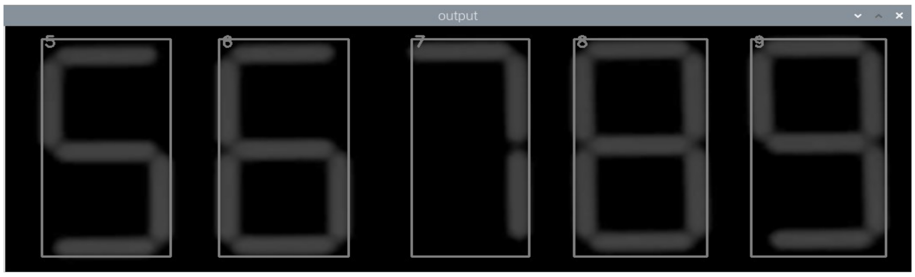


Figure 6. Threshold and output images from a single OCR job.

The numlog was then run in batch mode, and here is its terminal output:

```
$ numlog.py
Seven segment LCD number logger

Current directory:
/home/pi/python/20210819-numlog

ROI file found in current directory: 0.3005,0.3963,0.5464,0.3046

Enter unit (Default=s: <Enter>):
Default unit selected: s

Extension for analysis images included in search (Default jpg: <ENTER>)?
Default selected: jpg

Select start pass filter for analysis file names (Enter=None)? frame

Enter y axes text (Default=Value: <Enter>):
Default text selected: Value

Select interval: (1e-09...1000000000.0 s, Default=1: <Enter>): 0.1

First image start time (Default=0 s: <Enter>)?

Add filename column to data (Y/N, Default n: <Enter>):
Default selected: Filename column disabled
01/44: 56789
02/44: 56789
03/44: 56789
…
```

After a successful OCR, a log and a data csv files are written and a time graph saved to the current directory. Here is an example of a log file, and the time graph is shown in Figure 7.

```
Log name: numlog20210820_160057

Image name:

Batch OCR       : Enabled
Capture image   : -
Real-time logger: -

Unit                : s
Extension           : .jpg
Start pass filter   : frame
Interval            : 0.1 s
Start time          : 0 s
Add filename column : No

Inverse display : Yes
H/W ratio       : 2.0
Tilt angle      : 10.0
Threshold       : 32

Digits     : -
Decimals   : -

Black point: -
White point: -
Gamma      : 1.5

Show image : No
Debug      : No

Command:
numlog.py -b -i -t 32 -gamma 1.5
```

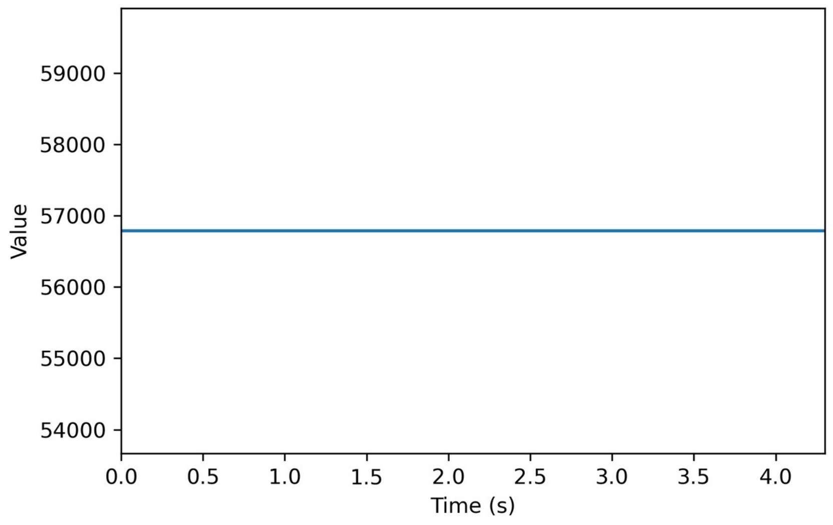


Figure 7. Time graph: OCR number as function of time. No numbers changed in this experiment, which therefore resulted in a horizontal line.