COM-389

Sultan Umarbaev

Michael Brady

Mathematical Expression Solver using camera

The current project is a modified implementation of the idea of mathematic calculator that uses camera. The idea is the calculator that will solve mathematical expressions and equations from the images. The calculator takes a picture of the expression then reads and solves those expressions The project is different from the idea in the way that it works in real-time, is capable of solving several expressions at once and puts the answer near the expression. It does not have the options to take a picture of expression and press button to solve it. The goal for the project was to create real-time mathematical expression and equation solver. At the current stage it only is capable of solving basic arithmetic tasks such as addition, subtraction, multiplication and division. The project is consist of 3 parts: text detection, text recognition, expression solving. For this project OpenCV and TinyExpr libraries and Tesseract OCR engine were used.

OpenCV is an open source machine learning and computer vision library. It is free and cross-platform for use. OpenCV is written natively in C++, however it also has Python, Java and MATLAB interfaces. The library has more than 2500 optimized algorithms for image and video processing and machine learning. These algorithms can be used for detecting and recognizing faces, identifying objects, tracking movements and actions etc. The library has many modules for using those algorithms such as *highgui, objdetect, imgproc, dnn* etc. In addition, it supports CUDA and OpenCL interfaces to work with GPU for faster and more effective performances.

TinyExpr is a very small recursive parser and evaluation engine for math expressions. It

is free and open source library. The library is simple and fast for use. TinyExpr supports standard math operators and also the standard C math functions.

Tesseract is free and open source optical character recognition engine for different platforms. It is considered one of the most accurate free and open source OCR for printed text, not for handwritten text. Tesseract supports UTF-8 and is capable of recognizing more than 100 languages. Tesseract is written in C and C++ languages. It can be used from command line interface and can be integrated to C++ and Python programming languages with the help of Tesseract API. In the latest Tesseract version 4 a Long Short Term Memory (LSTM) was added, it is a type of a Recurrent Neural Network (RNN) which is good for sequence of characters rather than a Convolutional Neural Network (CNN) which is better for a single character.

Since the launch of the program it initializes the Tesseract API and loads EAST model, which is the trained network for the text detection. The project runs in real-time by working with every frame of the camera. Text detection part is done with the help of OpenCV's deep neural network (*dnn*) module and its function *blobFromImage* for preprocessing every frame, which are passed to the function as an input image, and preparing them to pass it through the network. The function performs mean subtraction of the frame by the mean values that are set manually and passed to the function with input image. The mean subtraction is preprocessing method that subtract the mean from each input channel, the *Red, Green* and *Blue* channels, of the input image. In addition, after mean subtraction the function can apply scaling process to the input image, by dividing the result of mean subtraction by scaling factor. The scaling factor is also set manually or can be set to 1 to perform mean subtraction only. The output result of the function is 4-dimensional blob that is created from input image. After preparing the frame as input image for

the network, another function of the *dnn* module is used that is called *forward*. It passes the input image through the network. The output of the function is consist of 2 parts: one is geometry which is information about the text box and other is the confidence score of the detected box. The geometry output is used for calculating position of text box and drawing a rectangle around the text (Figure 1). The confidence score information is used to filter out the most probable text box by applying threshold score. In addition for last process, the Non-Maximum Suppression (NMS) technique is applied. The dnn module has its own function for this method. One text box can be detected many times that results in messing, therefore the NMS technique is used to eliminate this problem and detect each text box only one time by choosing the best one out of those multiple detections.

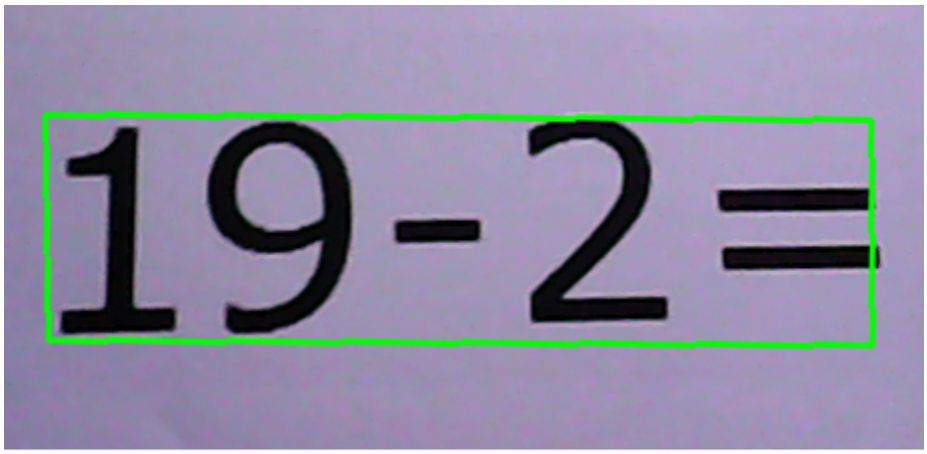

Figure 1: Text detection

The next step or part is text recognition. This step requires the use of Tesseract OCR API and *imgproc* module. Once text is detected, it is cropped from the frame. After getting the cropped image, the several image preprocessing techniques are used for image data improvement. First, image is converted to the grayscale which reduces the complexity that occur with colored images. Second, image is blurred using Gaussian blur method to smooth image and

get rid of noises and distortions. Final preprocessing operation is binarization, image is is converted to black and white one by applying thresholding. All three operations are performed by the functions *cvtColor, GaussianBlur, adaptiveThreshold* (Figure 2). The last step of this part is to pass the image to the Tesseract OCR engine. The output text can be stored as *char** variable (Figure 3).

Figure 2: Image preprocessing for recognition

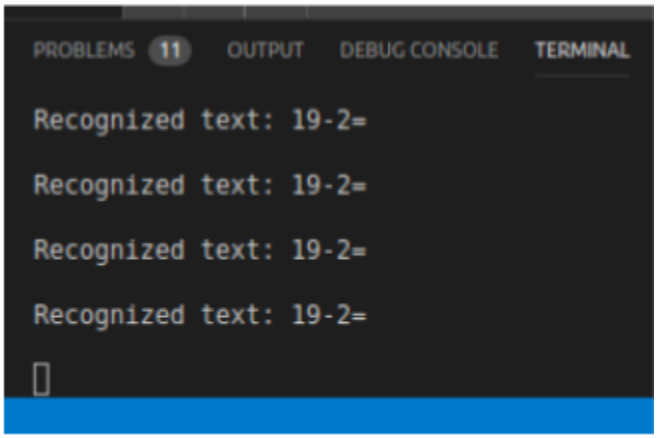

Figure 3: Recognition result

Third and final part is expression solving. For solving mathematical expressions the TinyExpr library was used. After getting the output result from the OCR engine, it is converted to the *string* data type for simplifying the parsing process. Next, expression is getting parsed and unnecessary characters are removed such as ‘ ‘ and ‘=’ that distract the work of TinyExpr library. Some characters are substituted, for example, ‘x’, ‘X’ are changed to multiplication sign, ‘:’ is changed to division sign. The expression is converted back to *char** and passed to the evaluation

engine. The output result is stored in *double* variable and put near to the expression in the frame (Figure 4).

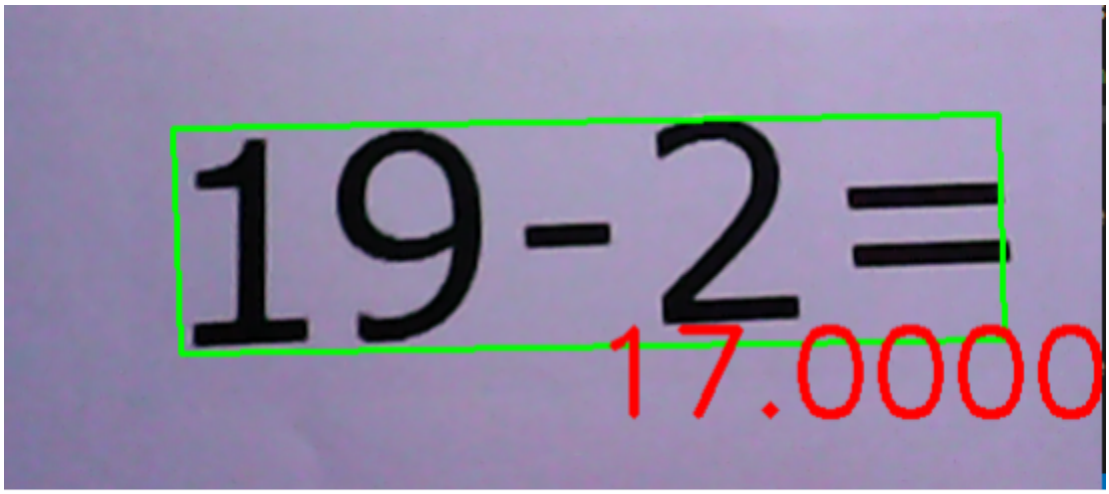

Figure 4: Final result of the solution put near the expression.

In addition, the option of cropping detected text box from rotated frame was implemented. In order to crop text box from rotated image, the image needs to be straightened according to the text box (Figure 5). The geometric transformation can be applied using *warpAffine* function. The function is used to rotate the image around a defined center point which in this case would be used to straighten the rotated image. The defined center point is the center of text box. The function requires the use of transformation matrix which can be computed using *getRotationMatrix2D* function.

The program is only capable of solving basic arithmetic tasks. Future developing includes the option of solving simple and quadratic equations and the ability to detect and recognize handwritten expression and equations.

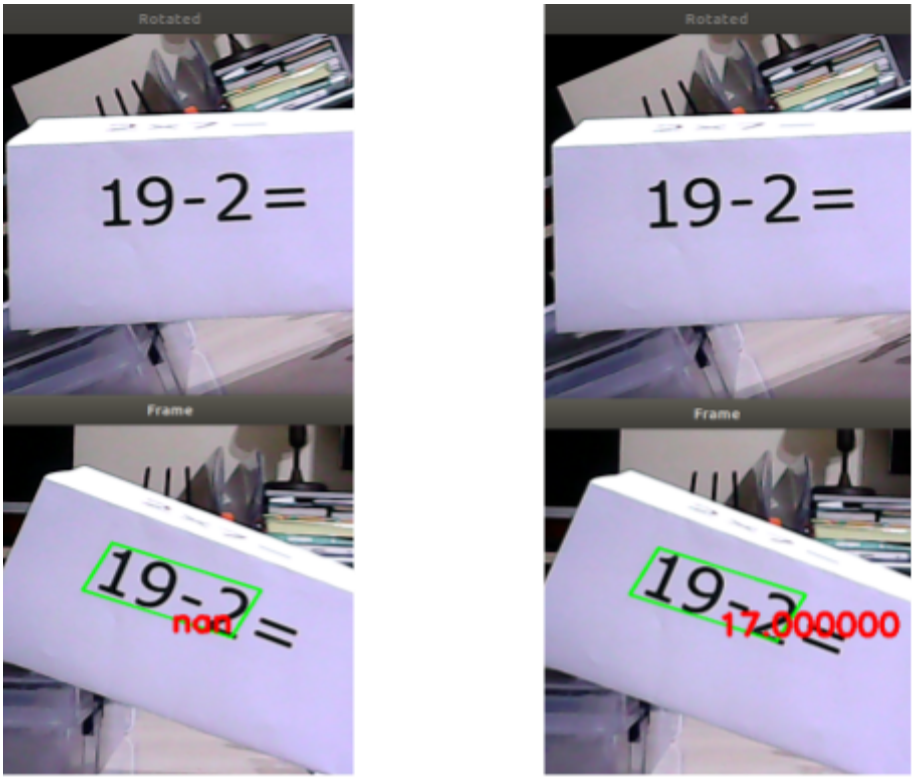

Figure 5: Cropping from rotated image

References

1. https://docs.opencv.org/
2. https://github.com/tesseract-ocr/
3. https://github.com/codeplea/tinyexpr
4. https://www.learnopencv.com/deep-learning-based-text-recognition-ocr-using-tesseract-and-opencv/
5. https://www.learnopencv.com/deep-learning-based-text-detection-using-opencv-c-python/
6. http://felix.abecassis.me/2011/10/opencv-rotation-deskewing/
7. http://par.cse.nsysu.edu.tw/~algo/paper/paper16/B2_1.pdf
8. https://stacks.stanford.edu/file/druid:yj296hj2790/Sikka_Wu_Automatic_Equation_Solver_and_Plotter.pdf
9. https://blog.ayoungprogrammer.com/2013/01/equation-ocr-part-1-using-contours-to.html/
10. http://www.willforfang.com/computer-vision/2016/4/9/artificial-intelligence-for-handwritten-mathematical-expression-evaluation