IEEE Second International Conference on Data Stream Mining & Processing
August 21-25, 2018, Lviv, Ukraine

Embedded Vision Modules for Text Recognition and Fiducial Markers Tracking

Ievgen Gorovyi
It-Jim
Kharkiv, Ukraine
ceo@it-jim.com

Vitalii Vovk
It-Jim
Kharkiv, Ukraine
ceo@it-jim.com

Maksim Shevchenko
It-Jim
Kharkiv, Ukraine
ceo@it-jim.com

Valerii Zozulia
It-Jim
Kharkiv, Ukraine
ceo@it-jim.com

Dmytro Sharapov
It-Jim
Kharkiv, Ukraine
ceo@it-jim.com

***Abstract*—In the paper, two examples of embedded vision modules are described. Firstly, it is demonstrated how fiducial marker tracking algorithm can be adopted for operation on Raspberry Pi. Usage of proposed ideas allows to achieve around 60fps speed of binary marker tracking. Secondly, we describe the problem of text detection and recognition in outdoor environment. Experimental results indicate on acceptable results and good potential to provide low-cost and efficient embedded vision system for this purpose. Technical details of both embedded vision modules are comprehensively discussed.**

Keywords—computer vision, Raspberry Pi, fiducial markers, tracking, text recognition.

I. INTRODUCTION

Computer vision (CV) is a rapidly growing discipline making machines to percept and understand their surroundings as humans do [1]. There are a lot of practical applications of CV in medicine, industry, entertainment and many more [2]. CV algorithms can be run on different hardware: desktops, mobile phones, various digital signal processing (DSP) units. A particular interest is related with usage of low-power hardware such as Raspberry Pi [3]. Indeed, Raspberry is light weight, cheap and widely available in the market. Embedding of CV solutions transforms it into mobile autonomous intellectual system. Fig. 1 contains an example of Raspberry (Fig. 1a) and its setup with camera (Fig. 1b).

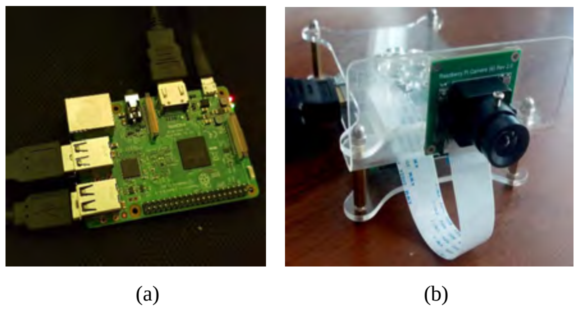

Fig. 1. Raspberry PI 3 and camera. (a) Raspberry PI Model 3B, (b) Raspberry with camera

Research community made a lot of CV experiments with Raspberry Pi. Example of multiple objects tacking can be found in [4]. In [5] a compact stereo-vision system is described. A full stereo matching pipeline is constructed allowing to use the system as a depth-meter for outdoor scenarios. A specific use case in given in [6], when bees behavior is analyzed using embedded vision. Other examples include face recognition [7], license plates detection and recognition [8], autonomous cars applications [9], robotic assistants [10] and many more.

In the paper, we study two important problems. Firstly, it is demonstrated how to integrate and optimize fiducial marker recognition algorithm for Raspberry. Secondly, we describe the initial experimental results on number plate recognition as a separate embedded vision module.

Section II contains description of algorithm for fiducial marker recognition. Developed optimization steps are described as well. Section III contains information about text detection and recognition on Raspberry. Experimental examples are included in each section.

II. FIDUCIAL MARKERS RECOGNITION

In this section, we describe a pipeline for recognition of fiducial markers. Firstly, common steps for such process are explained. After that, some specifically developed improvements are discussed.

A. Main Steps of Marker Recognition Algorithm

Binary markers are used in many applications including robotic navigation [11]-[13], augmented reality [14] and logistics [15]-[17]. Examples of typically used binary markers are shown in Fig. 2.

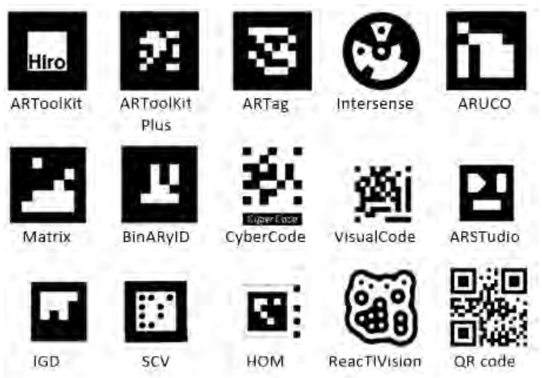

Fig. 2. Example of binary markers

The key advantage of fiducial markers usage is that we can properly build the marker structure in order to achieve high recognition rate. Also, in contrast to image markers [14], binary markers can be detected in the case more challenging geometry conditions. In our experiments, we use ArUco markers [18]. A typical marker from this collection is 7×7 blocks size containing 5×5 blocks inside (Fig. 2). A great advantage of such markers is possibility to generate a vocabulary of specific size.

A key idea of the marker recognition algorithm is to detect marker on input video frame and recognize it by comparison with previously formed database. Typically, this involves several basic steps, as shown in Fig. 3.

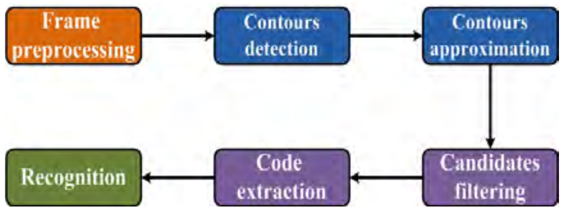

Fig. 3. Binary marker detection pipeline

Firstly, image thresholding [1] is applied to input frame. After that, contours are extracted from obtained binary image [1]-[2]. At the next step, all located contours are approximated by quadrangles. Finally, the binary codes are extracted from filtered marker candidates and then matched with existing markers database to find a proper coincidence [2].

The challenge is that the default solution is quite slow. Initial tests of performance on Raspberry give around 10 FPS. Fortunately, we have found the way to significantly boost the performance. Key proposed ideas are described in the next subsection.

B. *Integration to Raspberry*

It is common way to perform image thresholding before contour finding by usage of adaptive threshold (because of its robustness to image brightness changes). Since fiducial markers are binary images, such operation works as edge detector. We have found that usage of Canny edge detector [1] instead of adaptive threshold (see Fig. 4) provides better detection and code extraction quality, as well as has less computational complexity. Additionally, median blurring is applied to input image to suppress the noise.

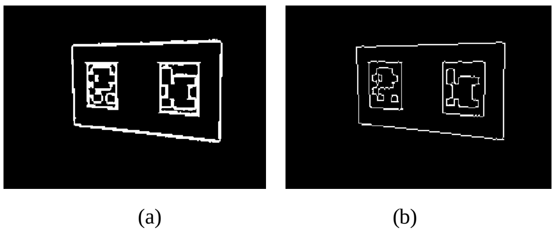

Fig. 4. Use of adaptive threshold (a) and canny edges detector (b) to prepare frame for finding contours

One of the challenges for marker recognition is existence of perspective distortions due to varying geometric conditions. In order to extract marker code, inverse perspective transformations is often applied to marker candidates. This involves calculation of corresponding transformation matrix, which is computational consuming.

In our version of the algorithm, we use improved method for code extraction that avoids the perspective transformation step. It only requires building of a grid equal to marker size directly from found contours of marker candidates, and testing pixels values on such grid (see Fig. 5).

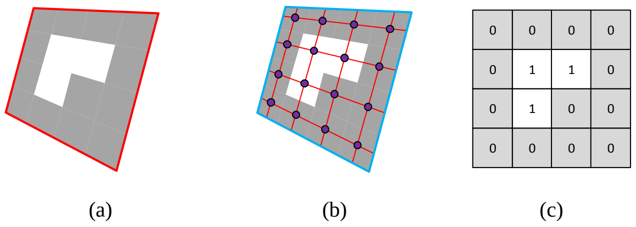

Fig. 5. Code extraction example.
Found contour (a), grid building (b) and extracted code (c).

Another challenge in code extraction is presence of glare (Fig. 6a) or shadows on frame. In such cases wrong marker code extraction may appear due to incorrect thresholding parameter. For example, in Fig. 6 even dark regions of markers have a very high pixel intensity, and constant threshold leads to incorrect code extraction.

To increase the marker recognition robustness, we propose to estimate the minimum and maximum value in extracted marker image. Then threshold value is estimated as:

$$t = c \cdot (mx - mn) + mn, \qquad (1)$$

where t is threshold value, mn and mx are minimum and maximum values within found contour, c = 0.8 is am empiric threshold coefficient.

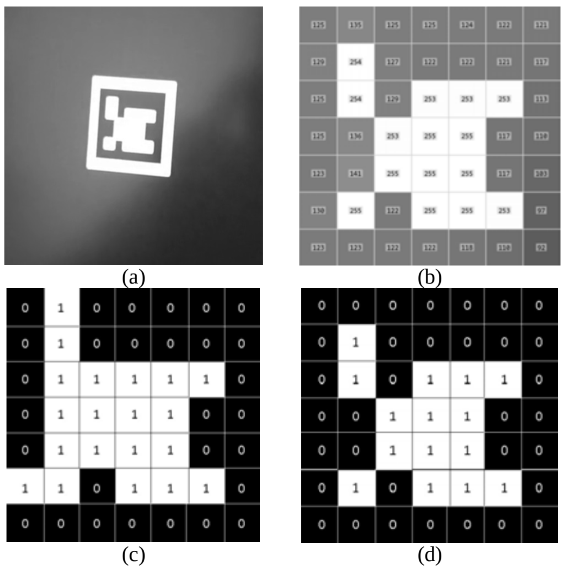

Fig. 6. Frame with marker in glare (a), extracted pixels values (b), wrong code extraction with fixed threashold (c) (value = 128), and correct code extration with dynamically computed threshold (d) (value = 222,4)

The only requirement is that all true “white” pixels within marker region have higher intensity than any “black” one.

C. *Dynamic scaling and tracking*

Image pre-scaling has significant effect on frame processing time. For example, as it is shown in Fig. 7, pre-scaling input VGA frame with scale factor 0.3 reduces total frame processing time more than 2 times with no loss in processing quality. It was found that it is possible to recognize ArUco marker even of only 30×30 pixels size for 7×7 bins. In this case, each marker bin will contain 4×4 pixels. Thus, for a particular marker we can find the scale factor make it of size 30×30 pixels and still successfully locate and recognize it:

$$scale = \frac{m}{s} \tag{2}$$

where $m = 30$ is a marker required size; s is marker current size.

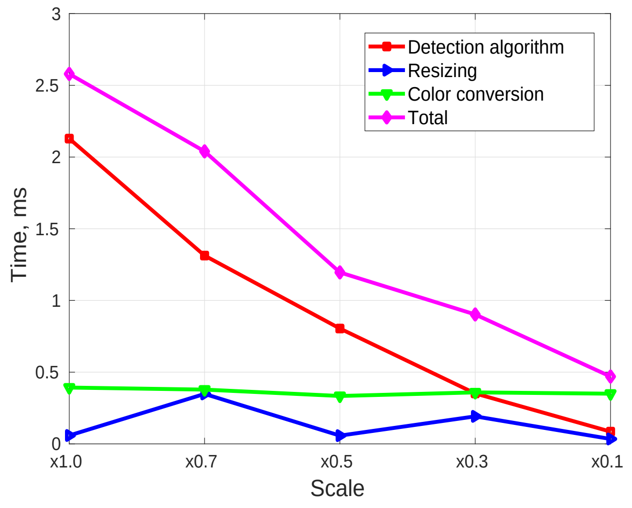

Fig. 7. Effect of pre-scaling on frame processing time.

Individual markers rescaling requires tracking of found markers and saving their last found size and position to process only small area of image or region of interest (ROI) with corresponding scale factor. Having predicted marker position and size we can build ROI around it. Because of marker position estimation error (both because of model discrepancy and marker relative velocity estimation error), found region can contain only part of the marker, or even totally miss it. In order to provide stable detection of marker, found region is extended by a precomputed value. We have found a specific empirical parameter allowing to find the required extension region

$$e = c \cdot \max(w, h), \tag{3}$$

where w and h are the current frame width and height, and extension constant $c = 0.075$. Then, ROI for analysis is equal to predicted marker's bounding box extended at each edge by value e:

$$\begin{gathered} x^{*} = x - e,\ \ y^{*} = y - e, \\ width^{*} = width + 2 \cdot e,\ height^{*} = height + 2 \cdot e. \end{gathered} \tag{4}$$

One should notice that additional analysis of ROI position and size should be performed to avoid violation of frame borders. Described solution of marker ROI scaling gives reasonable increase in frame processing performance (Fig. 8). One can see that tracking gives boost in performance in comparison with direct full frame analysis approach in the most cases. Only in rare cases tracking takes more time: when all markers are lost (because it requires full frame processing, but also additional resources for tracks re-initialization), and if integral ROIs' area exceeds source frame area. All of the improvements above provide increase of FPS from 10 in initial setup to near 60 in final algorithm version (see Tab.1) with high recognition accuracy.

TABLE I. ACHIEVED FRAME RATE ON DIFFERENT STAGES OF ALGORITHM IMPROVEMENT

	Initial setup	Improved detection and code extraction	Pre-scaling	Tracking
FPS	10	15-20	35-40	~60

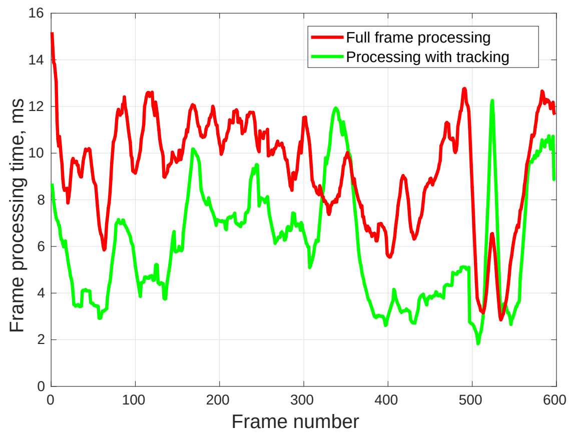

Fig. 8. Comparison of frame processing with (green) and without (red) marker tracking.

Thus, we have developed and integrated fiducial marker recognition solution on Raspberry.

III. AUTOMATIC TEXT RECOGNITION ON RASPBERRY

Another analyzed problem is related with text detection and recognition on Raspberry. For this purpose, we have made several outdoor videos with cars and tried to localize and recognize number plates. In order to perform scene text recognition, firstly we need to localize the number plate. To solve this problem, we have utilized the extremal regions (ER) detector [19]-[20]. The developed text detection pipeline is the following (Fig. 9).

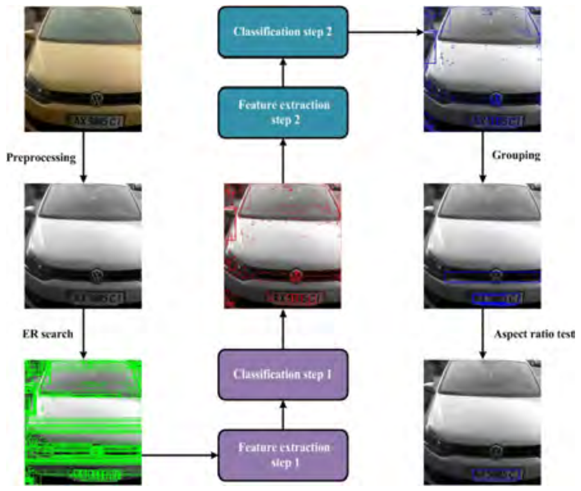

Fig. 9. Main steps of number plate detectoin algorithm

Firstly, contrast enhancement and adaptive thresholding are applied. After that a set of morphological operations is applied to filter out the noise (Fig. 10). After that letters segmentation is used. Finally, Tesseract engine is used for text recognition.

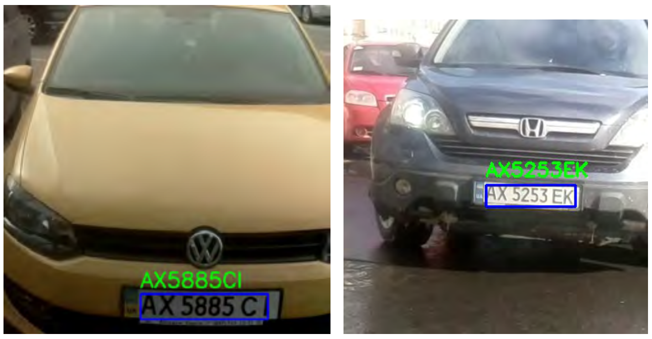

Fig. 11. Example of number plate recognition

Fig. 11 contains an example of number plate detection and recognition in Raspberry camera frames. In the near future we are planning to optimize the developed algorithm.

IV. Conclusions and Future Work

In the paper, we considered two examples of embedded vision systems. Firstly, we comprehensively analyzed the fiducial marker recognition framework and developed several optimization steps. Experimental results indicate a significant improvement from both accuracy and speed points of view. In addition, we have shown an example of a prototype of number plate recognition system. Initial results show a good potential of production of low-cost and efficient module for intelligent transportation applications.

References

[1] R. Szeliski, Computer vision: Algorithms and Applications. London etc.: Springer, Sept, 2010.

[2] D. Baggio, S. Emami, D. Escriva, K. Ievgen, J. Saragih and R. Shikrot, Mastering OpenCV 3 - Second Edition. Birmingham: Packt Publishing Ltd, Apr, 2017.

[3] https://www.raspberrypi.org

[4] A. Dziri, M. Duranton, and R. Chapuis, “Real-time multiple objects tracking on Raspberry-Pi-based smart embedded camera,” Journal of Electronic Imaging, vol. 25(4), 2016

[5] James Cooper et. al., “A Raspberry Pi 2-based Stereo Camera Depth Meter,” International Conference on Machine Vision Applications, Nagoya, Japan, pp. 274-277, May 8-12, 2017.

[6] Gang Jun Tu, Mikkel Kragh Hansen, Per Kryger, and Peter Ahrendt, “Automatic behaviour analysis system for honeybees using computer vision,” Computers and Electronics in Agriculture, vol. 122, pp. 10–18, 2016.

[7] R. Mo, and A. Shaout, “Portable Facial Recognition Jukebox Using Fisherfaces (Frj),” International Journal of Advanced Computer Science and Applications, vol. 7, no. 3, pp. 9-14, 2016.

[8] K. Sri Sasikala, and Shakeel Ahmed, “Implementation of Number Plate Extraction for Security System using Raspberry Pi Processor,” International Journal of Engineering Research & Technology (IJERT), vol. 5, iss. 03, pp. 317-321, March-2016.

[9] Gurjashan Singh Pannu, Mohammad Dawud Ansari, and Pritha Gupta, “Design and Implementation of Autonomous Car using Raspberry Pi,” International Journal of Computer Applications, vol. 113, no. 9, pp. 22-29, March 2015.

[10] Rizqi Andry Ardiansyah, “Design of An Electronic Narrator on Assistant Robot for Blind People,” MATEC Web of Conferences, 42: 03013, 2016.

[11] Rafael Munoz-Salinas, Manuel J. Marin-Jimenez, Enrique Yeguas-Bolivar, and R. Medina-Carnicer, “Mapping and localization from planar markers”, Pattern Recognition, vol. 73, pp. 158-171, 2018.

[12] K. Horak, and L. Zalud, “Image Processing on Raspberry Pi in Matlab,” Advances in intelligent systems and computing, p. 25, 4 November 2015.

[13] A. Babinec, L. Jurisica, P. Hubinsky, and F. Duchon, “Visual Localization of Mobile Robot Using Artificial Markers,” Procedia Engineering, vol. 96, pp. 1-9, 2014.

[14] Ievgen M. Gorovyi, and Dmytro S. Sharapov, “Advanced Image Tracking Approach for Augmented Reality Applications,” Signal Processing Symposium (SPSympo-2017), 12-14 September, Jachranka, Poland, pp.266-270, 2017.

[15] Sherin M. Youssef, and Rana M. Salem, “Automated barcode recognition for smart identification and inspection automation,” Expert Syst. Appl., vol. 33, pp. 968-977, 2007.

[16] C. Ozgur, C. Alias, and B. Noche, "Comparing sensor-based and camera-based approaches to recognizing the occupancy status of the load handling device of forklift trucks,” Logist. J. Proc., pp. 1-9, 2016.

[17] C. Alias, C. Ozgur and B. Noche, “Monitoring production and logistics processes with the help of industrial image processing,” 27th Annual POMS Conference 2016, Orlando (FL), USA, 2016.

S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, “Automatic generation and detection of highly reliable fiducial markers under occlusion,” Pattern Recognition, vol. 47, iss. 6, pp. 2280–2292, June 2014.

[19] L. Neumann and J. Matas, “Real-Time Lexicon-Free Scene Text Localization and Recognition,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 38, no. 9, pp. 1872-1885, 2016.

[20] L. Neumann and J. Matas, “Text Localization in Real-World Images Using Efficiently Pruned Exhaustive Search,” ICDAR Proc. International Conference on Document Analysis and Recognition, pp. 687-691, Sept, 2011.

[21] R. Smith, “An Overview of the Tesseract OCR Engine,” ICDAR Proc. Ninth Int. Conference on Document Analysis and Recognition , pp. 629-633, 2007.