Baltic J. Modern Computing, Vol. 9 (2021), No. 1, 115-125
https://doi.org/10.22364/bjmc.2021.9.1.06

Automatic Recognition of ArUco Codes in Land Surveying Tasks

Zoltán SIKI, Bence TAKÁCS

Budapest University of Technology and Economics, Faculty of Civil Engineering, Department of Geodesy and Surveying, Műegyetem rakpart 3., Budapest, Hungary

siki.zoltan@epito.bme.hu, takacs.bence@epito.bme.hu

Abstract. Photogrammetry is experiencing its renaissance in the 21st century. The main driving forces are research in computer vision as well as drone (UAV/UAS/RPAS) technology. There are popular open-source libraries and applications available in these fields (e.g. OpenCV and ODM). In this paper, we report our experiments using OpenCV in specific land surveying projects. Our efforts concentrate on two areas. The first one is the application of close-range photogrammetry for deformation observations. Fitting a simple camera to the eyepiece of a total station we reached sub millimetre movement detection from few meters. This method was used for dynamic test load of a bridge and a floating platform. The second area is automatic ground control point (GCP) detection in images taken from drones. Optimizing the size and colours of the used ArUco markers we achieved nearly 100 percent results in different light conditions.

Keywords. Deformation analysis, open-source, OpenCV, ArUco codes, Python, template matching

1. Introduction

Beyond teaching, translating and developing FOSS4G (Free and Open Source Software for Geospatial) software, in our Geo4All Laboratory at the Department of Geodesy and Surveying of the Budapest University of Technology and Economics, scientific research is also a main activity. In 2008 we started an open source project, named Ulyxes (ULtimate YX Estimation System) (Ulyxes, 2020). Its main aim is to drive a wide range of sensors used in land surveying and to collect sensor data. Deformation analysis and monitoring were in focus from the very beginning. From the relatively slow total station sensors, our attention turned to cameras in recent years to be able to detect faster movements and deformations. We prefer open hardware elements, like Raspberry Pis and Pi Cameras. To increase the resolution of the Pi Camera, we often fix it to the telescope of a total station. This way even sub-millimetre movements can be detected from a range up to a few tens of meters. We managed to apply the developed technology to measure the deformation of engineering structures during dynamic test loads.

New Python modules were added to the Ulyxes code base to handle images and video records (Siki et al., 2018). We applied two algorithms from the OpenCV library, the template matching and the ArUco code detection (OpenCV contributors, 2020). With template matching, we can detect the shift of the target in a perpendicular plane to the

axis of the camera. Using ArUco module of OpenCV rotated and scaled markers can also be detected, and even pose estimation is possible, since the normal of the marker can be estimated from the image. The use of coded targets (unique black and white patterns) is quite common in close-range photogrammetry.

Another application of our research is the GCP detection in images in aerial photogrammetry. Using drones, the recommended overlapping rate among lags and images is typically very high (70-80%); therefore, you can get hundreds or even thousands of images for an area of a few hectares. The same GCP is visible in several (usually more than ten or even twenty) images. During the processing of the images all GCPs have to be marked in each image which is not only a manual but quite a time-consuming job. Open-source software typically does not provide an automatic solution to find these markers in the images. Based on ArUco markers and OpenCV ArUco library, we implemented a Python solution for automatic detection of ArUco markers, which can also create the necessary GCP file for well-known open-source photogrammetric software, like Open Drone Map (ODM) and VisualSfM. The Python code of our small module is also open-source.

Various tests were carried out on reflectivity, size, colours and resolution of the markers. The first practical experiences are reported in this paper, and some recommendations are also given.

2. ArUco markers and calibration

ArUco markers (Garrido-Jurado, 2014) were developed primarily for augmented reality applications, but are also used for indoor robot navigation, in simultaneous localization and mapping (SLAM) for instance. The OpenCV computer vision open-source library (Shilkrot and Escrivá, 2018) contains a contrib module to generate and detect ArUco markers (Fig. 1.). These square markers are composed of a wide black border and an internal matrix of black and white smaller squares. Different dictionaries are available with 4x4, 5x5, 6x6, 7x7 matrices. The larger the matrix is the more different unique patterns are available in the dictionary. Depending on the target distance different marker sizes and matrix resolutions are optimal.

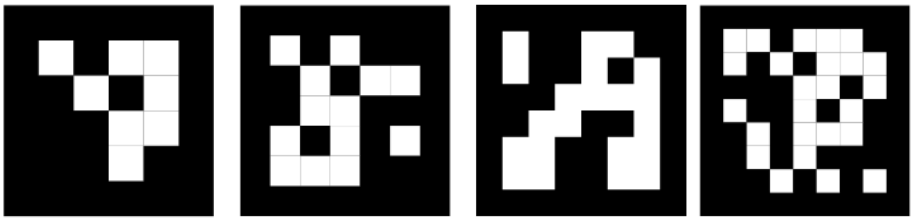

Fig. 1. ArUco markers with different matrix sizes (4x4, 5x5, 6x6, 7x7 matrices)

The OpenCV ArUco module is capable among others to detect markers in digital images returning its corner position, marker identification and pose. When measurements are made on images, eliminating lense distortions is a key point. There are several functions available to calibrate cameras to determine camera intrinsic parameters as well as distortion coefficients. OpenCV provides chessboard and circle board

calibration, while the ArUco module extends these possibilities with ArUco board and ChArUco board calibration.

Calibration is more important in case of low cost cameras, e.g. Raspberry Pi Cameras. When the target distance is too large or the resolution of the image should be increased, the camera can be fixed on the telescope of a surveyor instrument. This case the whole system with all the lenses has to be calibrated together (Völgyesi and Tóth, 2020).

We chose the ChArUco board calibration, since it has some advantages compared to OpenCV provided methods. Partial view of the board can be used and occlusions can occur. The result of the calibration is the camera intrinsic matrix (3x3 matrix with focal length and principal point offset) and the distortion parameters (6 radial and 2 tangential distortion coefficients). We developed a user-friendly short Python program for the ChArUco camera calibration and for undistorting the images (Fig. 2.). We made the program available in our Find-GCP project on GitHub (Find-GCP, 2020).

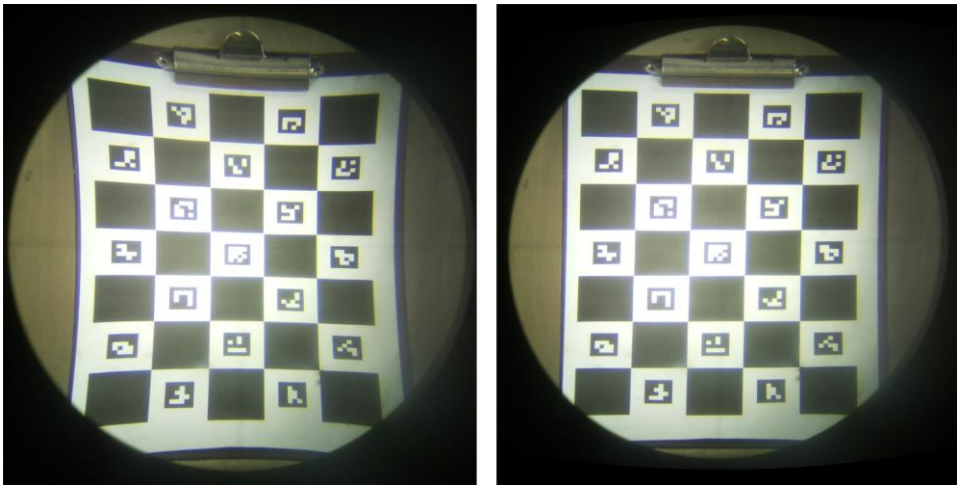

Fig. 2. Original Pi camera image through the telescope with negative radial distortion (left) and the undistorted one (right)

3. Deformation detection

There are several automatic systems to detect movements and deformations using robotic total stations, some of them are open-source (Engel and Schweimler, 2016; Siki et al., 2018). These systems are designed mostly for slow motion objects, buildings for instance, where the frequency is quite low, observations are repeated typically on a daily or hourly basis. Other times land surveyors have to detect faster movements, for instance in dynamic test loads of engineering structures. Not only does the more frequent sampling mean a challenge but also the detection of extremely small movements. To meet these challenges, digital cameras can be used taking images up to 50 Hz. To increase the resolution of the image, the camera can be mounted on the eyepiece of a theodolite or a total station. This way even 0.1 – 0.01 mm movements can be detected depending on the target distance and camera resolution.

The camera fixed on a theodolite was used in the dynamic test load of the Rákóczi Danube bridge in Budapest in 2015 (Kovács et al., 2016). That time a digital microscope camera was fitted to the eyepiece of a theodolite and a laptop computer recorded the video stream of the camera through its USB port. A laser interferometer was also applied to check the camera measurements. The diagrams of the two methods (camera and interferometer) showed a strong correlation (Fig. 3.). OpenCV template matching algorithm was used to get the movements from the video. Its post processing frame by frame yielded a satisfactory solution since neither any rotation nor a scale factor change occurred. At first, displacement of the marker position was detected in pixels, then it was converted into a metric value using the image scale. The camera and the theodolite were few metres away from the target and the differences to the observations of the interferometer were below 0.1 mm.

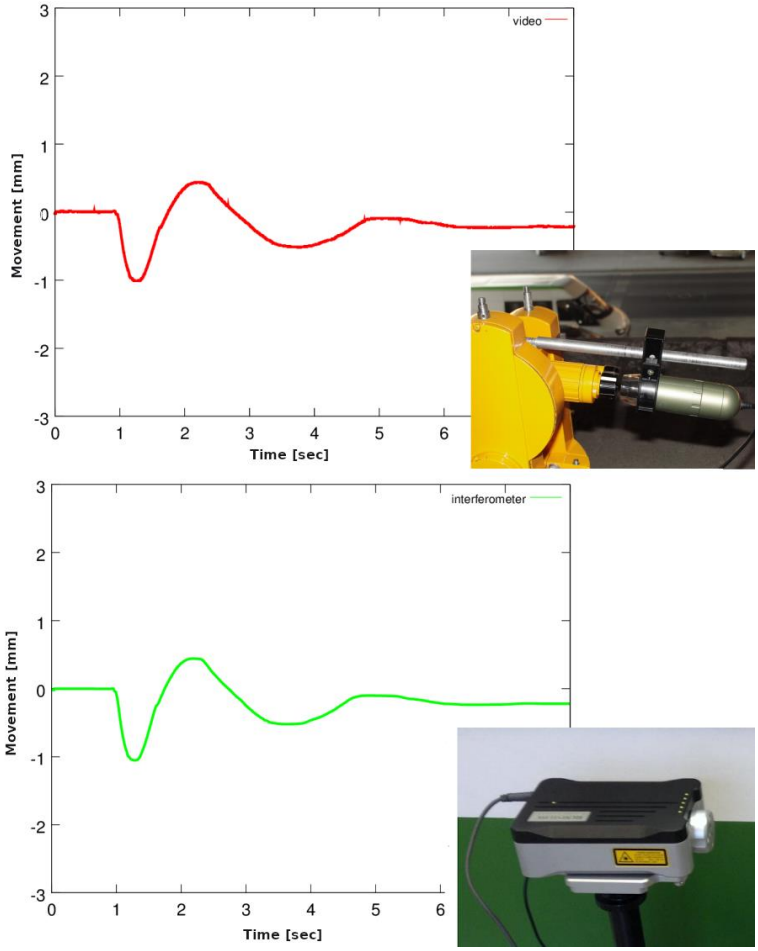

Fig. 3. Time/movement diagrams from image processing (up) and interferometer (down)

After the first attempts using cameras in movement detection, further efforts were made to prove the usefulness of this technology. Our Geo4All Laboratory launched a

project to clean up the Python code for template matching (Hashemi et al., 2016) and make it public on GitHub as an incorporated part of our Ulyxes project. OpenCV provides six different methods (objective function) to calculate correlation between the template and the image. Some cases template matching gives false results (Brunelli 2009). Unfortunately, we had to face this issue when brightness or other parameters of the image were changed significantly. According to our experience, the best results can be achieved using the normalized cross-correlation method.

Fig. 4. Raspberry Pi model 4, Pi Camera V2 fitted on the telescope of a total station

Template matching gives satisfactory results when the movement of the marker is parallel to the image plane. This restriction cannot be completed in all cases, hence feature detection algorithms, based on the aforementioned ArUco markers as well as OpenCV ArUco module, proved to be an alternative solution. If ArUco markers are fixed to the moving structure, both methods can be used. In addition to the improvements of the algorithms, some hardware elements were also updated. We switched to Raspberry Pi model 3 and model 4 as well as to Pi Camera V2. A special adapter was developed to mount the camera on the eyepiece of an old basic Leica total station (Fig. 4.). Since only the telescope of the total station is used, there are no special requirements of the instrument. Beyond the small size and low energy consumption, its price made this instrument a preferable choice.

This new configuration was used during the test load of a floating port platform where 2 Hz data sampling was required due to the waving of the river. In order to detect movements of the structure, three total stations with Raspberry Pi cameras were used at the same time (Fig. 5.) Although the OpenCV library and our Python programs can be installed on Raspberry microcomputers, the hardware and the storage unit (SD card) are not fast enough to process frames on-the-fly. Therefore, videos were recorded with 2 fps and the data were processed afterwards on personal computers. During the load test various geodetic and other instruments were also applied (high precision Real-time

Kinematic GPS receivers, robotic total stations, surveyors' levels, inclinometers and draft sensors).

Fig. 5. Floating port platform and image through the telescope

At the present investigation we focus on the vertical displacement of the platform and the effect of waving which were detected from digital images. The results of the processing of the video with pattern matching can be seen on Fig. 6. The upper diagram on Fig. 6 shows the vertical movement as the load is incrementally removed. The bottom one shows the quality of the pattern matching, from the six provided methods of OpenCV we got the best results using the normalized cross-correlation matching method (1). The match quality is measured by the value of normalized cross-correlation matching between the template and the part of the image, where zero means the total mismatch and one is the perfect match.

$$R(x,y) = \frac{\Sigma_{x',y'}\left(T(x',y')\cdot I(x+x',y+y')\right)}{\sqrt{\Sigma_{x',y'} T(x',y')^2 \cdot \Sigma_{x',y'} I(x+x',y+y')^2}} \qquad (1)$$

where:

x, y – column and row position in the image

x', y' – column and row position in the template

I – image greyscale value at position
T – template greyscale value at position
R(x, y) – normalized cross correlation at position x,y

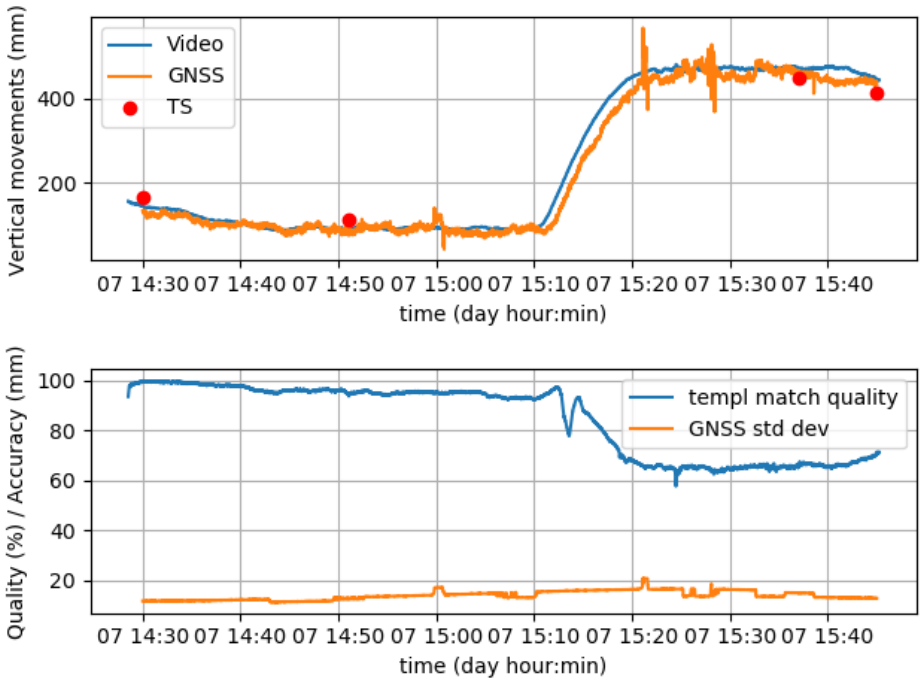

Fig. 6. Test load of a floating port platform (~0.35 m vertical movement)

To be able to validate the proposed approach based upon digital image processing, its results were compared to the vertical displacements measured by RTK GNSS and total station (TS) as reference. Seven load cases were chosen when significant vertical displacements (typically 20 to 30 centimetre) over short period of time (typically 10 to 20 minutes) were measured (Fig. 6). Results agree within 12 percent of the displacement which seems to be well promising. On the other hand, further investigations should be carried out to improve the initial results for more distance targets (30-50 m), especially in the field of camera calibration, marker establishment and outlier detection during image processing.

4. GCP detection

Automatic GCP detection on drone images can be done only with feature detection since the direction and the size of the markers can vary from image to image. Another important difference is that the size of the marker is significantly smaller compared to the augmented reality or the formerly discussed deformation detection applications. Fortunately, the developers of the ArUco module of OpenCV provide several customizable parameters to the users. The *minMarkerPerimeterRate* parameter defines the ratio of the marker perimeter and the longer image size for instance. The default

value of this parameter is set to 0.03. Our DJI drone makes 5472x3648 pixels images, so the minimal marker perimeter should be 164 pixels and the marker size 40 pixels.

Granted that the usual flight altitude is 20-100 metres, the pixel size in real world measurements is ~1-3 cm in general. Following the previous calculation the marker size should be ~40-120 cm which makes it difficult to use it in the field. Consequently the default value of the *minMarkerPerimeterRate* parameter should be decreased. A 0.01 value yields to set the marker size to about 15-30 cm. To demonstrate its effect, three tests were carried out with different markers and flight altitude using a DJI 4 Pro drone. Approximately 80% overlap was among images, so the same marker is visible in several images.

During the first test black and white 4x4 markers were used in cloudy weather. The results were not really well-promising since only less than half of the markers were detected (Table 1.).

Table 1. Summary of first attempt of ArUco marker automatic detection

No. of images	Marker size [cm]	Flight altitude [m]	Marker size [pixel]	No. of markers	No. of detected markers
10	20x20	20	33x33	23	13
5	20x20	35	27x27	15	5
4	20x20	50	20x20	15	8
3	20x20	70	12x12	13	2

In the majority of the cases the white area of the markers were burnt on the photos, that is to say, the white parts became larger and the black ones smaller (Fig. 7.). Changing the parameter named *perspectiveRemoveIgnoredMarginPerCell* the id estimation becomes more robust, the effect of burnt could be significantly reduced. It is a relative value and defines the percent of pixels to ignore in the margins of cells (OpenCV contributors, 2020). In case of high burnt effect, we used 0.33 (e.g. 33% reducement of matrix elements), the considered areas are green boxes on Fig. 7. right. Another experience was about the minimal marker size in pixels, it should be at least 20 pixels wide/high.

During our first attempt the markers were printed on normal paper. A new idea was that if the marker quality (contrast) were better, less burnt effect would occur. Consequently, new markers were created using self-adhesive non-reflective black foil on white plastic board. In addition, the size of the boards was increased to 28 cm. We had the second test on a sunny day with the new markers. The results were not so promising as in the first test, since the burn-in-effect of the white parts was even higher because of the strong sunshine (Table 2.). Only the larger markers (over 40 pixels) were detected in the images.

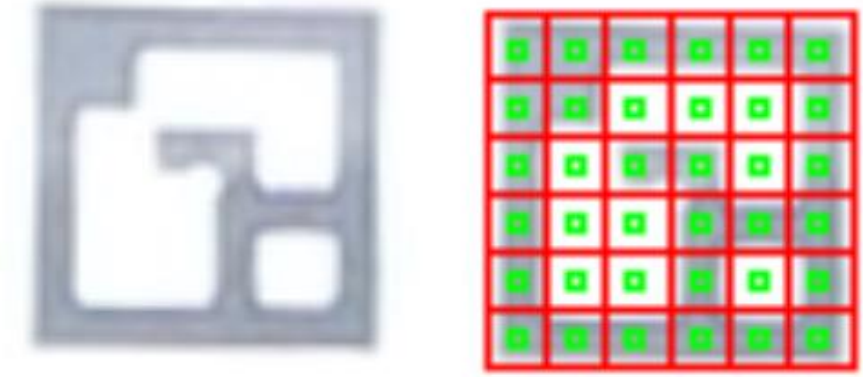

Fig. 7. Burnt marker from a drone image (left) and the effect of perspectiveRemoveIgnoredMarginPerCell parameter (right)

Table 2. Summary of second attempt of ArUco marker automatic detection

No. of images	Marker size [cm]	Flight altitude [m]	Marker size [pixel]	No. of markers	No. of detected markers
4	28x28	20	45x45	9	6
5	28x28	30	30x30	9	0
4	28x28	50	24x24	16	0
2	28x28	70	15x15	12	0

After the second test we realized that increasing the contrast of the markers did not help. Examining the code and documentation of the ArUco module of OpenCV we learned that the module uses adaptive thresholding (Roy et al., 2014) to distinguish white and black areas of the markers. With this in mind, we changed the white blocks to grey, keeping the board size at 28x28 cm. Another improvement was to use 3x3 matrix, hoping the larger matrix elements could be detected from higher elevations without increasing the total marker size (Fig. 8.). These two improvements contributed to finding the ArUco markers in images (Table 3.). Markers were placed partly in strong sunlight and partly in shadow to test both situations.

Our GCP marker detection program can create ODM and VisualSfM compatible GCP files, which can be used to georeference point clouds in the aforementioned programs.

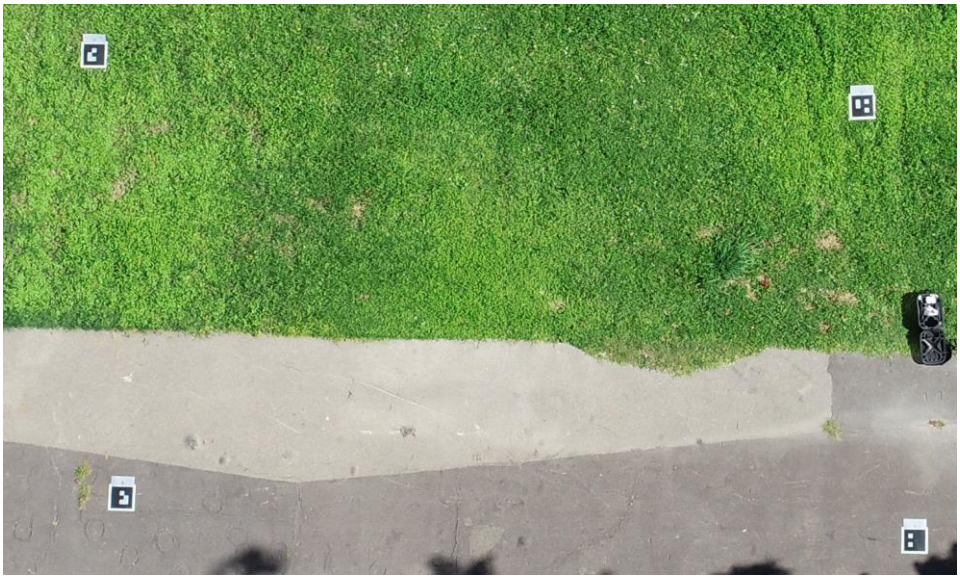

Fig. 8. Grey/black 3x3 markers on a sunny site

Table 3. Summary of third attempt of ArUco marker automatic detection

No. of images	Marker size [cm]	Flight altitude [m]	Marker size [pixel]	No. of markers	No. of detected markers
6	19x19	20	33x33	20	15
2	19x19	30	23x23	9	7
7	28x28	30	34x34	36	36
4	28x28	50	20x20	31	31
7	28x28	65	15x15	59	52

5. Conclusion

We have reached a point in both presented projects where results and the open-source code can be useful for other researchers/users. The deformation detection solution based on camera images and ArUco markers has already been successfully applied under real conditions in a real test loading task.

In case of the automatic GCP detection we have only test results using a few dozens of images. The latest results are promising, more extensive tests are required in larger, real life projects.

All parts of the used software were open-source, the operating system Raspbian and Ubuntu, the development environment Python, pudb and the libraries OpenCV, ArUco, Matplotlib, etc. Thanks to the open components (open source software, open hardware) everybody can easily implement the introduced technology.

The two small open-source program packages were released with a simple but user friendly Command Line Interface (CLI), which gives the highest flexibility to use them

in other projects. We encourage everybody to use, improve and extend the programs introduced in this paper.

The developed methods, algorithms, Python codes can be improved in number of points. Our work is in progress, we have plans, ideas for the future.

Acknowledgement

Support of Grant BME FIKP-VÍZ by EMMI is kindly acknowledged.

References

Brunelli, R. (2009). *Template Matching Techniques in Computer Vision*, John Wiley & Sons Ltd, United Kingdom

Engel, P., Schweimler, B. (2016). Development of an Open-Source Automatic Deformation Monitoring System for Geodetical and Geotechnical Measurements, *ISPRS Journal of Photogrammetry and Remote Sensing* XL-5/W8.25-30**.** DOI: 10.5194/isprs-archives-XL-5-W8-25-2016

Find-GCP project home page and source code (2020). Available at https://github.com/zsiki/Find-GCP

Garrido-Jurado, S., Muñoz Salinas, R., Madrid-Cuevas, F. J., Marín-Jiménez, M. J. (2014). Automatic generation and detection of highly reliable fiducial markers under occlusion, *PatternRecognition* **47**, 2280–2292. DOI: 10.1016/j.patcog.2014.01.005

Hashemi, N.S., Aghdam, R.B., Ghiasi, A.S.B, Fatemi, P. (2016). Template Matching Advances and Applications in Image Analysis, *American Scientific Research for Engineering, Technology, and Science* **26**(3), pp. 91-108

Kovács, N., Kövesdi, B., Dunai, L., Takács, B. (2016). Loading Test of the Rákóczi Danube Bridge in Budapest, *Procedia Engineering* **15**, pp. 191-198

OpenCV contributors (2020). OpenCV ArUco Marker Detection, available at https://docs.opencv.org/master/d9/d6a/group__aruco.html

Pajankar, A. (2020) *Raspberry Pi Computer Vision Programming – Second Edition*, Packt Publishing Limited, Birmingham

Roy, P., Dutta, S., Dey, N., Dey, G., Chakraborty, S., Ray, R. (2014) Adaptive Thresholding: A comparative study, *2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)*, pp. 1182 – 1186. DOI: 10.1109/ICCICCT.2014.6993140

Shilkrot, R., Escrivá, D. M. (2018). *Mastering OpenCV 4 – Third Edition*, Packt Publishing Limited, Birmingham

Siki, Z., Takács, B., Égető, Cs. (2018). Ulyxes and open source project for automation in engineering surveying, *PEERJ PREPRINTS* **6,** Paper: e27226v1 DOI: 10.7287/peerj.preprints.27226v1

Ulyxes, (2020). Ulyxes project home page Available at http://www.agt.bme.hu/ulyxes/

Völgyesi, L., Tóth, Gy. (2020). Calibration of CCD sensors mounted on geodetic measuring system, *Surveying Review* **52,** pp. 1-10. DOI: 10.1080/00396265.2019.1703506

Received August 31, 2020, revised January 12, 2021, accepted January 12, 2021