Michał **BUGAŁA**

OBJECT ORIENTATION RECOGNITION USING THE OPENCV LIBRARY

Abstract. The article presents the basic methods of detecting objects in an image from a digital camera and attempts to apply methods of determining the orientation of an object.

Keywords: augmented reality, OpenCV library, image analysis, image processing, 3D graphics.

1. INTRODUCTION

OpenCV (*Open Source Computer Vision*) is a library of functions used in image processing and based on open source code and initiated by Intel Corporation. It includes about five hundred functions which deal with such areas as: medical imaging, robotics, camera calibration, user interface, quality control. As learning systems[1] are often applied in specialized image processing, the OpenCV library includes an MML (Machine Learning Library) sublibrary devoted to pattern and cluster[2] recognition [1].

Detecting an object and recognizing its orientation in 3D was initiated by studying the basic methods of image comparison. Fast and accurate (two-dimensional) image comparison method is crucial for further calculations and transformations of results into the three-dimensional space we are interested in.

Figure 1 shows an object (a box) known a priori, the representations of which under various angles were used as templates for comparison with the real image from camera (Figure 2) and for analyzing selected methods.

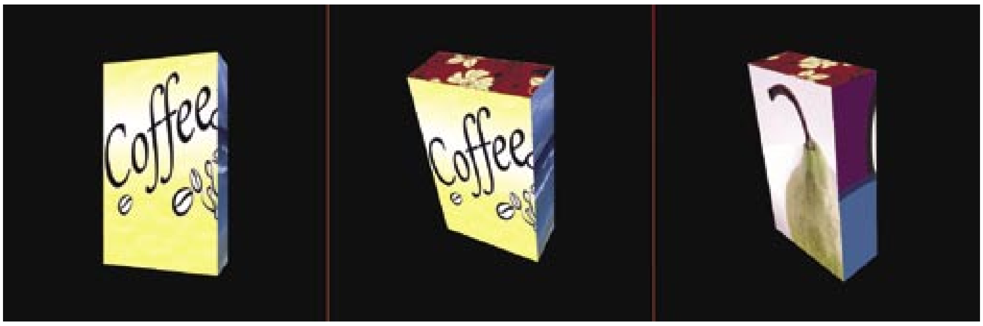

Fig. 1. Examples of comparative templates (a virtual object in a 3D environment depicted under various angles)

[1] Machine learning – a discipline of sciences dealing with the problems of artificial intelligence [2].

[2] Cluster - a set of elements of similar characteristics.

Michał **BUGAŁA** – Ośrodek Badawczo-Rozwojowy Urządzeń Mechanicznych „OBRUM” sp z o.o., Gliwice

2. METHODS

The basic methods included in the OpenCV library were studied and compared to determine the most appropriate, i.e. fast and accurate. The methods were subjected to an experiment that consisted in comparing about 1200 various templates (virtual object depicted at various angles) with a camera image and finding the best fit and thus the approximate orientation of the real object.

2.1. Template matching

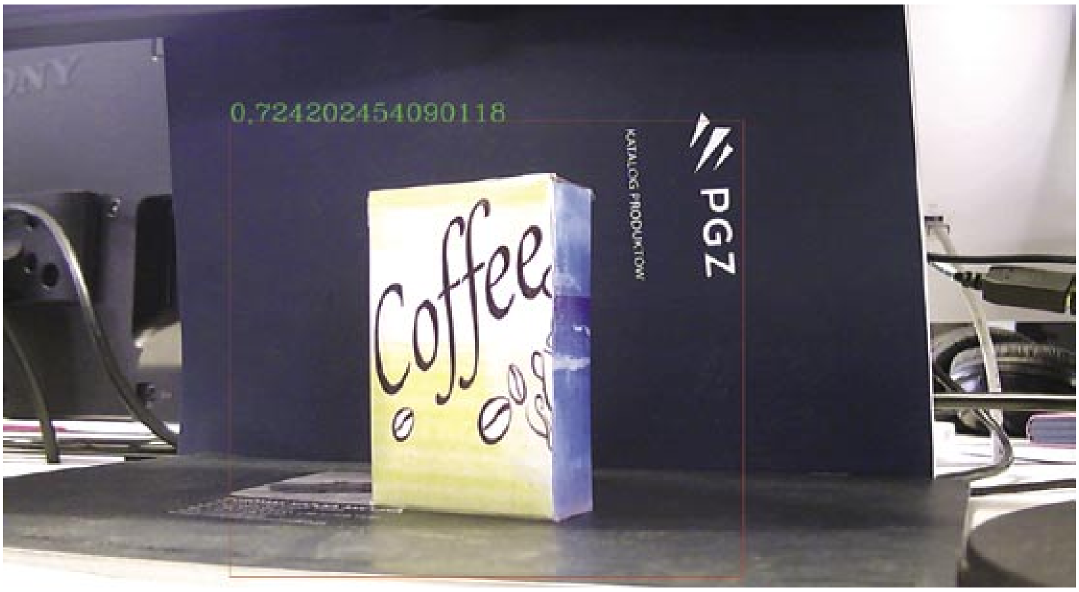

Fig. 2. An example of a camera image with an outline of the comparative template

Template matching is effected by means of the *cvMatchTemplate* function of the OpenCV library. The advantage of the template matching method is the ability to obtain information on the 2D position of template in the camera image and the fit coefficient, which in this case approaches 1 (perfect fit). The disadvantage of this method is long computation time.

Short summary of the method:

- advantages:
  - the function returns the fit coefficient,
  - ability to acquire information on 2D position of the template in the camera image,
  - accurate method,
- disadvantages:
  - long computation for a single fit (≤62 ms for images of the following dimensions: 1280x720 pixels (camera), 614x563 pixels (template).

2.2. Comparison of shapes

Shape comparison is effected by means of the *cvMatchShapes* function and images prepared beforehand (Fig. 3). The *cvMatchShapes* function uses Hu moments [3] and does not provide satisfactory results in terms of accuracy. A lot of effort would also be required to separate the object from the background and compare images of the same size[3].

[3] The function compares shapes, rather than fits them.

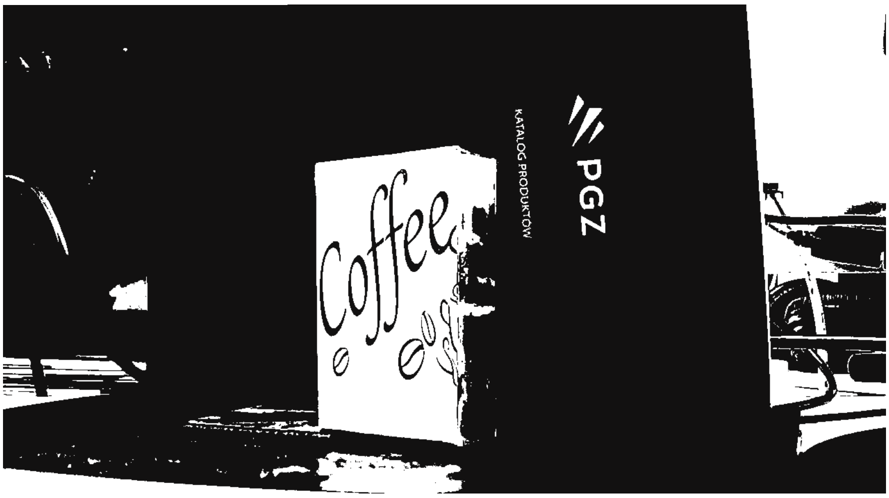

Fig. 3. Image from camera with the shape separated

Short summary of the method:

- advantages:
  - the function returns the comparison coefficient,
  - short computation time (≤16 ms for images of the following dimensions: 1280x720 px and 1280x720 px),
- disadvantages:
  - the method is not accurate and requires proper preparation of images (suitable for analyzing less complex shapes).

2.3. Comparison of contours

Contours are compared by means of the *cvMatchShapes* function and the *cvFindContours* function which determines contours. As was the case of shape comparison, this method has not proved too accurate, despite relatively well-prepared images (Fig. 4).

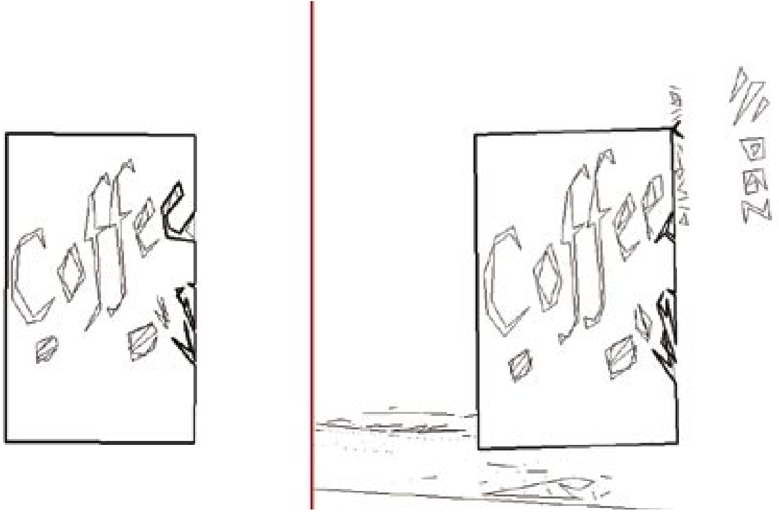

Fig. 4. Comparison of template contours (left) and camera image

Short summary of the method:

- advantages:
  - the function returns the comparison coefficient,
  - short computation time (≤16 ms for images: 1280x720 px and 1280x720 px),
- disadvantages:
  - the method is not accurate and requires proper preparation of images.

2.4. Comparison of skeletons

Comparison of skeletons (Fig. 5) is a proposal for the preparation of images for analysis and can be the starting point for the chamfer matching method [4], not discussed here, in which the template (contours, or skeleton in this case) is matched with the image processed by the distance transform or map algorithm [5] - Fig. 6.

Skeleton comparison was based on template matching *cvMatchTemplate* and contour comparison *cvMatchShapes*, but no improvement of results was observed after skeletonization[4].

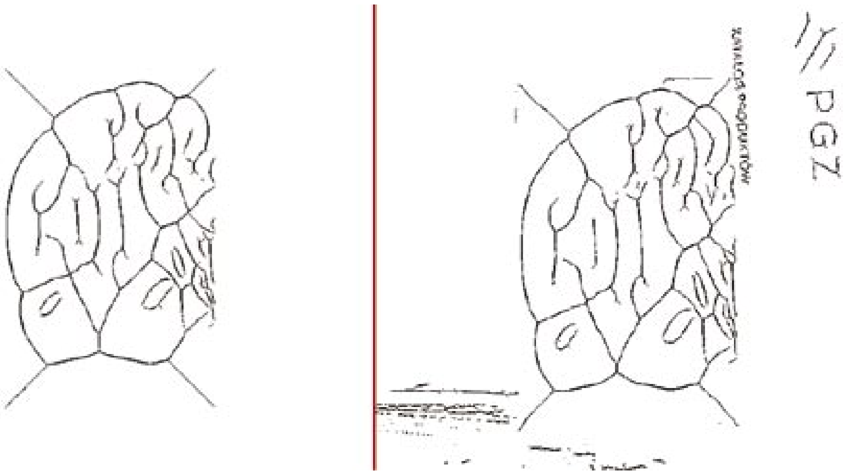

Fig. 5. Comparison of template skeleton (left) and camera image

Short summary of the method:

- advantages:
  - the function returns the comparison coefficient,
  - computation time depends on contour comparison,
- disadvantages:
  - computation time depends on the applied method of contour matching,
  - the method is not accurate (also when *cvMatchTemplate* function is used) and requires proper preparation of images (skeletonization).

[4] Skeletonization is the operation of determining linear representation of figures on the analyzed image [6].

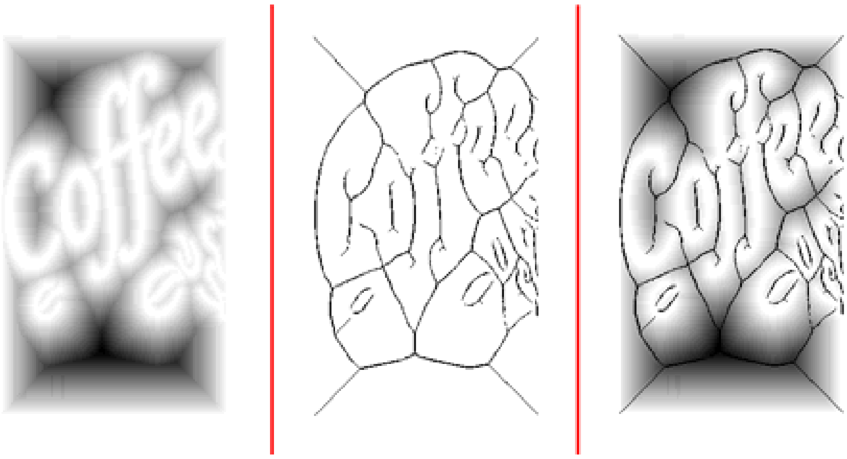

Fig. 6. Concept of chamfer matching with the use of distance map of the camera image (left) and template skeleton (middle)

2.5. Comparison using the SURF method

The SURF (Speeded-Up Robust Features) method can be used because the OpenCV library includes the *cvExtractSURF* function. The method consists in matching robust features of the template with those of the camera image [7] - Fig. 7.

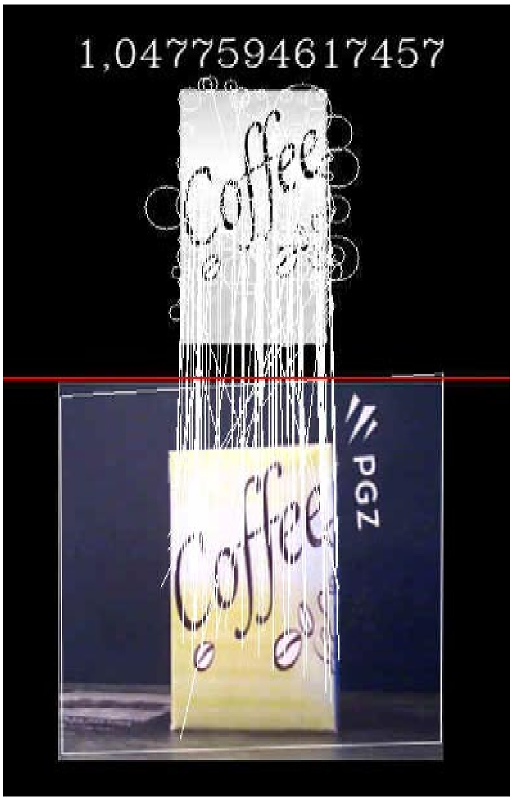

Fig. 7. SURF visualization

Short summary of the method:

- advantages:
  - the function returns the comparison coefficient,
  - acceptable computation time (≤30 ms for images: 307x281 px and 307x281 px),
  - ability to acquire information on 2D position of the template in the camera image,
- disadvantages:
  - direct calculation of position and orientation in 3D space is not possible (PnP problem for points lying in one plane - the OpenCV 2.4.11 library does not contain a ready solution for this problem),
  - the method is relatively accurate (large variation of results in repeated comparisons).

2.6. Characteristic pair matching

Comparison of images by matching characteristic pairs is a proposal for finding the orientation of objects in recorded images. The method consists in matching sections of a template with the most characteristic features with the camera image. The entire template is analyzed by the *cvGoodFeaturesToTrack* function of the OpenCV library, where the said function finds characteristic features (best locations for tracking) of the represented 3D object (template).

The characteristic points (features) are shown in Fig. 8, where the sections are taken from the nearest vertex of the 3D object[5] - at a later stage this will facilitate the computation of object orientation.

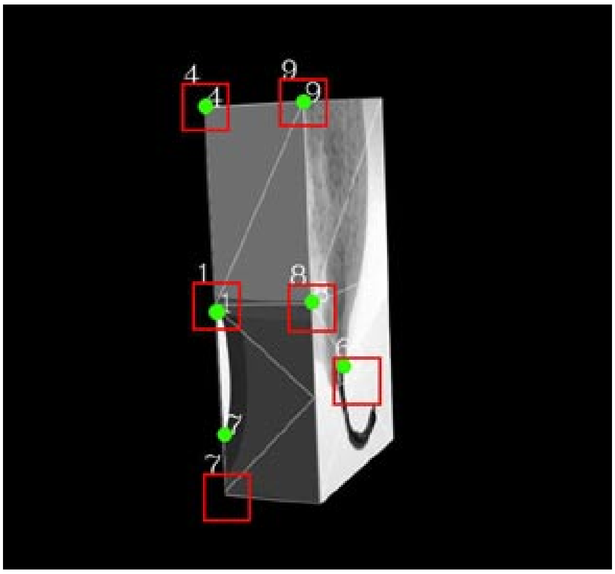

Fig. 8. Template with indicated section outlines, 3D object network and characteristic points

In order to accelerate computation, the template sections, information on the 3D object vertices and 2D positions of the sections are saved to a hard disk and there is no need to re-analyze the object. In addition the template sections were reduced to 16x16 px and processed with the edge detection algorithm *cvCanny* (OpenCV library). The size of sections can be freely modified and thereby the quality to speed ratio can be controlled.

[5] The pair matching method may be extended by enabling addition of a vertex to the 3D object at its most characteristic point, rather than picking the nearest vertex.

After proper data preparation, the algorithm is reduced to the standard process of matching template sections to the image using the *cvMatchTemplate* function (Fig. 9), removing the wrong matches and selecting the best pairs for the current template.

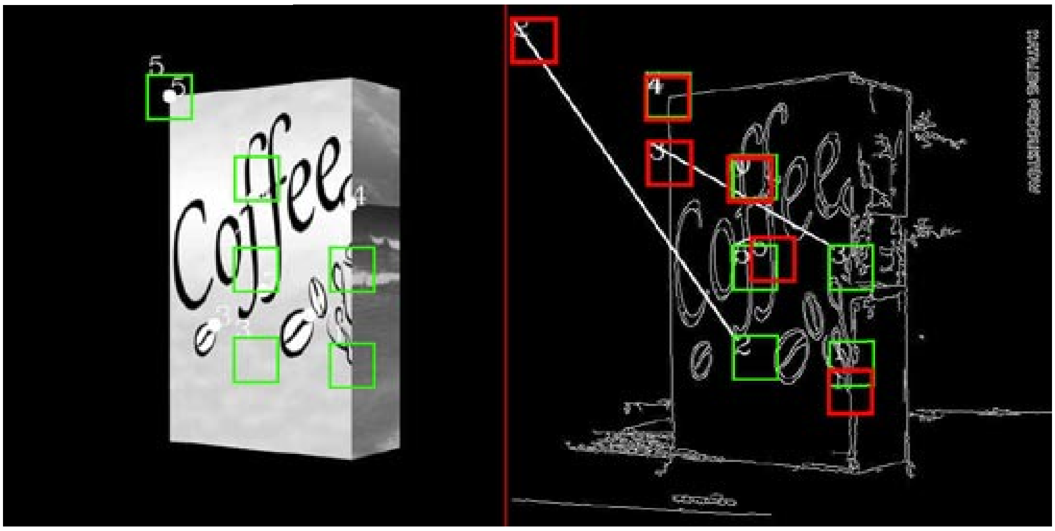

Fig. 9. Visualization of matching template sections (left) with the camera image (numbers of sections in the left and right image are not conclusive)

Fig. 10 shows how the algorithm works in the case of a non matching template.

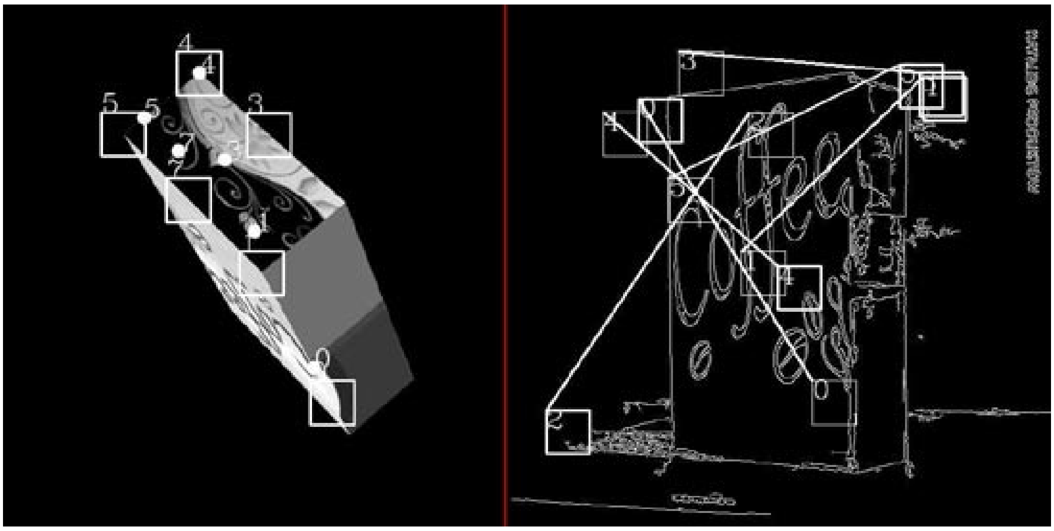

Fig. 10. Visualization of matching sections (left) in the case of a non-matching template (the camera image (substantial displacement of pairs on the right side of the image)

Short summary of the method:

- advantages:
  - the function returns the fit coefficient,
  - ability to acquire information on 2D position of the template in the camera image,

  - position and orientation in 3D space can be computed (*cvPOSIT* function[6]),
  - accurate method,
- disadvantages:
  - short computation time (<=16 ms for six sections 16x16 px and camera image 307x281 px),
  - need to create a data table (this, however, has no impact on the speed of the method).

3. CONCLUSIONS

The methods of comparing shapes and contours are fast but not very accurate for highly complex images. The SURF method and the proposed method of matching characteristic pairs have acceptable execution times and are accurate. The most time-consuming method was the first of the analyzed methods – basic method of template matching.

The studied methods of matching and comparing images contained in the OpenCV library enable, with varying effects, determining the orientation of the objects recorded in the image, but none of them can be used in real time (in the form presented). Analysis of hundreds of templates (virtual objects depicted at different angles) cannot be performed during a time shorter than 40 ms (40 ms = 25 fps), so as to ensure smooth playback of the camera image while calculating the object orientation.

The method of characteristic pair matching also cannot be used, but it can be useful in the first step of tracking an object for determining the starting position, which means that it allows finding the orientation of the 3D object and assigns the vertices of the object (3D points) to the 2D points in the camera image (this enables direct calculation of object position and orientation using the *cvPOSIT* function in the OpenCV library). In the next step of implementing a method of real-time determination of object position and orientation (with object scale and image perspective taken into account), use should be made of motion tracking techniques and of more advanced functions included in the MML sublibrary.

Image analysis and augmented reality are those domains of computer science that have a very high potential for development and for application in military and civilian industries. The scope of issues related to the augmented reality that can be explored and analyzed is still enormous and awaits new, brilliant solutions.

4. REFERENCES

[1] Bradski G., Kaehler A.: Learning OpenCV, O'Reilly Media, 2008, ISBN: 978-0-596-51613-0.

[2] Mitchell M. T.: Machine Learning, 1997, McGraw-Hill Science, ISBN: 978-0070428072.

[3] Hu M. K.: Visual Pattern Recognition by Moment Invariants, http://www.sci.utah.edu/~gerig/CS7960-S2010/handouts/Hu.pdf [Retrieved: 15.08.2016].

[6] The *cvPOSIT* function returns information on the position and orientation of an object when the positions of vertices (3D points) and of the corresponding 2D points in the camera image are known (PnP problem for points not lying in one plane) [1].

[4] Mitsubishi Electric Research Laboratories: Fast Directional Chamfer Matching, http://www.merl.com/publications/docs/TR2010-045.pdf [Retrieved: 04.04.2016].

[5] Mitsubishi Electric Research Laboratories: Adaptively Sampled Distance Fields: A General Representation of Shape for Computer Graphics, http://www.merl.com/publications/docs/TR2000-15.pdf [Retrieved:15.08.2016].

[6] Tadeusiewicz R, Korohoda P.: Komputerowa analiza i przetwarzanie obrazów, 1997, Wydawnictwo Fundacji Postępu Telekomunikacji, ISBN: 9788386476152.

[7] OpenCV 2.2 documentation: Feature detection and description, ttp://opencv.jp/opencv-svn_org/c/features2d_feature_detection_and_description.html [Retrieved: 04.04.2016].

[8] Lepetit V., Moreno-Noguer F., and Fua P.: EPnP: Efficient Perspective-n-Point Camera Pose Estimation, https://infoscience.epfl.ch/record/160138/files/top.pdf?version=1 [Retrieved: 15.08.2016].