ARCHIVES OF MECHANICAL TECHNOLOGY AND MATERIALS
Vol. 35 2015

ANDRZEJ MILECKI*, ŁUKASZ SAWICKI**

APPLICATION OF VISUAL SYSTEM FOR HAND CONTROL OF 3D MANIPULATOR

The article concerns controlling of a 3-axles manipulator by hand movement, recognized by visual system. In the investigations the Kinect sensor was used, which enabled the acquiring of 2D hand images with depth parameter. The OpenCV library was used for recognition of hand position in a real-time computer based control system. The control program was written in C++ language, which allows the processing of 15 frames per second. The proposed control system enabled to operate the manipulator with the same frequency.

Key words: visual control, manipulator, control, OpenCV, gesture recognition.

1. INTRODUCTION

Nowadays a vast majority of robots are used in factories, but there is a need to use specialized robots in non-manufacturing industries such as agriculture, health care, entertaining and other services. In the personal or service robotics domain the application of special interaction between human and robot is crucial. During the last 50 years many human friendly interfaces for communication with robots have been made and investigated. The example is the keyboard or the mouse used to transfer operator movements to robot controller. This method is now commonly used in so called *teach-in* mode. The interesting idea is to transfer the human thoughts to controller, however this method is still under early development.

Scientists do not stop searching for new ways to communicate with machines and robots. Various sensor modalities including touch, vision and hearing a robot have been built [8], which in some cases displayed intelligence and communicates in a user-friendly way. The idea of using video cameras for robot control is more and more interesting thanks to the development of low-cost, high-quality digital cameras and advances in computational processing enabling real-time control. In visual control of servo drives two methodologies, namely, position-based control [9] and image-based control [10] are traditionally used. The first one uses vision to estimate the absolute position of the robot endings and calculates the position error, which is used in the control algorithm.

* Institute of Mechanical Technology, Poznan University of Technology, andrzej.milecki@put.poznan.pl,
** Institute of Mechanical Technology, Poznan University of Technology, lukasz.w.sawicki@doctorate.put.poznan.pl

The second method computes the error directly in the image plane of the camera and therefore, it is less sensitive to kinematic and calibration errors.

In the year 2010 Microsoft introduced Xbox 360 with Kinect as a sensor, which enabled to control of the games, by the human body language. This was achieved thanks to putting into one Kinect box inter alia an infrared emitter and infrared camera [11]. These two elements are used for three dimensional tracking. With the Kinect sensor also a software development kit from Microsoft appeared on the market. It allows to use C++, C# or Python language to program the usage of Kinect [3]. However, there are some limitations of their use. One of them is lack of compatibility with a popular library called OpenCV, which is a free computer vision library for C/C++ programmers available for Windows, Linux, Android and iOS. OpenCV was designed for computational efficiency improvement necessary in real-time applications [2].

This article is about a method, in which the operator may control the robot by hand movements. In the investigations described in this paper, the Kinect hardware and OpenCV software were used. The main issue presented here, was to compute hand movements in 3D using Kinect sensors which is discussed in many papers. However among authors of the available literature the application of the described in the paper algorithm was not found.

2. GESTURE RECOGNITION

The scheme of the presented in this paper manipulator control system is shown in Fig 1. The operator's image was obtained by Kinect's infrared camera,

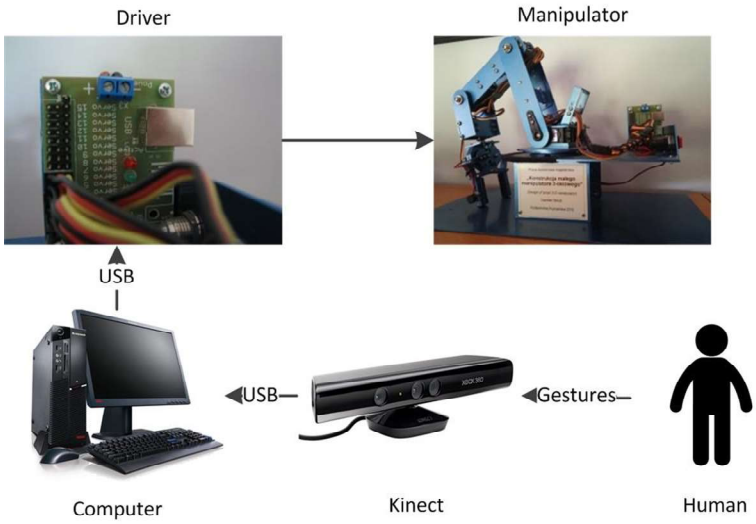

Fig. 1. Manipulator's control scheme

which has delivered a 3D monochrome video stream to the control computer. The received 3D data are processed by the PC on which a special software was running. In order to make the gesture recognition possible, several functions from OpenCV library were used. The obtained 3D coordinates are sent to the small robot controller.

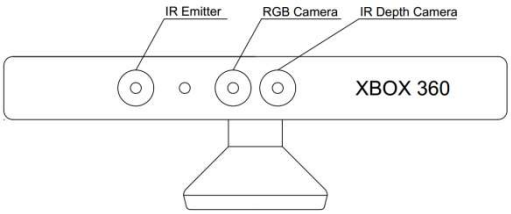

Fig. 2. Kinect sensor

The use of infrared depth camera for receiving three dimensional images, seems to be much easier than the use of stereo camera [13] or two ordinary RGB cameras spaced from each other by a known distance. Another advantage of the use of an infrared camera is that the lighting variations do not effect on the calculation result [14].

The IR emitter emits the infrared structured light which reflection is received by the camera and is recorded. Later on, the obtained picture is converted into a grayscale image where the value of grayscale is consistent with the pixels distance from Kinect (Fig. 2). The image has a resolution of 640x480 pixels [11]. The range of the measured distances by the infrared camera was from 0.7 to 6 meters and had a viewing angle of 43° vertically and 57° horizontally. The Kinect's depth camera generates unsigned 16 bit numbers describing every pixel distance from the sensor. In order to use the OpenCV library these numbers were converted to a 8 bit [1].

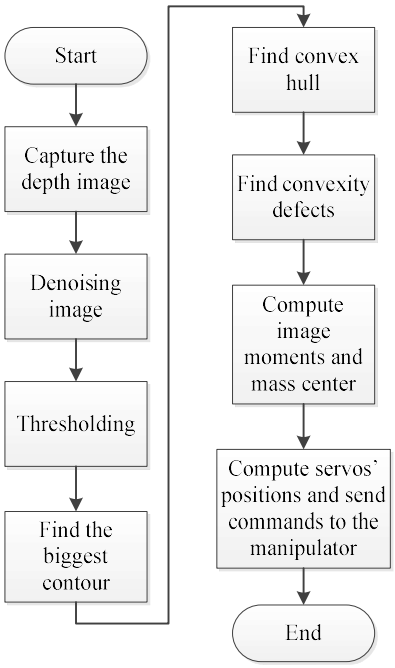

Fig. 3. Scheme of the algorithm

The scheme of the algorithm used in the presented here research is shown in Fig. 3. This approach is quite simple and therefore the control of the manipulator can be made in real time. In the first step of the obtained depth image, the noise reduction is performed, which is sometimes called *picture opening* [7]. To this end, the morphological transformations were used, which removed some of the noises from the picture. Two functions are applied *erode*()

and *dilate*() [2], in this strict order. This arrangement is significant, because these operations are not convertible [7]. During the erosion procedure, the image regions were reduced and the protrusions were smoothed. The dilation is a converse operation, during which the boundaries of the foreground details were increased and holes within those details were decreased.

It was assumed that the operator uses one of his hands to control of the manipulator, and that the hand is in front of the rest of his body and head. Therefore, the algorithm was searching only for the nearest part to the Kinect. This part (here the hand) should be found and used to the control of the manipulator. Of course, there are some safety measures undertaken, to prevent the program of tracking small objects that are close to the sensor, but they are not a hand like. That's why the Kinect workspace should be reduced. To this end, a segmentation method based on a threshold was used. This method performs a comparison of each pixel intensity value, with respect to a threshold and eliminates all pixels which distance to the sensor is bigger than the threshold, which was assumed in this case about one meter. This classification is called in OpenCV library as *Threshold to Zero, Inverted* [2]. Following equation was used:

$$\mathrm{dst}(x,y) = \begin{cases} 0 & \text{if } \mathrm{src}(x,y) > \mathrm{threshold} \\ \mathrm{src}(x,y) & \text{if } \mathrm{src}(x,y) \le \mathrm{threshold} \end{cases} \tag{1}$$

where: x, y – pixel coordinates, scr – source function (grayscale values in x, y point).

Thanks to it, any pixels above the threshold are regarded as irrelevant i.e. equal to 0 and only those pictures which are under this limit were used in the next steps of the procedure. This procedure finds all closed contours in a picture. As a result a list of points creating curves in an image is obtained, which in fact are contours of the objects found in a picture. OpenCV created it using obtained edges and connecting them to receive a closed object. Contours are like boundaries of regions in an image [1].

Then, an OpenCV function called *findContours()* was used for counters recognition [2]. This function converts at first image to grayscale, blur it, detects edges using canny and finally draws all contours. In the next step the algorithm was searching for the biggest contour, which will be used in further processing. The final result of this part of the algorithm is presented in Fig. 4; it is a black contour around the hand. The next used function was *convexHull()* [2]. The convex hull of a set A of points in the Euclidean plane is the smallest convex set that contains X. Formally, the convex hull may be defined as the intersection of all convex sets containing X or as the set of all convex combinations of points in X. A set of points is defined to be convex if it contains the line segments connecting each pair of its points.

Fig. 4. Picture after contures recorgnition

Fig. 5. Hand convex hull

Convexity defects are the differences between the hull and the contour. In the Fig. 5 the convex hull is shown as a dark line around the hand. OpenCV library uses the Sklansky's algorithm developed in 1982 to find the convex hull [12]. At first the algorithm finds extreme apexes horizontally and vertically. Later on, the plane is divided by using lines that are connecting vertices, which had been founded earlier. The goal is to receive four monotone chains within four regions which were procured after the previous step. By connecting these chains the algorithm obtained the convex hull.

In the next step the function *convexityDefects()* was used [2]. It finds all convexity defects of the input contour and returns a sequence of vectors. They give information to the algorithm how many differences are between the convex hull and the contour. Based on this number the algorithm knows when the hand is open or closed.

Image moments generally describe contours by summing up all of the pixels within them. They are mostly used to compare two contours in order to find a matching pair. In the algorithm the function *moments*() was used to procure a central point of the hand to control the manipulator. To the control instead of using the exact contour of the hand, its convex hull was used. It was because it is more stable and less susceptible to changes in the image [1]. At first spatial moments were computed using the following equation:

$$m_{ij} = \sum_{x,y} \left(array\,(x,y) * x^j * y^i \right) \tag{2}$$

where: array (x, y) – pixel coordinates of the contour.

In the next step the hand convex hull centre coordinates were calculated using equations:

$$x_c = \frac{m_{10}}{m_{00}} \ , \ y_c = \frac{m_{01}}{m_{00}} \tag{3}$$

where: x_c, y_c – coordinates of the mass centre of the convex hull, m_{00} – moment 00 (contour length), m_{10} – moment 10 (for X coordinates), m_{01} – moment 01 (for Y coordinates).

The third coordinate z_c was obtained by getting the grayscale value in the image in the mass centre of the convex hull. As it was mentioned this value is consistent with the distance from the Kinect to the hand.

3. MANIPULATOR CONTROL

As a result of previously described procedures values of three coordinates of the centre point of a hand were obtained. They are tracked in three dimensions and used to control the manipulator.

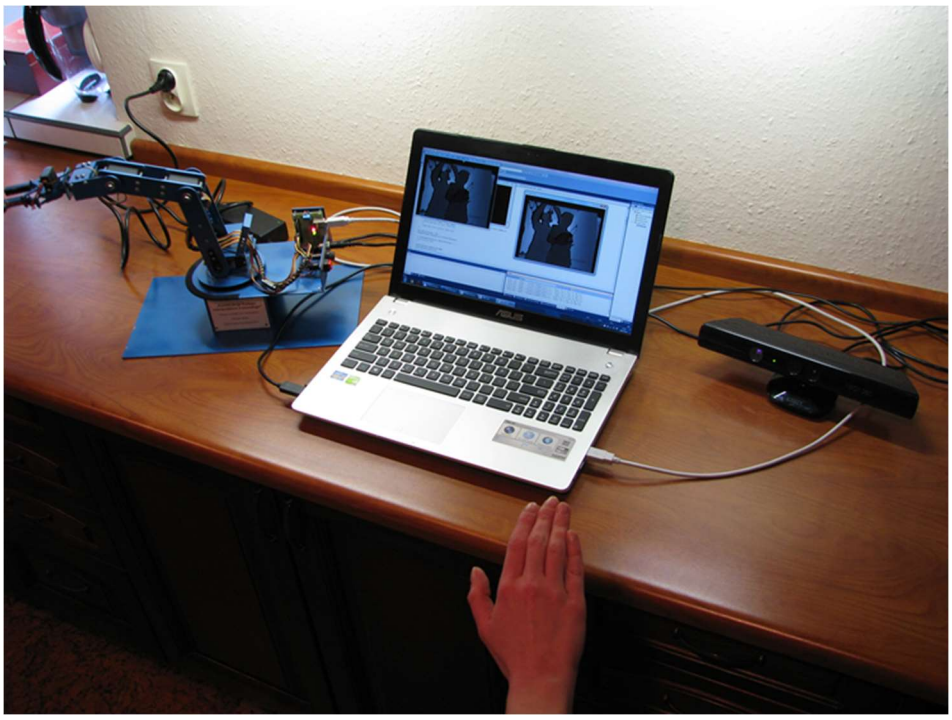

Fig. 6. Picture of the control system and manipulator

The earlier presented algorithm was tested with an anthropomorphic manipulator with seven servo motors from which there were controlled six. The manipulator is equipped with five servos with a torque of 0.92 Nm which are used to move the main arm and two with a torque of 0.26 Nm for the gripper. Its lifting capacity is 0.05 kg and the range of motion is up to 300 mm with a maximum rotation of 200°. In order to control the motors a servo controller is used which allows to steer with accuracy of 16000 steps via a PWM signal. The controller is connected via USB with a computer and accepts commands and data in the form of characters from the ASCII table.

Servos require absolute position values which were computed in accordance with the following equations:

$$Xc = \frac{x_c}{Image\ width\ (640\ px)} * Servo's\ resolution(16000) \tag{4}$$

$$Yc = \frac{y_c}{Image\ height\ (480\ px)} * Servo's\ resolution(16000) \tag{5}$$

$$Zc = \frac{z_c}{Threshold\ in\ grayscale(45)} * Servo's\ resolution(16000) \tag{6}$$

where: x_c, y_c, z_c – coordinates of the mass centre of the convex hull, *Xc, Yc, Zc* – values send to the servo controller.

In order to control the manipulator three coordinates obtained from the depth image were used. The coordinates are linked with specific servo motors in the way shown in Fig.7. The horizontal value x_c is tied with the motor in A1, the depth coordinate z_c with the two servos in B2, the vertical value y_c with two

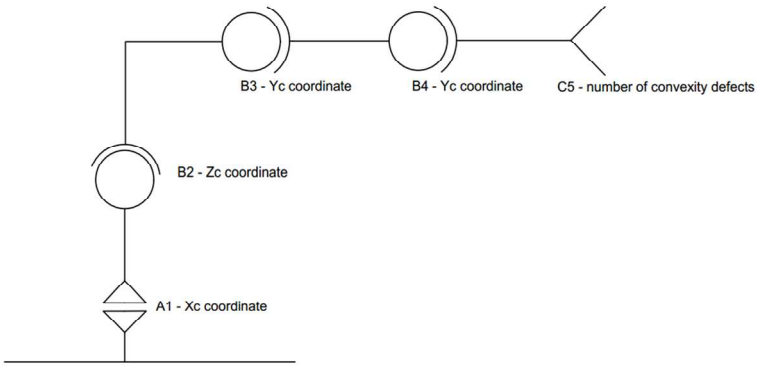

Fig. 7. Kinematic scheme of the manipulator

motors, namely B3 and B4.

Thanks to this approach, the manipulator always “knows”, where the hand is even when the hand is moving faster than the manipulator can. In such a case

a few intermediate positions are skipped and the arm pursues to the next actual hand position point without any break.

The last task of the controller was to control of the gripper. To this end a closing of opening of the operator's hand was used. When the operator hand is closed, the gripper should also be closed and when the hand is open, the gripper should be open too. In order to recognize the opening or closing of the hand, both the threshold and the number of convexity defects are used. The defects defined when the hand is open or closed. When it is closed, surprisingly the number of defects is bigger than if it is open. The reason of it is, that the contour of the hand has many small defects when is closed, and OpenCV detects all of them even though the human eye cannot see them in the image. When the hand is open, the amount of defects is smaller, because we have simply few big defects instead of many smaller. This method is far from being perfect but it worked very well in our control algorithm.

4. CONCLUSION

Algorithm presented above allows to control the manipulator in almost real time by using the depth image acquired from the infrared camera in the Kinect. Demonstrated approach has many advantages such as simplicity, three dimensional tracking, stability, speed and the fact that light variations do not affect the final result. It was achieved mostly thanks to using the depth camera instead of a regular RGB camera which requires a complex algorithm and large computer resources in order to obtain the same result.

REFERENCES

[1] **Bradski G., Kaehler A.,** "Learning OpenCV", O'Reilly Media, Inc. 2008
[2] Opencv documentation. [Online]. Available: http://docs.opencv.org
[3] Kinect for Windows. [Online]. Available: http:// www.microsoft.com/en-us/kinectforwindows/
[4] **Van Le B., Nguyen A. T., Zhu Y.,** "Hand Detecting and Positioning Based on Depth Image of Kinect Sensor", International Journal of Information and Electronics Engineering, May 2014
[5] **Murata T., Shin J.**, "Hand Gesture and Character Recognition Based on Kinect Sensor", International Journal of Distributed Sensor Networks Volume 2014
[6] **Hervé L., Lichti D.,** "Real-time hand gesture recognition using range cameras." Proceedings of the Canadian Geomatics Conference, Calgary, Canada, 2010
[7] **Tadeusiewicz R., Korohoda P.**, Komputerowa analiza i przetwarzanie obrazów", Wydawnictwo Fundacji Postępu Telekomunikacji, Kraków 1997
[8] **M. A. Abidi, R. O. Eason, and R. C. Gonzalez**, "Autonomous robotic inspection and manipulation using multisensor feedback," Computer, vol. 24, no. 4, pp. 17–31, Apr. 1991.
[9] **D. Surdilovic and J. Kirchhof**, "A new position based force/impedance control for industrial robots," in , 1996 IEEE International Conference on Robotics and Automation, 1996. Proceedings, 1996, vol. 1, pp. 629–634 vol.1.

[10] **D. Kulas, J. Wang, M. Obeng, and X. Wu**, "Image-based path following and target tracking control for a mobile robot," in 2013 Proceedings of IEEE Southeastcon, 2013, pp. 1–6.
[11] **T. B. Blair and C. E. Davis**, "Innovate engineering outreach: A special application of the Xbox 360 Kinect sensor," in 2013 IEEE Frontiers in Education Conference, 2013, pp. 1279–1283.
[12] **J. Sklansky**, "Finding the convex hull of a simple polygon," Pattern Recognition Letters, vol. 1, no. 2, pp. 79–83, Dec. 1982.
[13] **C.-Y. Lin and E. Setiawan**, "Object orientation recognition based on SIFT and SVM by using stereo camera," in IEEE International Conference on Robotics and Biomimetics, 2008. ROBIO 2008, 2009, pp. 1371–1376.
[14] **B. Y. L. Li, A. S. Mian, W. Liu, and A. Krishna**, "Using Kinect for face recognition under varying poses, expressions, illumination and disguise," in 2013 IEEE Workshop on Applications of Computer Vision (WACV), 2013, pp. 186–192.
[15] **H. Wu, M. Su, S. Chen, Y. Guan, H. Zhang, and G. Liu**, "Kinect-based robotic manipulation: From human hand to end-effector," in 2015 IEEE 10th Conference on Industrial Electronics and Applications (ICIEA), 2015, pp. 806–811.

STEROWANIE MANIPULATOREM W TRZECH WYMIARACH ZA POMOCĄ GESTÓW

S t r e s z c z e n i e

Artykuł skupia się na sterowaniu 3 osiowego manipulatora wykorzystując obraz głębi pozyskany z sensora Kinect oraz bibliotekę OpenCV, przeznaczoną głównie do obróbki obrazu w czasie rzeczywistym. Opracowany program został napisany w C++ i pozwala na operowanie manipulatorem z prędkością 15 FPS. Komunikacja ze sterownikiem manipulatora została zapewnioną za pomocą kabla USB.

Słowa kluczowe: sterowanie wizyjne, manipulator, sterowanie, opencv, rozpoznawanie gestów.