Object Detection for Pick and Place Industrial Robots Using OpenCV-Python

[1]Harshit S. Badiger,
[1]B.E Student
[1]Department of Electrical & Electronics Engineering,
[1] B.M.S Institute of Technology & Management
Bangalore, India

[2]H.D. Kattimani,
[2]Associate Professor,
[2]Department of Electrical & Electronics Engineering,
[2]B.M.S Institute of Technology & Management
Bangalore, India

[3]Adarsh S. Hegde,
[3]B.E Student
[3]Department of Electrical & Electronics Engineering,
[3]B.M.S Institute of Technology & Management
Bangalore, India

[4]Anshul Kumar,
[4]B.E Student
[4]Department of Electrical & Electronics Engineering,
[4]B.M.S Institute of Technology & Management
Bangalore, India

Abstract : **This paper deals with a method of Object Detection for pick and place automation tasks using open source software. On the one hand, this paper summarizes the basic kinematics, an operation of a Robotic Arm and controlling the Robotic Arm by programming approach. On the other hand, this paper talks about a specific curated object detection algorithm which is used for the above-mentioned automation task. Through various experiments and cases, the object detection algorithm is coupled with a Robot Arm which is programmable externally, and to achieve higher efficiency for the basic automation tasks, and to learn the importance of object detection in computer vision.**

Index Terms **– automation, object detection, computer vision, opencv-python, pick and place, robots.**

I. INTRODUCTION

The increased consumption, awareness of quality, safety has created an awareness for improved quality in consumer products. The demand of user specific customization, increase of competition has raised the need of cost reduction. This can be achieved by increasing the quality of products, reducing the wastage during the production, flexibility in customization and faster production. The human based quality control is not apt after a certain quantity of production. At higher level of production, it is important to have a system that simulate human acts. The vision system can be viewed as simulated system with combination of human eye (Camera) and intelligence (Computer). Machine Vision (MV) is the technology used to provide image-based analysis for applications such as automatic inspection, process control and robot guidance in industry. Vision Sensors/Machine Vision Systems analyze images to perform appearance inspections, character inspections, positioning, and defect inspections. The machine vision systems can be used in a wide range of applications because of their flexibility and versatile features. The use of vision systems in inspection and motion control applications imposes several real-time constraints on image processing. However, constantly increasing performances and decreasing costs of machine vision software and hardware make vision measuring systems more advantageous than the conventional measuring systems. These vision systems can be used to precisely measure variables such as distance, angle, position, orientation, color, etc.

The main advantage of a machine vision-based system is its non-contact inspection principle, which is important in the cases where it is difficult to implement contact measurements. Also, Machine Vision technology helps to achieve better productivity and aids in the overall quality management, thus posing a prominent competition to other industries which do not implement vision systems. The scope of Vison based systems is not only limited to the fields described here and it extends widely to much more industries such as welding industries, where Machine vision is used to identify and classify weld defects in welding environments, where human inspection is not efficient. **.** Moreover, it prevents human contamination of clean rooms and protects human workers from hazardous environments. With the advancements in this field, Computer vision encompasses even to human gait recognition system.

In this paper, we first talk about the kinematics of a Robot which are the basic foundation to study the automation in Robotics [1]. We will then discuss about object detection using OpenCV-Python and then verify its wide applicability on Robot Arm which is programmed using Python 3.5.

II. KINEMATICS OF ROBOT

The need of increased productivity and the delivery of end products of uniform quality, industry is turning more and more towards computer based automation. The inflexibility and generally high cost of these machines, often called high automation systems, have led to a broad-based interest in the use of robots capable of performing a variety of manufacturing functions in a more flexible working environment at lower production costs. With hankering in these, the research and development for a broader filed of work

is dealt with numerous operations on robot kinematics, dynamics, planning systems, sensing, control, programming and machine intelligence.

Robot kinematics [2] deals with the analytical study of the geometry of motion of a robot arm, with due respect to a fixed reference coordinate system without considering the forces/ moments that cause the motion. We'll understand the analytical description of the spatial displacement of the robot as the function of time, in particular the relations of the joint variable space with the position and the orientation of the end-effector of the robot. Computer based robots usually servo-ed in the joint-variable space, whereas objects to be manipulated are usually expressed in world coordinate system.

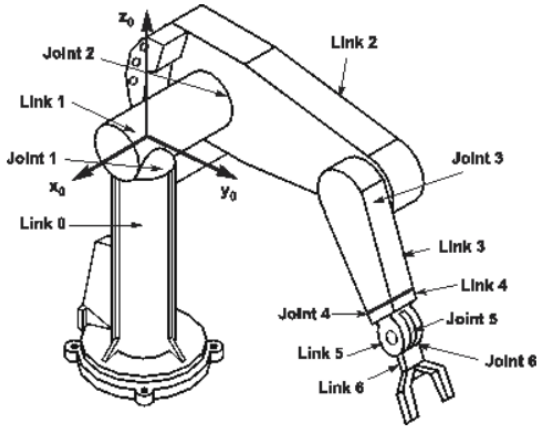

Fig. 1: Forward Kinematics of a Robotic Arm.
Image downloaded from https://medium.com/@sarvagya.vaish/forward-kinematics-using-orocos-kdl-da7035f9c8e

Robot arm dynamics deals with the mathematical formulation of the equations of robot arm motion, its end-effector usage and coordination in them. It enunciates set of the mathematical equations describing the dynamic behaviour of the manipulator. These are useful for the suitable control equations, computer simulation, and evaluation of the kinematic design and structure of a robot arm. These can be done by matrix algebra, iterative or geometric approach. Direct kinematics can be defined as finding the position and orientation of the tool point form the joint angles. Inverse kinematics [3] is the process where joint angles are found out to set, a particular position and orientation of the tool point. For kinematics problem, vector algebra and matrix method is utilized to develop a systematic and generalized approach to describe and represent the location of the links of a robot arm with respect to the reference frame. One frame provides us with one 3X3 matrix. The coordinates for 3X3 matrix does not give any provision for translation and scaling, thus a fourth coordinate is introduced to a position vector [4]. In general, the representation of the N-component position vector by an (N+1) components vector is called homogeneous coordinate representation.

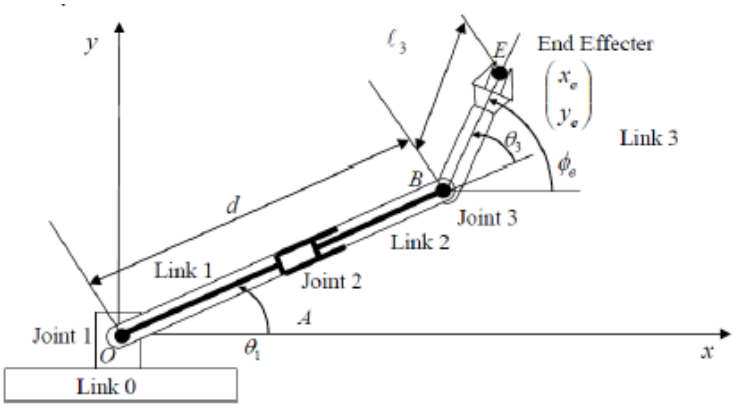

Fig. 2: Linking of individual manipulators to the coordinate axis. Image downloaded from http://www.ent.mrt.ac.lk/~rohan/teaching/ME5144/LectureNotes/Lec%205%20Kinematics.pdf

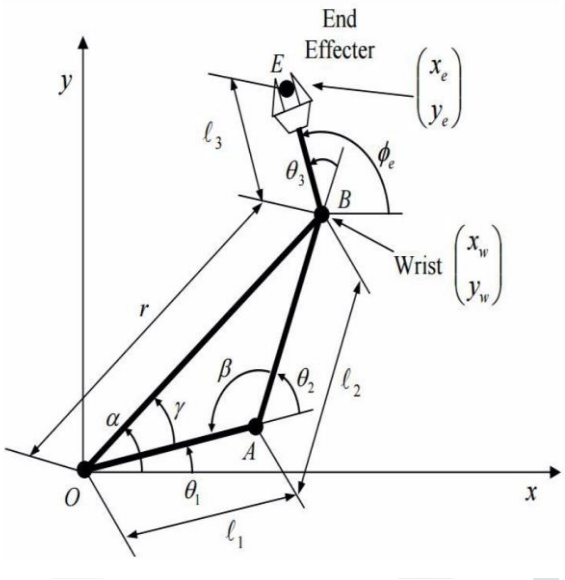

Fig. 3: Inverse Kinematics of a Robotic Arm. Image downloaded from http://www.ent.mrt.ac.lk/~rohan/teaching/ME5144/LectureNotes/Lec%205%20Kinematics.pdf

The manipulators and joint linking variables provide the movement of the robotic arm. It is evidently seen that the mathematical equations that follow the iterations must be used to form the approximate coordinate spatially. This statement is approached by Newton-Raphson method, Newton-Euler method etc. of matrix. Regarding the actual movement the initial calibration is done set to the known point as reference. The base is set for the robot considering the axis and tool for movement. The end effect coordinates is also set in this calibration.

III. Object Detection

Object detection and location in digital images has become one of the most important applications for industries to ease user, save time and to achieve parallelism [5]. The main aim of studying and researching computer vision is to simulate the behavior and manner of human eyes, directly by using a computer and later on develop a system that reduces human need. This method is used because of its strong adaptability and robustness, however, the detection speed needs to be improved, because it requires testing all possible windows by exhaustive search and has high computational complexity. The detection of the objects can be extended using automation and robotics using the image processing techniques that will be easier and more convenient.

IV. Image Processing

Image processing [6] is an operation on image/images to extract some features or useful information. In other words, image processing is a type of signal processing in which the input is an image and the output is useful information or features of the image. Image process has become an integral part in the domain of computer science and computer vision research.

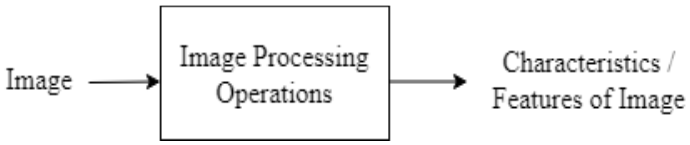

Fig. 4: Concept of Image Processing

4.1 Tools for Image Processing

There are various tools for image processing technique. Some of the known and widely used tools are MATLAB and OpenCV. Both of them have advantages and disadvantages in their own usage. But OpenCV is given an edge over MATLAB for various stated reasons:

- OpenCV is free for commercial usage and the source can be viewed and fixed. Whereas MATLAB is hideously expensive which makes it less usable [7].

- The collection of algorithms available in OpenCV dwarfs everything out there. The library is also optimized for performance. Over 3000 effective optimized algorithms are embedded inside OpenCV.

- OpenCV can be used in desktop application or as the backend of web application. Because of its focus on performance OpenCV (C/C++) is the vision library of choice in many embedded vision applications and mobile apps[8].

- There is a big community of developers (47,000 or so) that use and support OpenCV. Unlike the MATLAB community that consists of researchers, the OpenCV community is a mix of people from many fields and industries. The OpenCV development is funded by companies like Intel, AMD & Google.

- OpenCV can be programmed in C/C++ or Python which makes it much faster than MATLAB. A typical MATLAB program runs many times slower than a C++ program. Built-in MATLAB routines can be very fast, but the code written in MATLAB will usually run much slower [9].

4.2 Steps Involved in Image Processing

Any type of image processing contains of three major steps which results in effective operation namely [10],

Step 1: Importing image/images using image acquisition tools.

Step 2: Applying various image processing operations and manipulating the input using various algorithms in OpenCV or other tools.

Step 3: Extracting useful information or features as output.

V. IMAGE PROCESSING FOR OBJECT DETECTION

Image processing for object detection can be achieved by various methods. One of the convenient way is by using OpenCV-Python. There are various steps involved in image processing using OpenCV-Python which helps in extracting the required useful information. Some of the basic operations done for image processing are stated below [11].

The following block diagram in fig. 5 shows the process of image processing for object detection

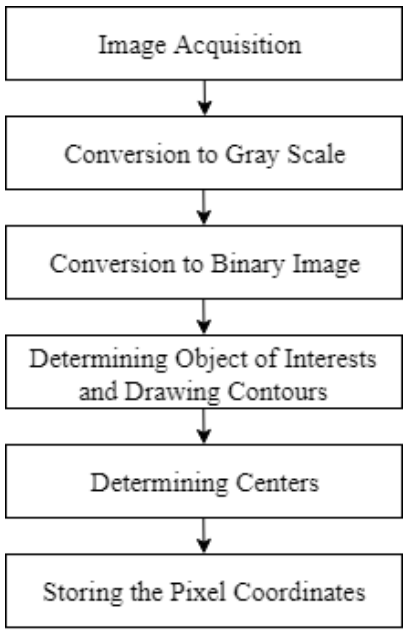

Fig. 5: Block Diagram of Image Processing for Object Detection using OpenCV-Python

5.1 Image Acquisition

The image is acquired using the cv2.imread ("filename", window name) function in OpenCV. The image to be processed be in the same directory as of the other files.

5.2 Conversion to Gray Scale

OpenCV does image processing on gray scale images. So it is necessary to convert the acquired image to a gray scale image using cv2.cvtColor (image, cv2.COLOR_BGR2GRAY) command

5.3 Conversion to Binary Image

In this specific image processing technique the gray scale image further converted to a binary image using thresholding where only blacks and whites are present using the following command.

```
ret,thresh1=cv2.threshold(img,127,255,cv2.THRESH_BINARY)
```

5.4 Drawing Contours around Objects of Interest

The object of interest is determined by approximately determining the pixels of that object and limiting the contour [12] draw function to that particular range.

5.5 Determining the Centers

The centroid of the object of interest around which the contour is drawn is found out by using the moments of images. The obtained pixel coordinates are stored in a python list or dictionary.

VI. Translation of Coordinates to Robotic Arm

By Image processing the centers of the object of interest are obtain. In order to translate these coordinates to the Robotic arm it is necessary to convert the pixel coordinate to Cartesian coordinate or the respective robot coordinate system. We are assuming that we have a rectangular work domain where the picking operation will be executed and the z axis w.r.t the robot will be constant.

6.1 Conversion of Pixel Coordinates to Cartesian Coordinates

In order to convert pixel coordinates to Cartesian coordinates we can use the following formulation. On similar guidelines it can be converted to other coordinate system.

6.1.1 Work Domain Setup

Let us consider a rectangular work domain of any dimension as shown below,

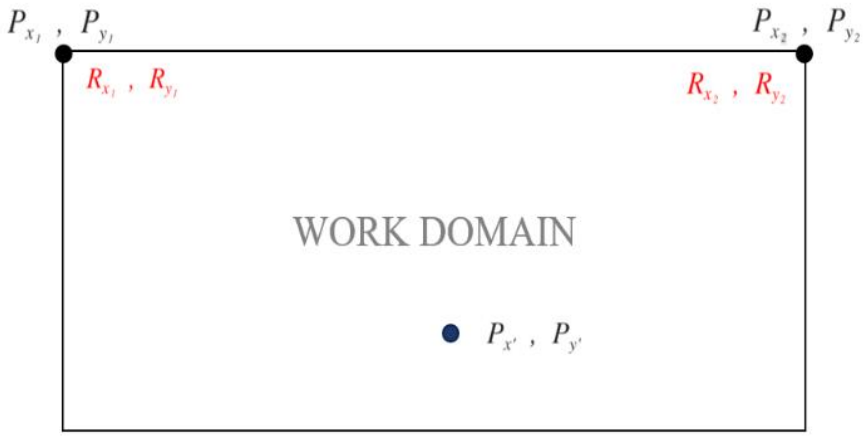

Fig. 6: Work Domain Setup

Let,
P_x = Pixel *x* Coordinate
P_y = Pixel *y* Coordinate
R_x = Robot *x* Coordinate
R_y = Robot *y* Coordinate

Move the robot arm to the respective positions to get the extreme *x* and *y* coordinates.

6.1.2 Tile Width Calculation

Tile width is calculated to obtain how many pixels are accommodated in a tile of work domain.

$$\Delta P_x = P_{x2} - P_{x1} \quad (1)$$
$$\Delta P_y = P_{y2} - P_{y1} \quad (2)$$
$$\Delta R_x = R_{x2} - R_{x1} \quad (3)$$

$$\Delta R_y = P_{y2} - P_{y1} \qquad (4)$$

From eq. (1) – (4)

$$(Tile\ Width)_x = \frac{\Delta P_x}{\Delta R_x}$$

$$(Tile\ Width)_y = \frac{\Delta P_y}{\Delta R_y}$$

The equivalent coordinates for the coordinates are given as,

$$R_{x'} = R_{x_1} + \frac{P_{x'}}{(Tile\ Width)_x}$$

$$R_{y'} = R_{y_1} + \frac{P_{y'}}{(Tile\ Width)_y}$$

Where $R_{x'}$ and $R_{y'}$ the equivalent robot coordinates for the pixel coordinates $P_{x'}$ and $P_{y'}$.

6.2 External Communication with Robot Arm

There are plenty of Robot Arms available in the market which are programmable and can be used to do extensive automation and for research purposes. These robots can be programmed in C++, Python or any other ROS. The methodology dealt in this paper can be applied to such Robot Arms which can be controlled and programmed externally by WiFi module, Bluetooth module or USB. Once the external communication is set up, different movements like basic operation of pick and place is programmed using suitable programming language and by selecting an appropriate end effector. Z – Axis of the robot should be kept constant for the entire automation period. The programming and configurations must be done for x and y coordinates only.

VII. Experimental Results

The above expressed method was implemented on a Robot Arm which was externally programmable in Python 3.5. All the necessary packages were installed on the operating computer and the automation task was performed. The Robot Arm was first configured with basic movements for pick and place operations and then the image processing file was executed. The test was performed with 3 cases namely,

Case 1: The work domain consisted of 5 nuts.
Case 2: The work domain consisted of 5 bolts.
Case 3: The work domain consisted of 5 nuts and 5 bolts
Case 4: The work domain consisted of 10 nuts
Case 5: The work domain consisted of 10 bolts

Fig. 7 shows the work setup used for these automation tasks. The work setup consists of a Robot Arm, camera working area, the objects for pick and place and box in which the objects will be sorted. The automation task for the above mentioned work domain was performed and the following results and accuracy were obtained.

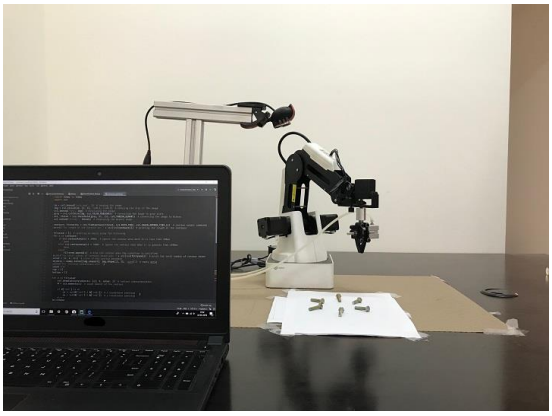

Fig. 7: Work Setup

The image processing task is one of the most important step which decides the accuracy of the automation task. Fig. 8 shows the sample image processing outputs. Once the image processing task is completed, the coordinates are converted and fed to the Robot arm to do the pick and place operation.

Fig. 8: Sample Input of Work Domain

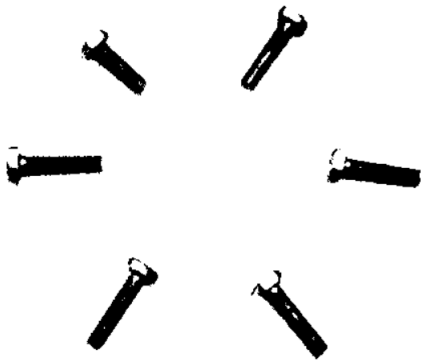

Fig. 9: Sample Binary Image Output of Image Processing of a Work Domain

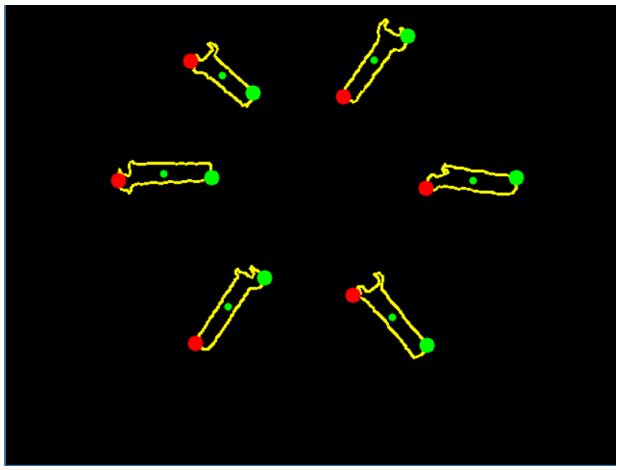

Fig. 10: Sample Object Detection using Image Processing of a Work Domain

Tests for Checking the Accuracy of Automation				
Case No.	No. of Nuts	No. of Bolts	No. of Nuts sorted	No. of Bolts sorted
1	5	-	4	-
2	-	5	-	5
3	5	5	4	5
4	10	-	9	-
5	-	10	-	8

Table. 1: Tests to check the accuracy of Robot

From the Table.1 it can be evidently concluded that the automation task accuracy to pick and place lies between 70% to 99%. This accuracy speaks for how many objects of interest were detected using the image processing and program and how many objects of interest were successfully picked up and placed.

The graph showed in Fig. 11 shows a visual representation of the overall results and accuracy.

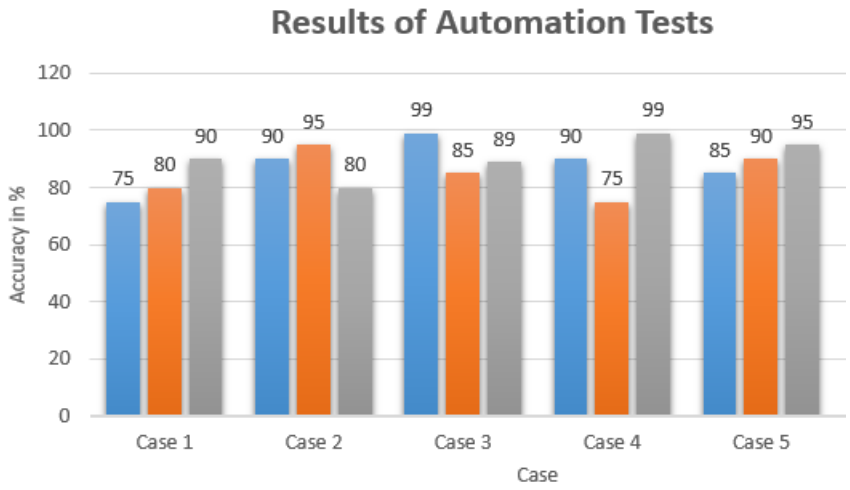

Fig. 11: Results of Automation Tests

VIII. Conclusion

This paper expresses the importance of object detection using open source software and pairing it with a Robot Arm to achieve the basic automation operations by giving more importance to cost efficient Machine Vision systems. Object Detection has become one of the most emerging research areas in the field of Computer Vision for making existing systems more efficient. Using cost efficient machine vision Industrial Robots for automation which will help in encouraging small scale industry owners to use the cost effective robots instead of proprietary robots which are very expensive. This will boost up the overall development and rate of production in various industries where automation is an important aspect.

IX. Acknowledgement

This research paper would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to the professors from Department of Electrical and Electronics Engineering of B.M.S Institute of Technology & Management for the guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project. We would like to express our gratitude towards Ekzen Robotics, Bangalore for their kind co-operation and encouragement which helped us in completion of this paper.

X. References

[1] S. Kucuk and Bingul, *"Robot Kinematics: Forward and Inverse Kinematics" in Industrial Robotics: Theory Modelling*

and Control, pp. 128-148, 2007, ISBN ISBN 3-86611-285-8.

[2] Fu, K.S (King Sun) Robotics: Controlling, sensing, Vision and Intelligence international edition 1987

[3] Ondrej Hock and Jozef Sedo: *"Inverse kinematics using transportation method for robotic arm"*, ELEKTRO, 2018

[4] Jing Huang , Xianlun Wang , Dongsheng Liu and Yuxia Cui , *A New Method for Solving Inverse Kinematics of an Industrial Robot,* 2012 International Conference on Computer Science and Electronics Engineering Year: 2012 , Volume: 3

[5] Sandeep Kumar, Aman Balyan and Manvi Chawla, *Object Detection and Recognition in Images,* Volume 5, Issue 4, ISSN: 2321-9939 IJEDR1704166 International, 2017

[6] Shashank Prasad; Shubhra Sinha; *"Real-time Object Detection and Tracking in an Unknown Environment",* World Congress on Information and Communication Technologies, 2011

[7] Perez and B. Granger, *IPython: a system for interactive scientific computing*, Computing in Science & Engineering, vol. 9, no. 3, pp. 21–29, 2007.

[8] Bradski, *The OpenCV library*, DOCTOR DOBBS JOURNAL, vol. 25, no. 11, pp. 120–126, 2000.

[9] P.Ramachandran, *Performance of various Python implementations for solving the 2D Laplace equation., 1991*

[10] Brian Thorne and Raphaël Grasset, Richard Green, *Using Python in Computer Vision: Performance and Usability*, HIT Lab NZ, 11 November 2014.

[11] Far´es Jalled and Ilia Voronkov, *Object Detection Using Image Processing*, Moscow Institute of Physics & Technology, Department of Radio Engineering & Cybernetics, 23 Nov 2016.

[12] Joseph Schlecht and Björn Ommer, *Contour-based Object Detection",* Interdisciplinary Center for Scientific Computing, University of Heidelberg, Germany, 2011.