Shape detection and classification using OpenCV and Arduino Uno

Xhensila Poda
Faculty of Natural Sciences
University of Tirana
xhensila.poda@fshnstudent.info

Olti Qirici
Faculty of Natural Sciences
University of Tirana
olti.qirici@fshn.edu.al

Abstract

“ *If We Want Machines to Think, We Need to Teach Them to See. --Fei Fei Li, Director of Stanford AI Lab and Stanford Vision Lab* ” [ERD12]

Working on Computer Vision is equivalent to working on millions of calculations in the blink of an eye with almost same accuracy as that of a human eye. It is not just about converting a picture into pixels, and then try to make sense of what’s in the picture through those pixels but it is thinking of what can be done by a machine when they will be able to see as accurate as a human eye.

1. Introduction

There is a clear distinction between two regular geometric shapes, for example a circle and a square. Identifying this difference for a human is easy, but how can a computer find the difference between these two objects? How can we build a system that takes an image as an input, detects and distinguishes the shapes present in it ?

This paper will introduce a system for shape detection recognition and classification. To make the system complete, a great importance is given to the necessary image processing techniques to be applied. We’ll make an effort to go all the way, ranging from the lower image processing layers to the stage in which a hardware system (based on *Arduino*) could classify objects in shapes of certain categories, through a mechanical arm. The main motivation for the research lies with discovering what the limits and possibilities are. But apart from the theoretical point of view there also is a great practical use when it comes to object detection. The motivation for wanting to build a system that can detect different shapes mainly stems from a desire to make the computer able to interact with the real world.

Computer Vision applications, which have been the subject of constant research interest for almost four decades, have now matured to a point where they can be applied efficiently for advanced tasks in various fields of human activity. One of the areas that has shown great interest in such systems is the Industry. In the process of assembling and quality control in industry, there is a strong need for *robot-based object detection and recognition systems*. The application of these systems in industry gave way to the realization of this work.

The Industrial Assembly process is part of the manufacturing process and can be defined as a set of tasks in which all parts of the discrete components are gathered to form a certain configuration. Fitting is often the weakest point in the entire production process because this activity takes a considerable part of the cost and total production time. One of the main reasons for this is the diversity of products that a single venture places on the market, so it is often difficult for companies to follow market demands when their assembly is mainly based on manual assembly processes. This is even more true as the market requires products that meet high expectations in the areas of quality, price and delivery time. The key to achieving this goal is the continuous improvement of the production process.

Today, industrial robots can perform tasks with precision and high speed. However, compared to human operators, robots are hampered by lack of sensory perception to achieve more sophisticated tasks. People make object detection and recognition seem like a trivial process. We can easily identify objects in the surrounding environment, regardless of their circumstances, whether they are upside down, different from color, other than composition, etc. And the objects that appear in many different shapes or objects that have significant deviations in the form such as trees, (rarely find two trees that have identical shape) can easily be generalized by our brain as the same kind of object. To go from the evidence of objects to the human world in the computerized detection of objects is necessary that we add to this machines the sense of sight. This concept, in the automated world, is known as Artificial Vision. But what is Artificial Vision and to what level does this system look like with the sense of sight in the human world? The technical definition of Artificial Vision is as follows:

"*The use of devices for optical, non-contact sensing to automatically receive and interpret an image of a real scene in order to obtain information and/or control machines or processes.*" [WHE97]

Numerous processing can help to extract some information about the forms and objects present in an image, but the level of abstraction achieved by people can't be compared to the Artificial Vision techniques nowadays. This gap between these two worlds has several reasons that are explained in the following paragraph.

2. Difficulties

2.1 Lighting

A perennial problem in working with images is the various lighting conditions a picture can have been taken under. The effect of lighting on colors is extremely large, and especially in color based recognition is considered to be one of the most difficult problems. Human eyes within a broad range automatically adjust for brightness. A computer does not have that capability. This greatly reduces its ability to recognize colors. Good examples are the very popular soccer robots, the Aibos. In most cases they are programmed to find their location in the field by identifying colors of objects around them. Every new match to be played by these robots is preceded by a large operation of recalibrating them to new lighting conditions. This takes a lot of precious time (it can easily take an hour) and can be troublesome because the time constraints often compromise the quality. Since we work with shapes instead of colors we are less dependent on these changing colors, but unfortunately this does not mean that lighting is not a problem for us. Mechanisms like edge extraction are sensitive to lighting conditions. Edges tend to disappear when lighting becomes dim and the difference in brightness of pixels tends to decrease. And brighter images can on their part lead to the extraction of too much edge information, as with a brighter image small color changes can become more visible. [VRI06]

2.2 Multiple objects

Just about the hardest problem of object detection in real images is that the objects are not likely to appear alone, making it very hard to detect them, since separation of objects in an image is not a trivial task. This is even harder when two objects are touching or overlapping, possibly letting them be identified as one. Such a problem, of separation of objects and clustering of various elements, by itself justifies a full research project and we have no other choice then to give this issue here a lower priority and refer the research to future work. [VRI06]

3. Tools

3.1 OpenCV

A multitude of software tools and libraries manipulating images and videos are available, but for anyone who wishes to develop smart vision-based applications, the OpenCV library is the tool to use. Since its introduction in 1999, it has been largely adopted as the primary development tool by the community of researchers and developers in computer vision. Below are listed the main reasons why i have chosen OpenCv to develop my project:

3.1.1 Cross-Platform Library

OpenCV is a cross-platform library, enabling users on different operating systems to use it without complications. It is even accessible on mobile systems like iOS and Android, making it a truly portable library.

3.1.2 Vast Algorithms

OpenCV (Open source Computer Vision) is an open source library containing more than 500 optimized algorithms for image and video analysis. This vast library allows programmers to perform a multitude of tasks in their software such as extracting 3D models of objects, object detection and tracking etc.

3.1.3 Extensive Use

OpenCV is being used by giant companies like Google, IBM, Toyota and startups such as Applied Minds and Zeitera as well as organizations in countries all over the world to conduct multifarious tasks. This gives users assurance that they are on board of a library that is being used extensively by enterprises and government institutions. Moreover, OpenCV has a vast community where users can ask for assistance and offer help to their fellow developers in case they have questions regarding codes or the platform. This lets developers access insights from real people about the library and its codes.

3.1.4 Efficient

OpenCV was designed for computational efficiency and with a strong focus on real-time applications. Written in optimized C/C++, the library can take advantage of multi-core processing. Enabled with OpenCL, it can take advantage of the hardware acceleration of the underlying heterogeneous compute platform. [LAG11]

3.2 Arduino

Arduino is a for-profit company that builds the self-entitled Arduino open-source hardware platform. What this means is that the hardware designs are open, as is all software used to program the boards. Third parties are welcome to build and distribute their own, but the name 'Arduino' cannot be used by third parties. A rather large enthusiast community has sprung up around the Arduino board, making it a popular platform for personal projects. The Arduino boards incorporate an ATMega microcontroller, along with necessary electronics to connect it to a computer via USB for programming, and to help regulate its power and sanitize its inputs and outputs to a degree. There are several types of Arduino boards, and the Arduino Uno with the ATMega328 microcontroller was the board chosen for this project. Arduino provides an open-source software development environment for programming the Arduino boards. The boards are programmed using a modified version of C++, in which some concepts esoteric to novice programmers are hidden. Straight forward functions are included for most common microcontroller tasks. The programming environment makes it very easy to write programs, and loading them onto the board is as easy as plugging in the USB cable and clicking the "upload" button.

3.2.1 Reason behind choosing Arduino

First, the device is an open-source hardware platform that is programmed using an open-source programming environment. Moreover, the Arduino boards are advertised to be straight forward to use and program; they are designed for use by artists and hobbyists. The C++ like programming language used to program the board is easy to grasp. The last advantage has to do with the cost. Arduino platforms have a low cost and compared to the variety of projects that gives the opportunity to develop this cost is insignificant.

4. Methodology used

4.1 Contours identification and utilized techniques

This project was developed in three main directions. First direction focuses on image processing techniques in order to detect and classify the regular shapes present in an image. The second direction focuses on how the output produced in the first stage will be transmitted in Arduino and how Arduino will interact with this output. The third direction is the construction of a hardware platform in which all the results obtained will be combined and will solidify the purpose of this research.

"*A contour is a closed curve of points or line segments, representing the boundaries of an object in an image.* " [RIC03]

In other words, contours represent the shapes of objects found in an image. If internal detail is visible in an image, the object may produce several associated contours, which are returned in a hierarchical data structure. Once the contours of an object are detected, we can do things like determine the number of objects in an image, classify the shapes of the objects, measure the size of the objects etc. The input to the contour-finding process is a binary image, which it will produced by first applying thresholding and/ or edge detection techniques. In a binary image, the objects to be detected should be white, while the background of the image should be black. A point worth mentioning is that it is not enough to merely identify the boundary pixels of a pattern in order to extract its contour. What we need is an ordered sequence of the boundary pixels from which we can extract the general shape of the pattern. [ALE17]

Contour tracing is one of many preprocessing techniques performed on digital images in order to extract information about their general shape. Once the contour of a given pattern is extracted, it's different characteristics will be examined and used as features which will later on be used in pattern classification. Therefore, correct extraction of the contour will produce more accurate features which will increase the chances of correctly classifying a given pattern.

Meanwhile, these questions may arise: Why waste precious computational time on first extracting the contour of a pattern and then collecting its features? Why not collect features directly from the pattern?

The appropriate answers to such questions would be: the contour pixels are generally a small subset of the total number of pixels representing a pattern. Therefore, the amount of computation is greatly reduced when we run feature extracting algorithms on the contour instead of on the whole pattern. Since the contour shares a lot of features with the original pattern, the feature extraction process becomes much more efficient when performed on the contour rather on the original pattern.

In conclusion, contour tracing is often a major contributor to the efficiency of the feature extraction process - an essential process in the field of pattern recognition. [JAN12]

As mentioned earlier before applying a contour detection algorithm we need to apply *Thresholding* or *Edge Detection*. But which of the techniques should we choose? Before we make a comparison of two techniques, let's look at the technical definition for Thresholding.

"*Thresholding is a nonlinear operation that converts a grayscale image into a binary image where pixels are divided into two groups depending on a value defined as the threshold value. The pixels that have a value greater than the threshold value receive the value of 1 while others take the value 0.* " [SHA01]

Thresholding just takes a look at intensities and sees whether or not each value is smaller or larger and we get "edge" points respectively. However, depending on the complexity of the scene, thresholding and edge detection would yield the same thing. For example, if you had a clean image that has a clear intensity difference between the foreground and background, then either edge detection or thresholding would work. The scene that we have to deal with is very complex, with multiple objects in it, so thresholding will not give good results. This is why edge detection is chosen.

Canny Edge Detection is a popular edge detection algorithm. It was developed by John F. Canny in 1986. OpenCv library has a built in method named Canny() which implements Canny Edge Algorithm. [ALE17]

The implementation of this algorithm in OpenCv goes through several phases that are explained below:

4.1.1 Noise Reduction

Since edge detection is susceptible to noise in the image, first step is to remove the noise in the image with a 5x5 Gaussian filter.

4.1.2 Finding Intensity Gradient of the Image

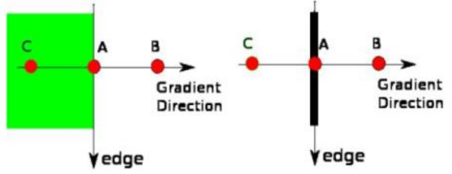
Smoothened image is then filtered with a Sobel kernel in both horizontal and vertical direction to get first derivative in horizontal direction (Gx) and vertical direction (Gy). From these two images, we can find edge gradient and direction for each pixel as follows:

Gradient direction is always perpendicular to edges. It is rounded to one of four angles representing vertical, horizontal and two diagonal directions.

4.1.3 Non-maximum suppression

After getting gradient magnitude and direction, a full scan of image is done to remove any unwanted pixels which may not constitute the edge. For this, at every pixel, pixel is checked if it is a local maximum in its neighborhood in the direction of gradient Point A is on the edge (in vertical direction). Gradient direction is normal to the edge. Point B and C are in gradient directions. So point A is checked with point B and C to see if it forms a local maximum. If so, it is considered for next stage, otherwise, it is suppressed (put to zero). In short, the result you get is a binary image with "thin edges".

Figure 1

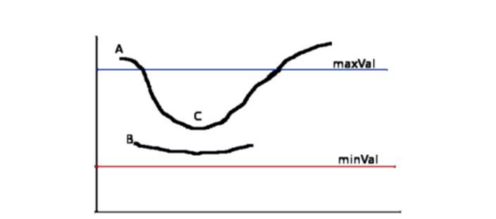


Non-Maximum Suppression

4.1.4 Hysteresis thresholding

This stage determines which edges are real edges and which are not. For this two threshold values are needed, $minVal$ and $maxVal$. Any edges with intensity gradient more than $maxVal$ are sure to be edges and those below $minVal$ are sure to be non-edges. Those who lie between these two thresholds are classified edges or non-edges based on their connectivity. If they are connected to "sure-edge" pixels, they are considered to be part of edges. Otherwise, they are discarded. For example edge A in Figure 2 is above the *maxVal*, so is considered as “sure-edge”. Although edge C is below *maxVal*, it is connected to edge A, so that also considered as valid edge and we get that full curve. But edge B, although it is above *minVal* and is in same region as that of edge C, it is not connected to any “sure edge”, so that is discarded.It is very important that *minVal* and *maxVal* are accurately selected to get the correct result. This stage also removes small pixels noises on the assumption that edges are long lines. [HOW15]

Figure 2



Hysteresis thresholding

4.1.5 Contour detection algorithm in OpenCv

Contour detection algorithms can typically be categorized into three types as follows

- Pixel following,
- Vertex following
- Run-data-based following

Of these, the pixel-following method is the most common. Pixel-following method traces contour pixels in a predefined manner and then saves their coordinates in memory according to the trace order. Pixel-following methods, such as the simple boundary follower (SBF) , modified SBF (MSBF), improved SBF (ISBF) Moore-neighbor tracing (MNT),the radial sweep algorithm (RSA) and the Theo Pavlidis algorithm (TPA) have simple rules for tracing contour pixels based on a chain code. These methods require a frame-size memory to trace the contour and generate erroneous images when the contour image is enlarged because they maintain only the pixel coordinates. [TAU]

OpenCv has a built in method that implements an algorithm for contour detection: cv2.findContours(). This method takes as input three parameters:first one is source image, second is contour retrieval mode and third is contour approximation method. The outputs are: image, contours and hierarchy. Contours is a Python list of all the contours in the image. Each individual contour is a NumPy array of (x,y) coordinates of boundary points of the object.

Figure 3

```
#contours
canny2, contours, hierarchy =
cv2.findContours(canny,cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
        for i in range(0,len(contours)):
            peri = cv2.arcLength(contours[i],True)
            approx = cv2.approxPolyDP(contours[i],peri*0.02,True,approxOut
```

Contour detection code

After detecting all contours in an image, in order to perform shape detection contour approximation will be used.

As the name suggests, contour approximation is an algorithm for reducing the number of points in a curve with a reduced set of points — thus the term approximation. This algorithm is commonly known as the *Ramer-Douglas-Peucker* algorithm, or simply the *split-and-merge algorithm.* Contour approximation is predicated on the assumption that a curve can be approximated by a series of short line segments. This leads to a resulting approximated curve that consists of a subset of points that were defined by the original curve. Contour approximation is actually already implemented in OpenCV via the *cv2.approxPolyDP()* method. [JAN12]

In order to perform contour approximation, firstly is needed to compute the perimeter of the contour followed by constructing the actual contour approximation as shown below:

```
peri = cv2.arcLength(c, True)
approx=cv2.approxPolyDP(contours[i],peri*0.02
,True)
```

Common values for the second parameter to cv2.approxPolyDP are normally in the range of 1-5% of the original contour perimeter. Given our approximated contour, we can move on to performing shape detection:

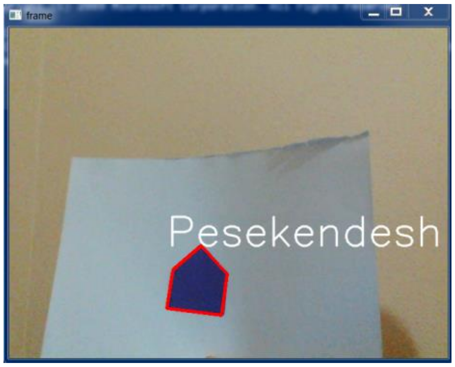
It’s important to understand that a contour consists of a list of vertices. We can check the number of entries in this list to determine the shape of an object.

For example, if the approximated contour has five vertices, then it must be a pentagon as shown below:

```
if(len(approx) == 5):shape = 'p'
arduino.write(shape.encode())
x,y,w,h = cv2.boundingRect(contours[i])
cv2.drawContours(frame, [approx], -1, (0, 0,
255), 3)
```

The output of this phase is shown below:(Figure 4)

Figure 4



Contour detection result

4.2 Serial communication between Arduino and OpenCV

4.2.1 Data input in Arduino

This part will demonstrate how to establish a connection between Arduino and Python script. The transferred information is just an indicator for the type of shape that has been detected. This can be done using Serial Communication.

But why does Arduino need this information? Because Arduino is the “brain” that will put the hardware device in move. Depending from the input that it takes from the python script ,Arduino will control the movement of the mechanical arm. Python and Arduino both contain a library that is specific for serial communication, respectively *PySerial* for python and *Serial* for Arduino. The sender of the information in this case is Python script while the receiver is Arduino. On the side of the sender we declare an object of the *PySerial* Library which will initiate the communication path from which the data will be transmitted . This object will save an important information , that is the port in which Arduino will receive the message. Another important element is information that will be sent. Serial communication transmit data bit after bit. When a specific shape is detected (i.e pentagon) in a variable we will save a character (i.e 'p') .This means that the detected shape is a pentagon and then this information is send to Arduino by *arduino.write*() command.

4.2.2 Data processing in Arduino

Now that the communication between these technologies has been made possible , is only left to see how this received information can be used by Arduino.

The code structure of Arduino is made of two main methods: *init()* in which we declare all the necessary objects and variables and *loop()* which will be executed as long as Arduino is in working conditions.

In the body of init() method *Serial.begin()* is called, which will initiate communication for Arduino.

Loop method manages the received information as shown below:

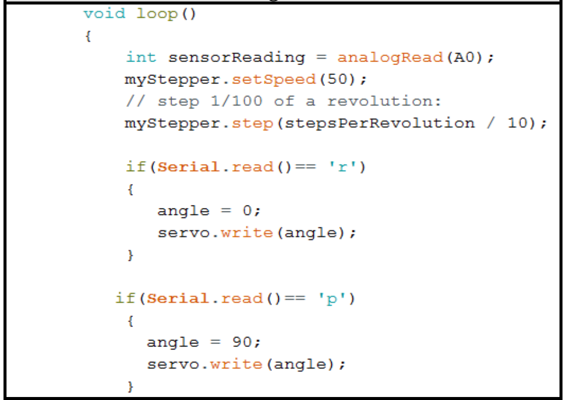
:if(Serial.read()== 'p')

So if message contains character 'p' that means python script has detected a pentagon. In this case Arduino will give the command to move servo motor ninety degrees. This command will open the first bin which is responsible for holding pentagons.

This control is possible only for two kind of shapes (triangle or rectangle,circle or triangle,etc) because hardware structure has only two containers .

Arduino code responsible for controlling the mechanic arm is shown below:

Figure 5



```
void loop()
{
    int sensorReading = analogRead(A0);
    myStepper.setSpeed(50);
    // step 1/100 of a revolution:
    myStepper.step(stepsPerRevolution / 10);

    if(Serial.read()== 'r')
    {
       angle = 0;
       servo.write(angle);
    }

   if(Serial.read()== 'p')
    {
      angle = 90;
      servo.write(angle);
    }
```

Servo control using Arduino

5. The hardware

"People who are really serious about software should make their own hardware"-- *Alan Kay*

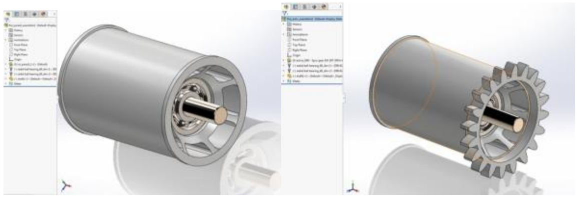
Although this is not entirely our case, it is a great inspiration to build the optimal hardware platform that best suits the software we have built up. The built-in hardware platform is made up of several parts that are explained in the following paragraphs. Most of the components of this platform are realized through 3D printing.

3D printing has been a popular method of creating prototypes since 1980 and is quickly becoming the fastest and most affordable way to create custom consumer goods. There are several different 3D printing methods, but most widely used is a process known as Fused Deposition Modeling (FDM). FDM printers use a thin thermoplastic filament, which is heated to its melting point and then layered to form a three-dimensional object.

3D printings produce very accurate, highly stable, highly efficient and very low cost structures. These qualities make these structures very suitable for building functional prototypes.

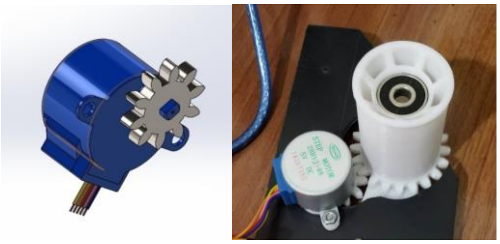
The most important part of this platform is the conveyor. (Figure 6) His role is to shift geometric shapes up to the mechanical arm. The conveyor is built by two rollers, one of which is passive and the other is active. The active one is moved by servo motor and is responsible for moving the entire conveyor.(Figure 7)

Figure 6



Conveyor design

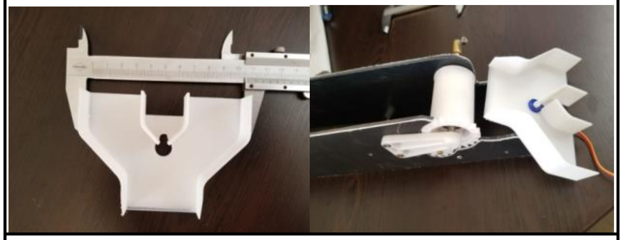
Figure 7



Parts of the conveyor

The second part is the classification structure. This part is responsible for the process of classifying shapes. The whole structure is static except the mechanical arm positioned in its center. This structure is supported by stepped booster which creates the slope necessary for the smooth passage of geometric shapes during the classification process. (Figure 8)

Figure 8



With the description of these parts we end up the explanation of hardware construction for this application.

The finalized project looks like this:

Figure 9

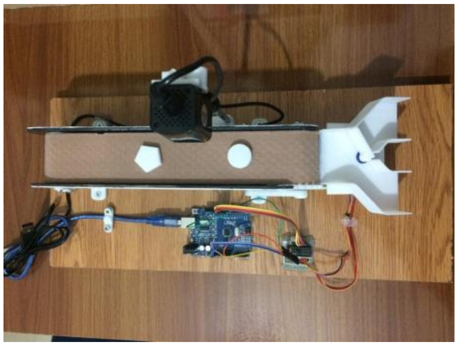
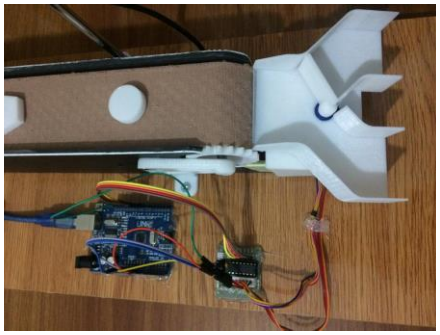


Figure 10



5. Conclusions

Enabling a machine to “see” and “understand ” gives you the ability as a developer to see and understand a lot, because this is a job that turns all your efforts into benefits. The world of computer vision and robotics in general is quite an intriguing world. The benefits that a software developer gets when he studies this field are countless. Thanks to Open Source technologies, prototypes can be realized at a very low cost. Today a modest system was built,tomorrow with the same software and with a more suitable real-world hardware we can build a system that is part of the manufacturing industry in Albania. Computer Vision is not just the implementation of an algorithm, it is not just mathematical calculation on a vector. Computer Vision is thinking about the power that can be given to each system when we give it a chance to perceive the surrounding environment.

References

[WHE97] P. Whelan, B. Batchelor. *Intelligent Vision Systems for Industry*, 1997

[VRI06] J. de Vries. *Object Recognition: A Shape-Based Approach using Artificial Neural Networks*, January 2006

[SHA01] L. Shapiro, G. Stockman. *Computer Vision (1st Edition)*, February 2001

[HOW15] J. Howse. *Learning OpenCv with Python (2nd Edition)*, September 2015

[TAU] G. Toussaint, *Course Notes: Grids, connectivity and contour tracing (PostScript)*

[PAV82] T. Pavlidis, *Algorithms for Graphics and Image Processing*, Computer Science Press, Rockville, Maryland, 1982

[MIK] Mike Alder, *Border Tracing (by radial sweep)*

[SOS] M. Soss, *Proof of correctness of Square Tracing algorithm when both pattern and background are 4-connected*

[LAG11] Robert Laganiere *OpenCV 3 Computer Vision Application Programming Cookbook - Third Edition* 2011

[JAN12] Jan Erik Solem *Programming Computer Vision with Python* 2012

[ALE17] Alexander Mordvintsev & Abid K *OpenCV-Python Tutorials Documentation Release 1* 2017

[ALE15] Aleš Ude *Robot Vision* 2015

[JUR16] Jürgen Beyerer, Fernando Puente,León Christian Frese, *Machine Vision Automated Visual Inspection: Theory, Practice and Applications* 2016

[MUB97] Mubarak Shah *Fundamentals of Computer Vision* 1997

[RIC03] Richard Szeliski *Computer Vision: Algorithms and Applications* 2003

[ERD12] E. R. DAVIES *Computer and Machine Vision: Theory, Algorithms,Practicalities* 2012