Journal of Engineering Sciences

Vol 11, Issue 7,July/2020

ISSN NO:0377-9254

LIQUID DETECTION

Krishnangshuk Paul[1] and * Pranam Paul[2]

[11]*B.Tech Student, CSE (Cyber Security), The Neotia University, India,*

pasujoy141@gmail.com

[2]*Assistant Professor, CSE department, The Neotia University, India,*
pranam.paul@gmail.com

***Abstract*— Nowadays object identification through cameras is a common thing. There are applications like Google lens which help you gain knowledge or identify any specific object and helps you know the things around you better. In this paper, we will be discussing how spilled liquids can be detected with the help of the camera without using Machine Learning. Detection and identification of any specific object is done by machine learning which of course is very reliable. But our objective will be to detect spilled liquids without the help of any machine learning algorithms. As you will see at the end that we have successfully been able to detect spilled liquid in real-time with the camera and all the steps involved are discussed below.**

***Keywords*— Liquid detection, Spilled liquid detection, Reflection detection, Liquid detection without machine learning.**

1. INTRODUCTION

In this paper, we are going to detect spilled liquids with the help of the reflection property of liquids. Every liquid reflects light falling on its surface, so the program will detect the presence of a reflection in a frame and check whether the reflection is within a bounding contour similar to that of spilled liquid. Checking whether the reflection is inside any contour will be done with the help of contour markings in extreme points. After checking the extreme points the area of the bounding contour to that of the reflection will be checked to determine whether there is any spilled liquid. There will be some limitations to this method of liquid detection which will be listed below. But with this method, we will be able to detect liquid without using any machine learning algorithms. The program can detect spilled liquid through the camera itself making it very useful in today's world. This can be used for detecting spilled liquid in various places automatically which will in turn be a great step towards things like water preservation.

2. LITERATURE REVIEW

All projects about the detection of spilled liquids are not very accurate using machine learning and deep learning. It is because of the ambiguous nature and the physical properties of liquids which makes its detection even harder. A liquid can take the shape of any container it is kept in and some liquids like water, oil, vinegar, etc. are transparent in nature which makes it even more difficult to identify. So the motive was to make a program that will be able to detect spilled liquids without the help of any machine learning or deep learning algorithms. A most unique property of liquid is the reflection of light which can be used to detect spilled liquid. The reflection property of liquids is used in this program for its identification, i.e., the whole program is based on the presence of reflection on the liquid surface. Help from many websites and papers were taken to accomplish this which are all listed in the references section. A major necessity of the program was to be able to use the mobile camera to detect the liquid by linking it to the program, this was accomplished using ipwebcam [14]. Finding the extreme points were necessary for the program to check whether the reflection was within any contour [16, 17]. For the detection of the reflection, the HSV color frame was used [11, 13]. Each frame captured was passed through the Gaussian blur filter to make it more smooth [1, 3]. Canny Edge detection method was used to detect the edges of the reflection and liquid [2, 6, 18]. The Inrange function was used for the detection of the reflection along with the HSV color frame[4]. Contours were drawn on every edge detected for further analysis [5, 7, 8, 9]. Help and ideas were taken from various papers as well [10, 12, 15].

3. ACTUAL WORK

The basic idea used in this program is to detect the reflection of light. With the help of the detected reflection and the contours of the image, the

program tries to determine and mark the possible region where the liquid is spilled.

In 3.1 we explain and discuss HSV Format Color Space. In 3.2 we explain the format of Gaussian Blur. In 3.3 we explain the working principle of Canny Edge Detection. In 3.4 we explain the format of Contours. In 3.5 we describe the procedure of the program step by step in a detailed explanation of how it is executed. The various features of OpenCV used in this program are:

3.1. HSV FORMAT COLOR SPACE

The HSV color front is used in reflection detection in the program. An **HSV** is another type of color space in which **H stands for Hue**, **S stands for Saturation** and **V stands for Value** [11, 13].

(i) **HUE**: Table I. shows the relation between the angle from 0° to 360° and the colors for HSV color frames.

Table I. Angle to Color correspondence of HSV

Angle	Color
0-60	Red
60-120	Yellow
120-180	Green
180-240	Cyan
240-300	Blue
300-360	Magenta

(ii) SATURATION: It indicates the range of grey in the color space. It ranges from 0 to 100%. Sometimes the value is calculated from 0 to 1. When the value is **'0,'** the color is grey and when the value is **'1,'** the color is a primary color.

(iii) Value: It is the brightness of the color and varies with color saturation. It ranges from 0 to 100%. When the value is **'0'** the color space will be totally black. With the increase in the value, the color space brightness up and shows various colors.

3.2. GAUSSIAN BLUR

In Gaussian Blur operation, the image is convolved with a Gaussian filter instead of a box filter. The Gaussian filter is a low-pass filter that removes the high-frequency components. The Gaussian blur is used for filtering the frames and reducing noise for proper judgment of the frames [1],[3].

3.3. CANNY EDGE DETECTION

It is a popular edge detection algorithm. It was developed by John F. Canny in 1986. It is used for identifying all the edges of the image for further processing [2],[6] . It is a multi-stage algorithm and here each stage is explained:

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

3.3.2. FINDING INTENSITY GRADIENT OF THE IMAGE: Smoothened image is then filtered with a Sobel kernel in both horizontal and vertical directions to get the first derivative in the horizontal direction (G_x) and vertical direction (G_y). From these two images, we can find edge gradient and direction for each pixel as follows [18] :

$$Edge\_Gradient\ (G) = \sqrt{G_x^2 + G_y^2}$$

$$Angle\ (\theta) = \tan^{-1}\left(\frac{G_y}{G_x}\right) \quad (1)$$

In equation (1) the edge gradient is found by the square root of the sum of the squares of Gradient x (G_x) and Gradient y (G_y) respectively. The Angle is calculated by the tan inverse of the ratio of the G_y to the G_x.

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

3.3.3. NON-MAXIMUM SUPPRESSION: After getting gradient magnitude and direction, a full scan of the image is done to remove any unwanted pixels which may not constitute the edge. For this, at every pixel, a pixel is checked if it is a local maximum in its neighborhood in the direction of the gradient. Check the image below.

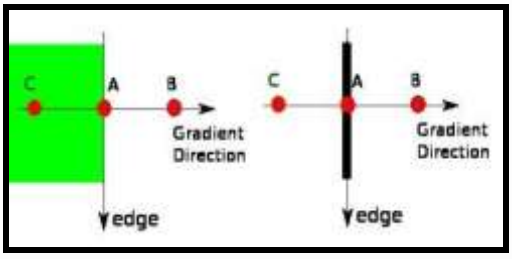

Fig 3.3.3.1 Non-maximum Suppression to find the actual edges of the image

In Fig 3.3.3.1 Point A is on the edge (in the vertical direction). Gradient direction is normal to the edge [18]. 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 the next stage, otherwise, it is suppressed (put to zero).

3.3.4.HYSTERESIS THRESHOLDING: This stage decides which edges are really edges and which are not. For this, we need two threshold values, 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, so discarded. 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 also discarded.

3.4. CONTOURS

Contours can be explained simply as a curve joining all the continuous points (along the boundary), having the same color or intensity. It is used for making all the edges of the frame for further processing. The contours are a useful tool for shape analysis and object detection and recognition [5],[7],[8],[9] .

- For better accuracy, use binary images. So before finding contours, apply threshold or canny edge detection.
- findContours function modifies the source image. So if you want a source image even after finding contours, already store it to some other variables.
- In OpenCV, finding contours is like finding white objects from a black background. So remember, objects to be found should be white and backgrounds should be black.

To draw the contours. It can also be used to draw any shape provided you have its boundary points. Its first argument is the source image, the second argument is the contours which should be passed as a Python list, the third argument is index of contours (useful when drawing individual contour. To draw all contours, pass -1) and remaining arguments are color, thickness, etc.

3.5. SYSTEM ARCHITECTURE

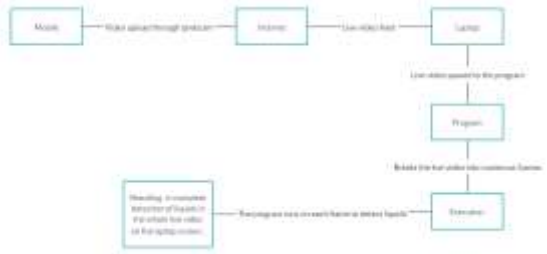

Fig 3.5.1 Block Diagram of the architecture used in the program

The system configuration of the laptop used to run this program are HP laptop, Intel(R) Core(TM) i36006U @ 2.00Ghz, 2000Mhz, 2 Core(s), 4 Logical Processor(s), 8Gb RAM(x64).

In Fig 3.5.1 the whole architecture used in the detection of the liquid is explained through a block diagram. Using ipwebcam the live video feed is uploaded to the internet and sent to the laptop. The laptop uses the live video feed in the program where it is divided into numerous frames and the program runs on each frame to detect liquid. Each frame is shown on the laptop screen after the program is finished with it, thus playing the live feed with the detected liquids marked on the video by the program.

3.6. STEPWISE EXECUTION OF THE PROTOCOLS

In section 3.6.1, the setup of the ipwebcam for this purpose is being discussed whereas section 3.6.2 guides us to the way of conversion of an image in various formats. To detect the contours and edges section 3.6.3 is used. Similarly, 3.6.4 and 3.6.5 are taking care of the status of the reflection and edge contours and drawing of liquid contours and labeling respectively.

3.6.1. SETTING UP THE IP WEBCAM:

1. Import the necessary libraries which in this case are cv2, urlib.request and NumPy [14].
2. Set up the IP webcam URL and capture the video, so that mobile phone cameras can be used instead of the machine webcam. Using the machine webcam to capture spilled liquid will be unnecessarily uncomfortable.

3. Run an infinite loop and take each captured frame for analysis.

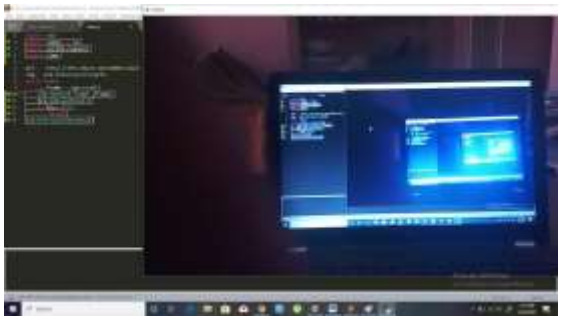

Fig 3.6.1.1. Setting up IP Webcam which is successfully passing the live video clip to the laptop

In Fig 3.6.1.1 a picture of the working ipwebcam is clicked by the connected phone is shown.

3.6.2. CONVERSION OF AN IMAGE TO VARIOUS FORMATS:

1. Convert the image into Hue, Saturation, and Value (**HSV**) [11],[13] format to help me detect the reflection by identifying the bright white in the image using cv2.inRange function [4].
2. On the other hand, convert the frame to **RGB** format and pass it through a Gaussian blur filter and then convert it into a gray image [1],[3] .

3.6.3. DETECTION OF CONTOURS AND EDGES:

1. Converting the image to blur and gray helps in detecting the edges faster and here we use the canny edge detector to detect the edges of each frame [2],[6] .
2. After finding the edges and the maximum reflection, find the contours of the reflection detected earlier in the program [5],[7],[8],[9].
3. If the program doesn't get any reflection then it is assumed that there's no liquid as the whole program is based on the reflection of light on the liquid.
4. If there is a reflection then after finding the contours of the reflection then find the contours of the edges detected earlier with the same findContours function.

3.6.4. STATUS OF THE REFLECTION AND EDGE CONTOUR :

5. Then identify the contour with the greatest perimeter and with an area that is close but more than the area of the reflection with normal if-else statements.
6. Calculate the extreme points of the reflection contour and the contour identified earlier and match them to see whether the reflection is inside the edge contour by using three functions – one for minimum edge detection, one for maximum edge detection and one for comparison [16],[17].
7. Then check if all the extreme points of the reflection are inside the extreme points of the edges with a simple if and else statement.

3.6.5. DRAWING THE LIQUID CONTOUR AND LABELING IT:

8. If the reflection is found to be within the specific edge contour identified earlier then draw the edge contour and put the text as a liquid above it with the help of the upper extreme point calculated earlier.
9. Lastly, it shows the frames every time and configures a waitKey() to close if the **Esc** button is pressed followed by destroying all windows .

AN EXAMPLE OF OUTPUT: Following the above-mentioned processes from step 1 to step 14 Fig 3.6.1 is detecting the liquid.

Fig 3.6.1. Initiation of the program showing how it detects liquids

4. RESULTS AND ANALYSIS

This program of liquid detection is completely dependent on reflection and edge detection. If the edges are not detected properly then the liquid will not be detected and if there is no reflection upon the liquid even then the liquid will not be detected. For clear edge detection of liquid, a substantial amount of light is needed at particular angles of the

liquid so that there can be a minor shadow to detect the edges properly. Given, the reflection is present and the light is favorable for the edges of the liquid to be detected then the liquid will be detected properly.

In 4.1 the Table of experiments with the various experiments done and their results is shown. In 4.2 Some Examples of Successful detection of liquid are shown and explained. In 4.3 Examples of errors in the detection of liquids are shown and explanations of the reasons for the errors are given. In 4.4 Examples of no detection occurring in liquids are shown and explanations of the reasons for no detection are given. In 4.5 Examples of wrong detections are shown and explanations of the reasons for the wrong detection are given. In 4.6 the limitations of the program have been discussed.

4.1. TABLE OF EXPERIMENTS WITH RESULTS AND THEIR ANALYSIS

Table II shows the table of experiments done on different liquids.

Table II. Experimental Table for liquid detection

Experiment No.	Liquid Nature	Identification Result	Reasons for Errors
Experiment No.1	Water	Identified Properly	No errors
Experiment No.2	Water	Not Identified properly	Edges of the liquid were not detected due to a lack of sufficient light.
Experiment No.3	Water	Not Identified properly	No reflection of light present in the liquid.
Experiment No.4	Water	Identified Properly	No errors
Experiment No.5	Water	Identified Properly	No errors
Experiment No.6	Water	Identified Properly	No errors
Experiment No.7	Water	Identified Properly	No errors
Experiment No.8	Water	Not Identified properly	No reflection of light present in the liquid.
Experiment No.9	Water	Identified Properly	No errors
Experiment No.10	Water	Not Identified properly	No reflection of light present in the liquid.
Experiment No.11	Water	Not Identified properly	Edges of the liquid were not detected due to a lack of sufficient light.
Experiment No.12	Water	Identified Properly	No errors
Experiment No.13	Water	Identified Properly	No errors
Experiment No.14	Water	Identified Properly	No errors
Experiment No.15	Water	Identified Properly	No errors
Experiment No.16	Water	Identified Properly	No errors
Experiment No.17	Water	Not Identified properly	Edges of the liquid were not detected due to a lack of sufficient light.
Experiment No.18	Water	Identified Properly	No errors

Experiment No.19	Water	Identified Properly	No errors
Experiment No.20	Water	Identified Properly	No errors
Experiment No.21	Water	Not Identified properly	Edges of the liquid were not detected due to a lack of sufficient light.
Experiment No.22	Water	Identified Properly	No errors
Experiment No.23	Water	Not Identified properly	Edges of the liquid were not detected due to a lack of sufficient light.
Experiment No.24	Water	Identified Properly	No errors
Experiment No.25	Water	Identified Properly	No errors
Experiment No.26	Water	Not Identified properly	No reflection of light present in the liquid.
Experiment No.27	Water	Identified Properly	No errors
Experiment No.28	Water	Identified Properly	No errors
Experiment No.29	Water	Not Identified properly	Edges of the liquid were not detected due to a lack of sufficient light.
Experiment No.30	Water	Identified Properly	No errors
Experiment No.31	Oil	Identified Properly	No errors
Experiment No.32	Oil	Not Identified properly	Edges of the liquid were not detected due to a lack of sufficient light.
Experiment No.33	Oil	Identified Properly	No errors
Experiment No.34	Oil	Not Identified properly	No reflection of light present in the liquid.
Experiment No.35	Oil	Identified Properly	No errors
Experiment No.36	Oil	Identified Properly	No errors
Experiment No.37	Oil	Identified Properly	No errors
Experiment No.38	Oil	Not Identified properly	Edges of the liquid were not detected due to a lack of sufficient light.
Experiment No.39	Oil	Identified Properly	No errors
Experiment No.40	Oil	Identified Properly	No errors
Experiment No.41	Vinegar	Not Identified properly	No reflection of light present in the liquid.
Experiment No.42	Vinegar	Identified Properly	No errors
Experiment No.43	Vinegar	Identified Properly	No errors
Experiment	Vinegar	Not Identified	Edges of the liquid were not detected due to a lack of

No.44		properly	sufficient light.
Experiment No.45	Vinegar	Identified Properly	No errors
Experiment No.46	Vinegar	Not Identified properly	Edges of the liquid were not detected due to a lack of sufficient light.
Experiment No.47	Vinegar	Not Identified properly	No reflection of light present in the liquid.
Experiment No.48	Vinegar	Identified Properly	No errors
Experiment No.49	Vinegar	Identified Properly	No errors
Experiment No.50	Vinegar	Identified Properly	No errors

Table II has four columns- The first column is Experiment No. which contains a count of the number of each and every experiment done, Second column is liquid nature which contains the type of the liquid used in that particular experiment for identification, the Third column is identification result which contains the result of each experiment whether the spilled liquid was identified or not and Fourth column is Reason for Errors which contains all the reasons for which the liquid for not being identified. The table shows the result of all the 50 experiments done. 60% of the experiments were done on water, 20% were done on Oil and another 20% were done on vinegar. Among these 50 experiments, 66% was successful, 14% was not detected due to the absence of reflection in the liquid and 20% was not detected properly as the edges of the liquid could not be identified.

4.2. SOME PICTURES OF THE DETECTION OF LIQUID

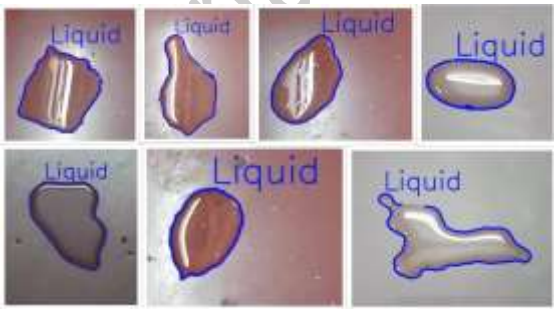

Fig 4.2.1. Some pictures of the results of the program on various liquids.

In Fig 4.2.1 some successful results of the detection of various liquids are shown. With proper light reflection and edge detection, the program is able to identify the liquid properly. Here some cases of successful edge detection of liquid are shown. 66% of the 50 experiments conducted in Table 4.1.1 were able to detect the liquid properly.

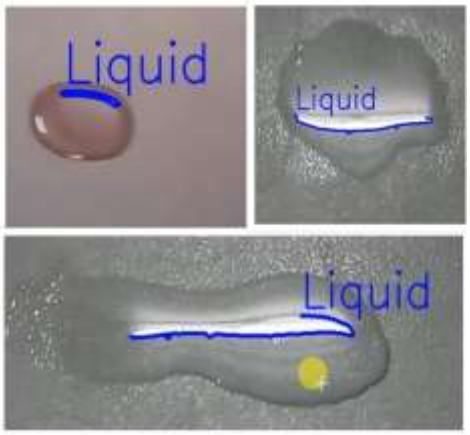

Fig 4.2.2. Some examples of errors in liquid detection

In Fig 4.2.2 some images where an error occurs in liquid detection are shown. These errors are caused due to problems in detecting the edges of the liquid. The problems in detecting edges of the liquid can occur because of an inappropriate angle or insufficient amount of light. If the edges of the liquid body are not detected properly then the program will mark the reflection itself instead of the liquid. As the highest edge contour encapsulating the reflection will be the contour of the reflection itself. 20% of the 50 experiments done in Table 4.1.1 had this error and could not detect the liquid properly.

Fig 4.2.3. Some examples of cases where the liquid is not detected

In Fig 4.2.3 some examples of cases where the liquid is not detected at all due to the absence of reflection are shown. Here the liquid body is not detected because there is no reflection present in the liquid body. As a result, the program is completely reflection-based and dependent is unable to detect the liquid body. Here three such examples are given in which the program is unable to detect the spilled liquid. 14% of the 50 experiments done in Table 4.1.1 were unable to detect the liquid due to the lack of reflection.

Fig 4.2.4. Some examples of wrong detections done by the program

In Fig 4.2.4 some examples of wrong detections done by the program are shown. Here the program is detecting some of the reflecting surfaces like mirrors and electronic screens that emit light that is close to that of reflection or itself reflects light. In these cases the program considered these to be liquid because it has a reflection and it satisfies the conditions set earlier that detected liquid properly. So some of the reflective surfaces or things that have a bright white color which is close to the color of reflection and satisfies the conditions for liquid detection coincidentally will also be detected as liquids.

As for the limitations of the program, there has to be a reflection of white light on the surface of the liquid. There has to be a sufficient amount and proper angles of light for the edge detection of transparent liquids. There should be an even surface for complete and proper detection of liquid. Objects or things with a color close to the color of a bright white reflection which coincidentally satisfies the other conditions will also be detected.

5. CONCLUSION

The goal of the program was to be able to identify spilled liquid with the help of the camera without using any machine learning algorithms. The program is successfully able to identify spilled liquid in 66% of the experiments as shown in the table in 4.1.1. 34% of the experiments were unsuccessful due to reflection and edge detection problems. Precisely, 14% of the experiments failed due to the lack of reflection being present in the liquid and 20% had errors due to the incorrect edge detections of the liquid. With the above results, we have shown how spilled liquid can be detected through the reflection property of liquid using simple methods like contours, edge detectors, and other image processing aspects of python.

ACKNOWLEDGMENTS

This paper would not have been possible without the expertise of my professor and guide Dr. Pranam Paul. He is an inspiration with great knowledge to me.

Also, my parents Mr. Sanjoy Pal and Ms. Madhuri Pal have supported truely to make this research stand, without their support this would not have been possible.

REFERENCES

[1] https://www.tutorialspoint.com/opencv/opencv_gaussian_blur.htm --Gaussian Blur

[2] https://docs.opencv.org/trunk/da/d22/tutorial_py_canny.html -- Canny edge detector

[3] E. S. Gedraite and M. Hadad, "Investigation on the effect of a Gaussian Blur in image filtering and segmentation," *Proceedings ELMAR-2011*, Zadar, 2011, pp. 393-396.

[4] https://www.google.com/amp/s/www.learnopencv.com/invisibility-cloak-using-color-detection-and-segmentation-with-opencv/amp/ -- Inrange function color detection

[5] https://www.google.com/amp/s/www.geeksforgeeks.org/find-and-draw-contours-using-opencv-python/amp/ --Contours

[6] Ilkin, Sumeyya & Hangişi, Fatma & Tafralı, Merve & Sahin, Suhap. (2017). THE ENHANCEMENT OF CANNY EDGE DETECTION ALGORITHM USING PREWITT ROBERT AND SOBEL KERNELS.

[7] https://www.thepythoncode.com/article/contour-detection-opencv-python -- Contours

[8] https://pysource.com/2018/03/01/find-and-draw-contours-opencv-3-4-with-python-3-tutorial-19/ -- Contours

[9] Puri, Raghav & Gupta, Archit. (2018). Contour, Shape & Color Detection using OpenCV-Python.

[10] https://github.com/ethantrokie/water-detection/blob/master/detectPuddle/Imagetransformations.py --Flood detection in streets project.

[11] https://www.bogotobogo.com/python/OpenCV_Python/python_opencv3_Changing_ColorSpaces_RGB_HSV_HLS.php –HSV color frame

[12] Hies, Thomas & Parasuraman, Suresh Babu & Wang, Y. & Duester, R. & Eikaas, Hans & Tan, Kok. (2012). Enhanced water-level detection by image processing.

[13] https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_colorspaces/py_colorspaces.html --HSV color frame

[14] https://stackoverflow.com/questions/53130370/how-to-access-phone-camera-using-python-script/53131060 --IP Webcam

[15] P. Santana, R. Mendonça and J. Barata, "Water detection with segmentation guided dynamic texture recognition," *2012 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Guangzhou, 2012, pp. 1836-1841, doi: 10.1109/ROBIO.2012.6491235.

[16] https://answers.opencv.org/question/176489/custom-contour-extreme-points-detection/ --Extreme points

[17] https://www.pyimagesearch.com/2016/04/11/finding-extreme-points-in-contours-with-opencv/ -- Extreme points

[18] https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_canny/py_canny.html - Canny edge detection