FEDERAL STATE BUDGET EDUCATIONAL INSTITUTION OF HIGHER EDUCATION
"SAINT PETERSBURG STATE UNIVERSITY"
(SPbSU)

Educational program "Engineering-oriented physics"

Coursework report
5th semester
"Automation of obtaining quantitative characteristics of the process of liquid drainage from a sample"

Completed by a 3rd year undergraduate student:
Tyuterev M. I.

Scientific adviser:
Ph.D. Vasilkov S.A.
Scientific consultant:
asp. Poluektova K.D.

Saint Petersburg, 2021

Table of contents

Introduction

The rate of water drainage from silicone rubber samples can provide additional information about the quality of the material. This issue is discussed in detail in the thesis by K.D. Poluektova. [1].

When studying the process of liquid drainage from samples, it becomes difficult to analyze frame-by-frame a large number of video images. As part of this course work, an automated method for processing video in real time is proposed.

The goal of the work was to quantitatively describe the process of liquid drainage.

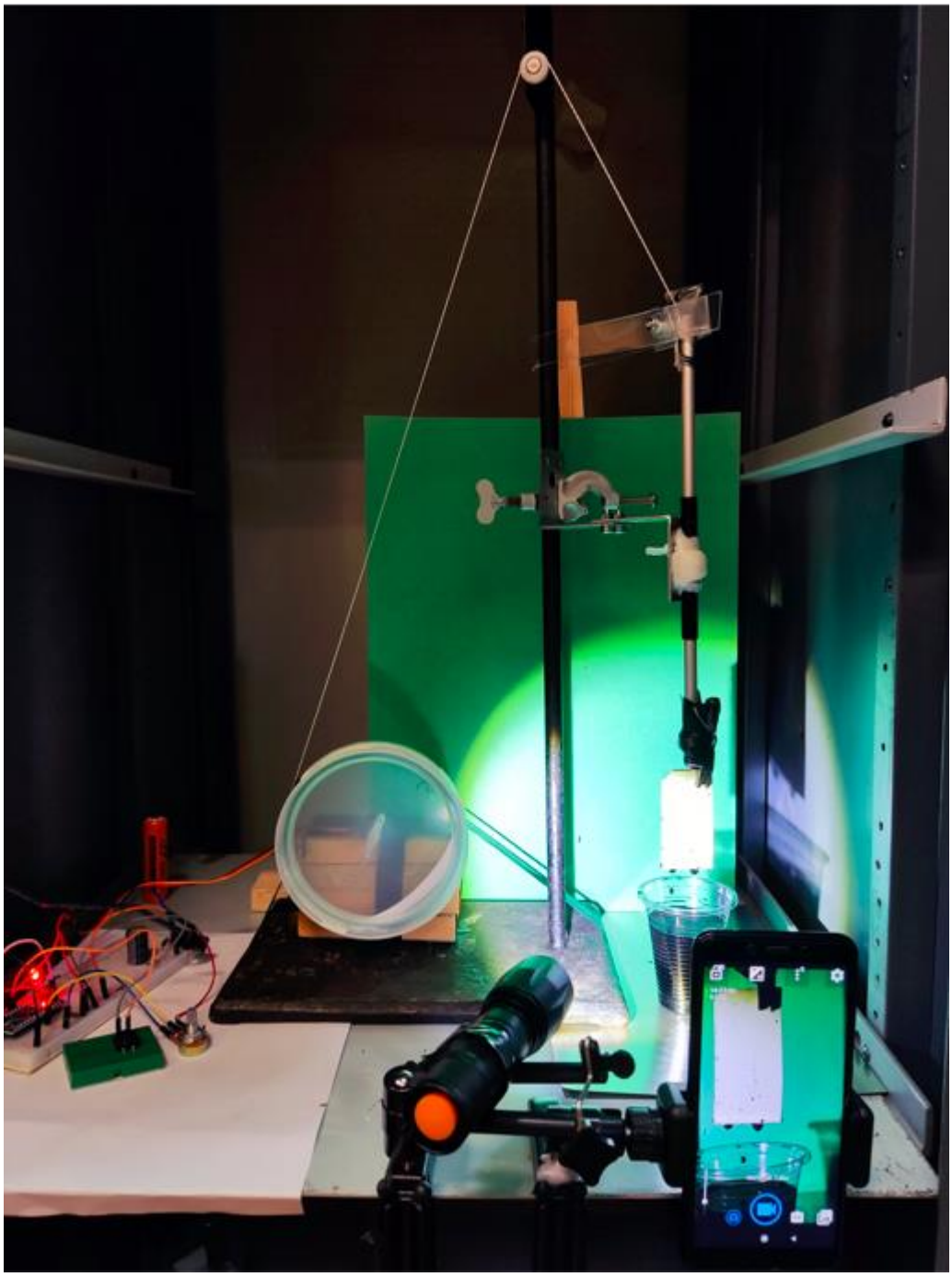
Tasks:

- Come up with a parameter for comparing samples;
- Write code;
- Modify the experimental stand to measure runoff;
- Conduct an experiment to confirm or refute the equivalence of ink and water when draining from a sample.

Main results of the practice

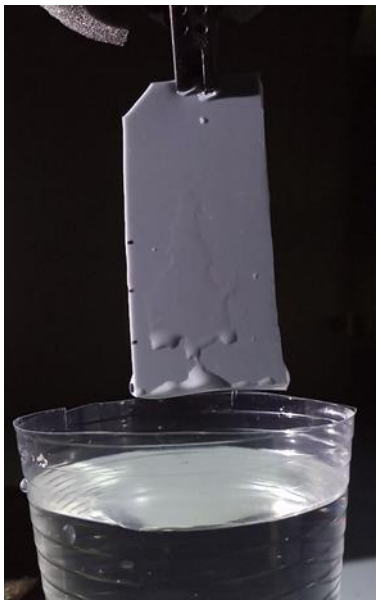
Experimental stand and its modification

As part of this work, the installation was modified, the final appearance of which is presented in Figure 1. The original installation, consisting of a lift that removes the sample from the liquid at the press of a button, was supplemented with a plain green A3 sheet mounted on a homemade stand and used as a background. A bright light source was also added, also mounted on a homemade stand, located at a certain point for a uniform incidence of light of maximum intensity.



Drawing1. Experimental setup

This setup had already been used in experiments, but with the equally distributed white light available at that time and the clear distilled water into which the sample was dipped, it was difficult to determine where the liquid was located: not only using the program, but even “by eye” (Figure 2).



Drawing2. Illustration of a faintly defined area of liquid on a sample

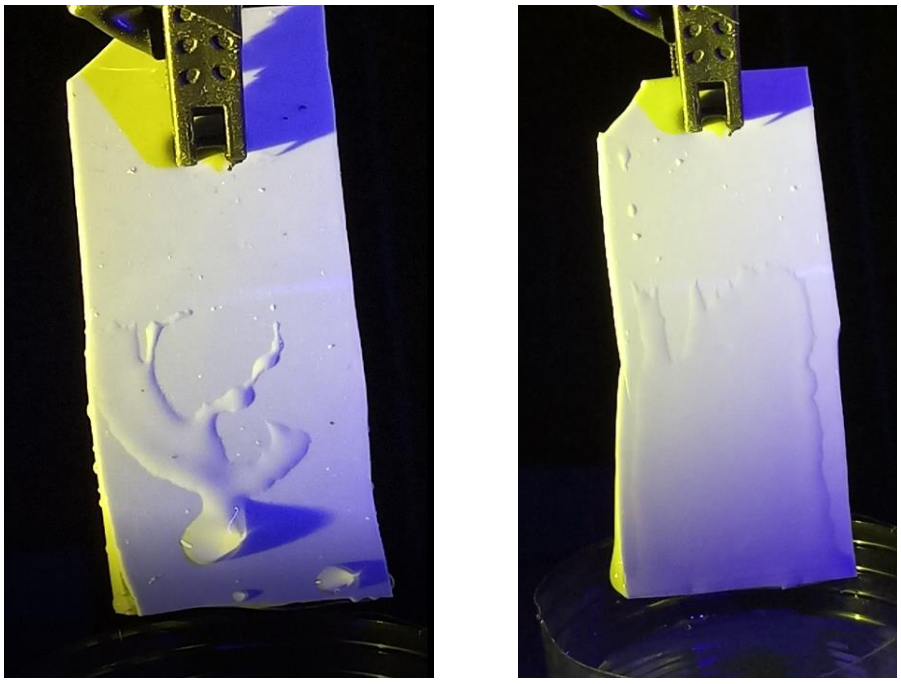
Therefore, it was decided to modify the setup so that the liquid, rubber sample and background were clearly visible.

How to isolate liquid from a sample

Figure 2 shows that clear water is difficult to isolate from the sample. There were 2 solutions to this problem: firstly, color the water with ink, but then you would have to prove that due to a certain concentration of ink in the water, its studied properties will not change.

You can also illuminate the installation using two light sources of different colors, one of which will be installed at a large angle to the normal of the rubber surface. Thus, it would be theoretically possible to ensure that the water in the profile, protruding relative to the plane of the rubber, would be illuminated by a bright source of a different color and there would be no need to carry out any manipulations with tinting the liquid.

This experiment was carried out in an attempt to isolate the liquid using only two sources, the light from which was passed through yellow and blue filters. The results turned out to be contradictory: on the one hand, with a fairly large contact angle, the liquid is clearly visible. On the other hand, if the surface is well wetted, a film of liquid forms on it and no relative arrangement of the sources makes it possible to observe a clear contour of the liquid over the entire area of the sample. (Figure 3)



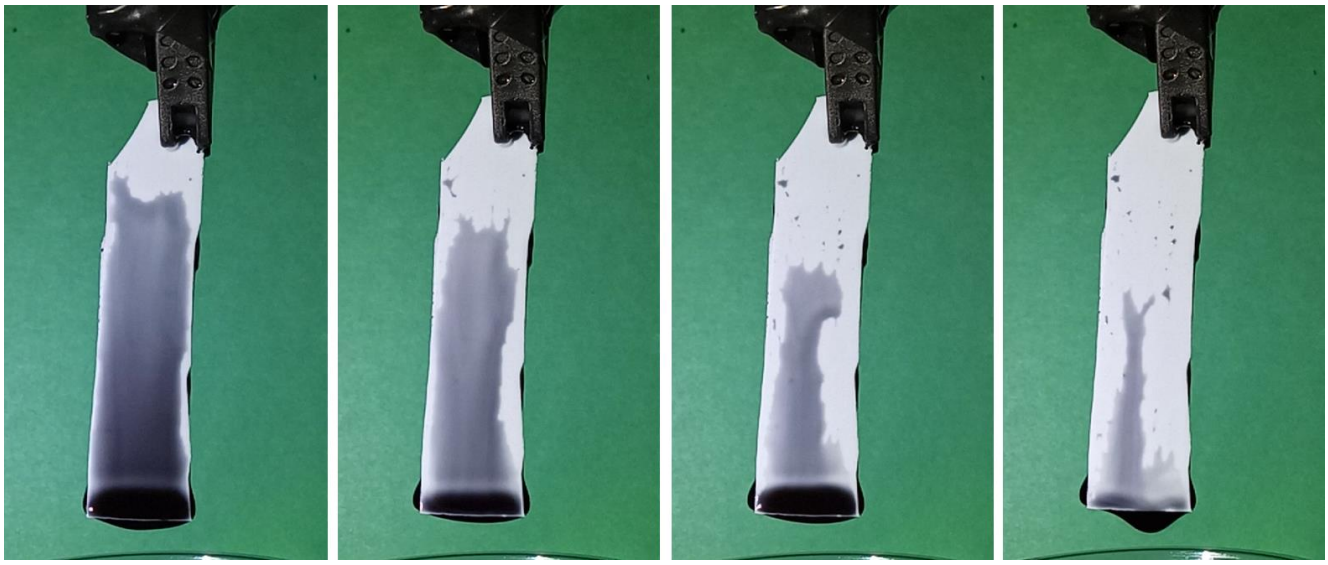
Drawing3. Liquid release using light sources: on the left - good, on the right - bad.

As a result, it was decided that the liquid still needed to be tinted with ink.

Experiment with the influence of ink

In order to show equivalence in the drainage of pure water and water with ink, 4 samples were soaked in the same distilled water, poured into cups, for the same amount of time. Then each of the samples was cut in half without removing it from the water. The first half was fixed to the installation, the sample was pulled out of the liquid, measuring the dependence of the time of its drainage on the serial number of pulling out. Then a similar experiment was carried out for the second half of the sample, but in this case the water was tinted with ink right before the experiment. Before the experiment, it was found that the ratio of 5 ml of ink to 95 ml of water is optimal between the concentration of ink and the visibility of the resulting solution. The drainage time data is presented in Table 1.

Of the 4 samples, liquid flowed down only two; a film remained on the rest. From the point of view of studying the equivalence of water without ink and with ink, the latter samples were not of interest and were excluded from consideration. An example of the experiment is presented in Figure 4.

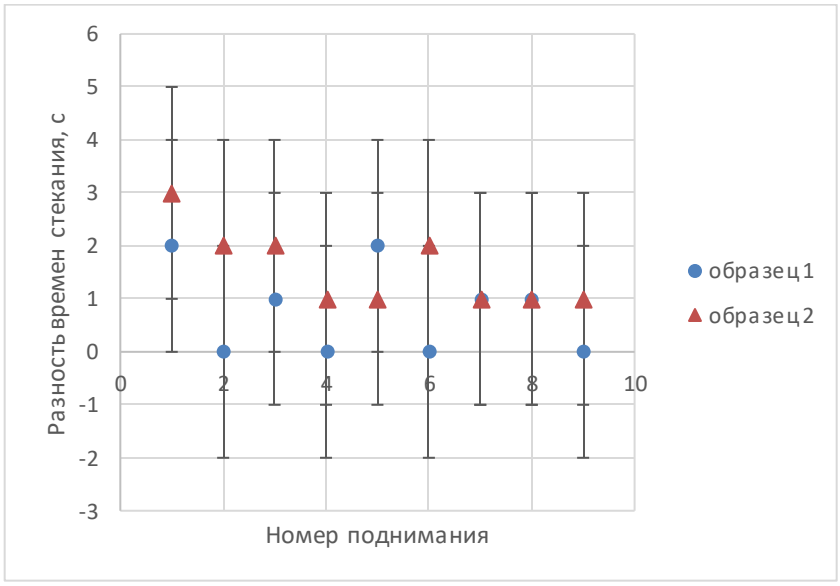


Drawing4. An example of an experiment with ink (experiment “ink 1” frames 1, 4, 6, 10 seconds)

Table 1. Dependence of drainage time (in seconds) on pullout number and sample.

NO.	PURE 1	CLEAN 2	INK 1	INK 2
1	10	22	12	25
2	4	6	4	4
3	4	5	3	3
4	3	5	3	4
5	4	4	2	3
6	3	4	3	2
7	4	3	3	2
8	3	4	4	3
9	2	4	2	3

The error in drainage time is assumed to be 1 second. Let us construct graphs of the modulus of the difference in drainage times from each sample. The errors are summed up accordingly.



Schedule1. Dependence of differences in drainage times on the number of rise

Based on the results of the experiment (graph 1), we can conclude that the presence of 5% ink diluted in distilled water does not affect the reliable drainage time of the liquid from the sample.

Writing code

The program execution cycle consists of three stages:

1) Receiving a video image and converting it frame-by-frame into black and white.
2) Thresholding an image to correctly detect fluid boundaries.
3) Executing the code and recording the dependence of the area occupied by the liquid on the drainage time (non-normalized) into the Matlab workspace for further conversion.

And now, first things first:

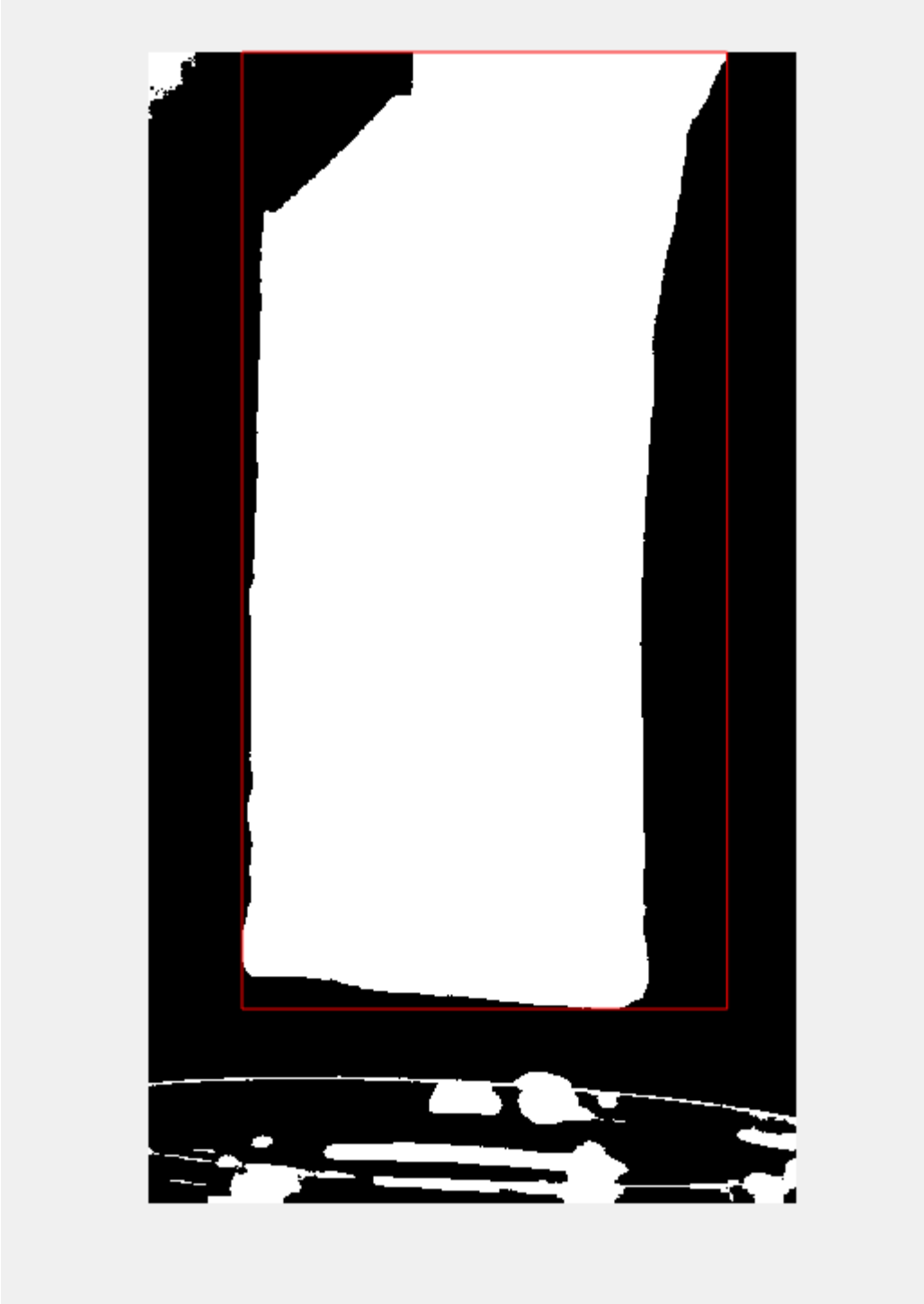
In the first part of the program execution, the image is read frame by frame and then, in a loop, 3 images are selected from each frame, each of which is written to its own array. This is a black and white image (Figure 5), and the intensities are blue and green.

Drawing5. Black and white image 50 frame video

Subsequently, from the difference in these intensities, an idea of the position of the rubber in the image will be obtained.

In the second part, the threshold value is selected. By default it is 0.65 and can be changed manually by the user in the range from 0.16 to 1 (0.15 is the basic threshold, it is also used. Below it there is no point in setting a custom threshold). First, the position

of the rubber itself in the image is found. It is defined as the maximum available object, including the background (Figure 6).

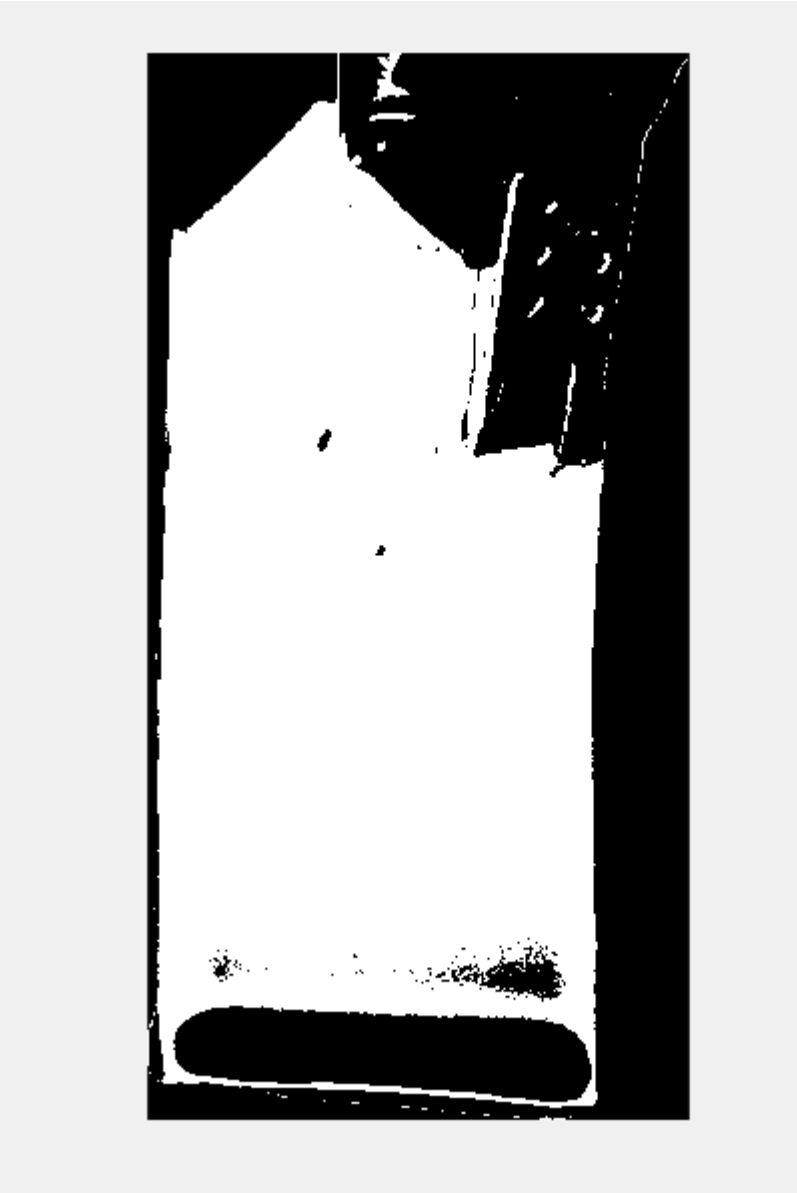


Drawing6. Selecting the area where the sample is located

This means that you need to shoot video so that the rubber occupies more than 50% of the image. A rectangular area is found in which the rubber is defined, and then the gray image is cropped, leaving only the desired rectangle. In addition, the gray image

is element-by-element multiplied by the inverse background image to immediately null the background inside the rectangle.

It is clear that not the entire rectangle is occupied only by liquid. There are at least more fasteners and the rubber itself that need to be removed. For this we need a threshold to leave only liquid. We define 2 binarization thresholds: basic (0.15) and custom.

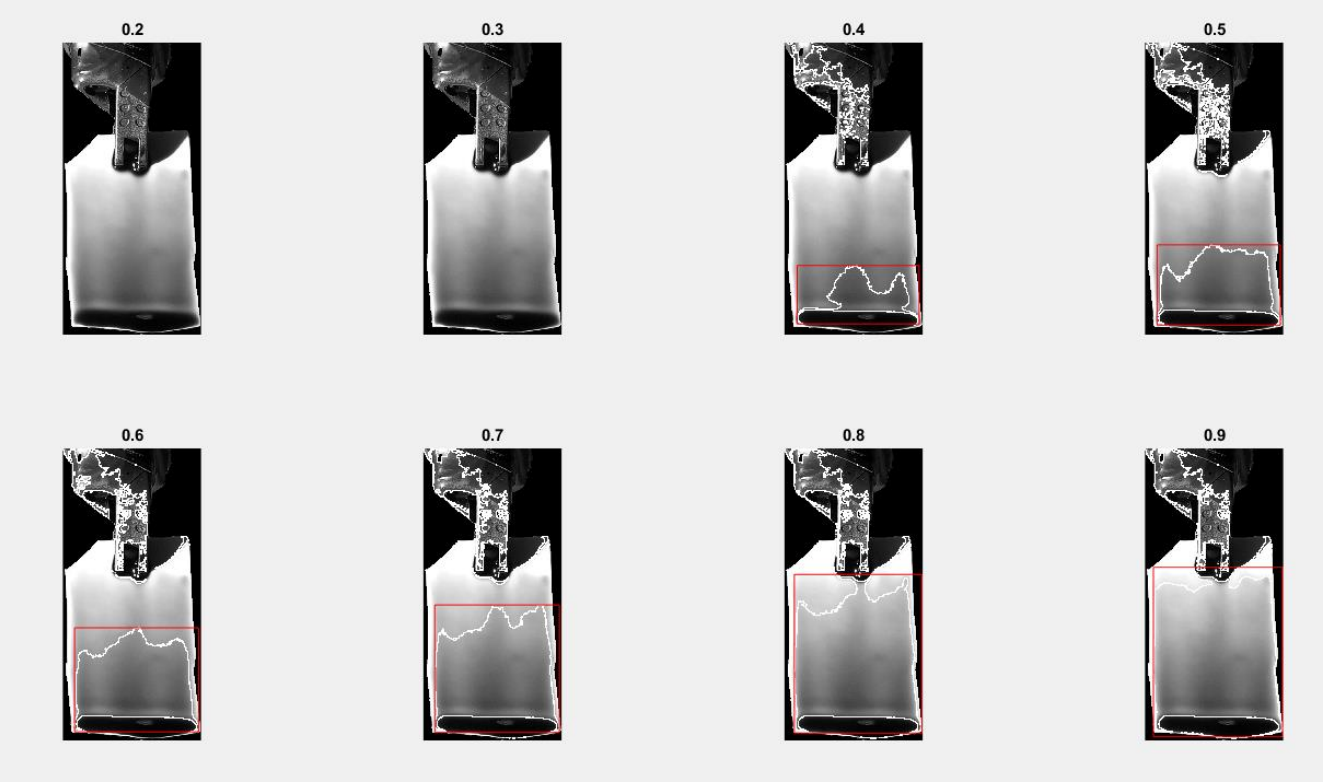


Drawing7. Determining the position of the rubber without fasteners. Applying a base threshold

The basic one locates all the rubber without fasteners (Figure 7), and the custom one locates the rubber not occupied by liquid (Figure 8).

Drawing8. Finding rubber that is not occupied by liquid. Using a custom threshold

The lower the user threshold, the less liquid the program will capture. On the other hand, the appearance of a granular selection, as in Figure 8 in the red circle, indicates that it is no longer worth choosing a higher threshold. Also in Figure 9 are contour plots of the released liquid depending on the value of the user threshold (the samples in Figures 8 and 9 are different). In this case, the optimal value would be 0.9

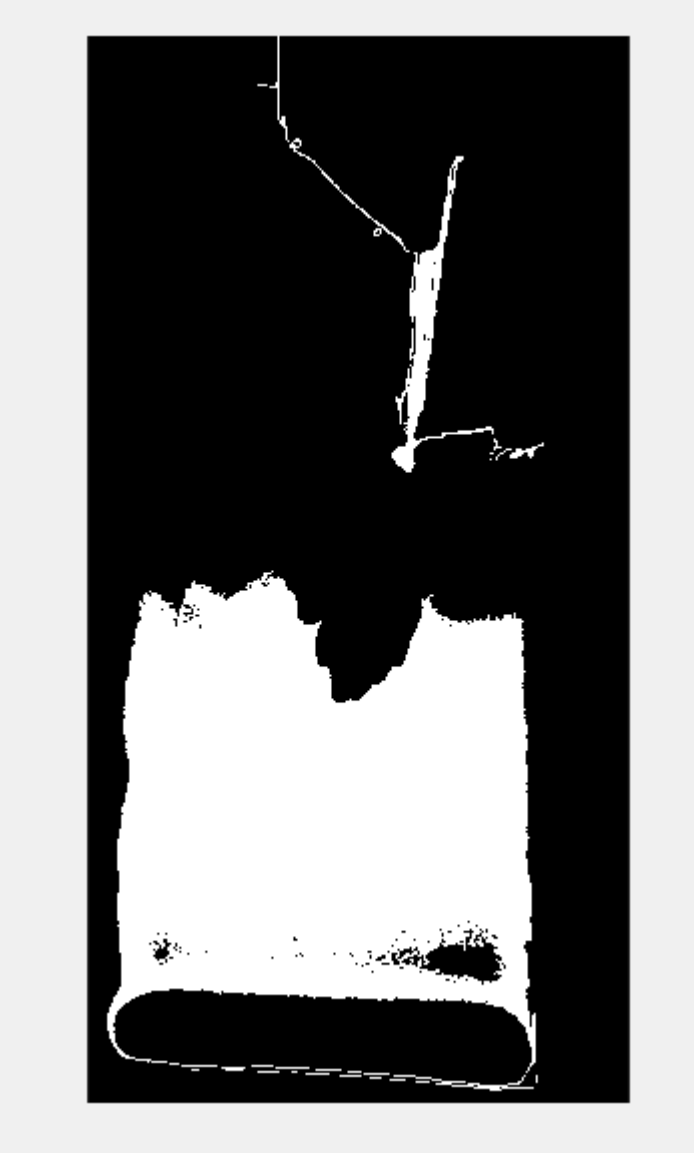


Drawing9. Liquid release depending on the selected threshold

Then, by subtracting Figure 8 from Figure 7 pixel by pixel, we can find the area occupied only by liquid (Figure 10).

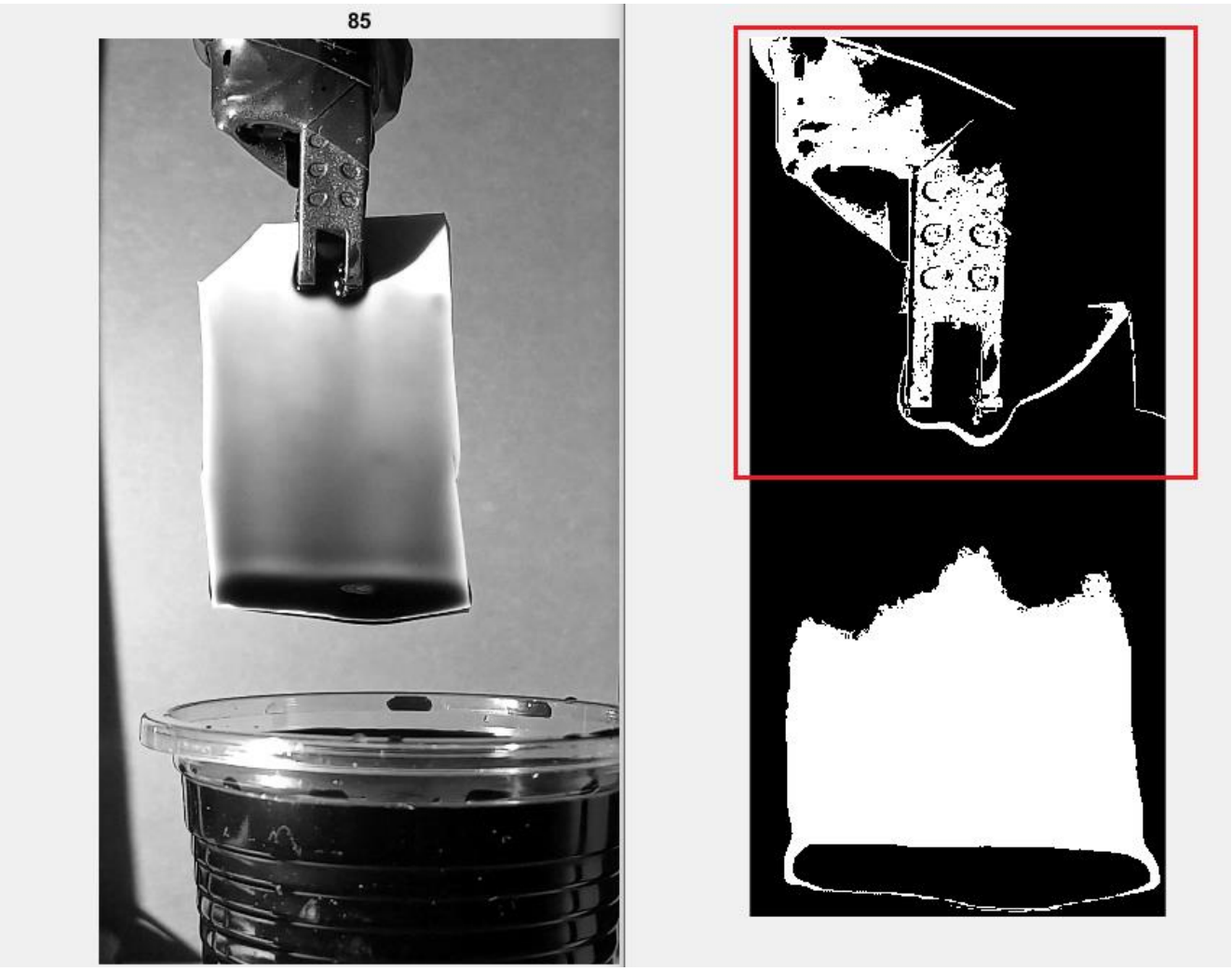
Drawing10. Water on the sample.

Now small white objects are removed: individual drops, glare from the mount, and the like (Figure 11). It was experimentally found that a threshold of 10,000 pixels is enough to remove most small objects.



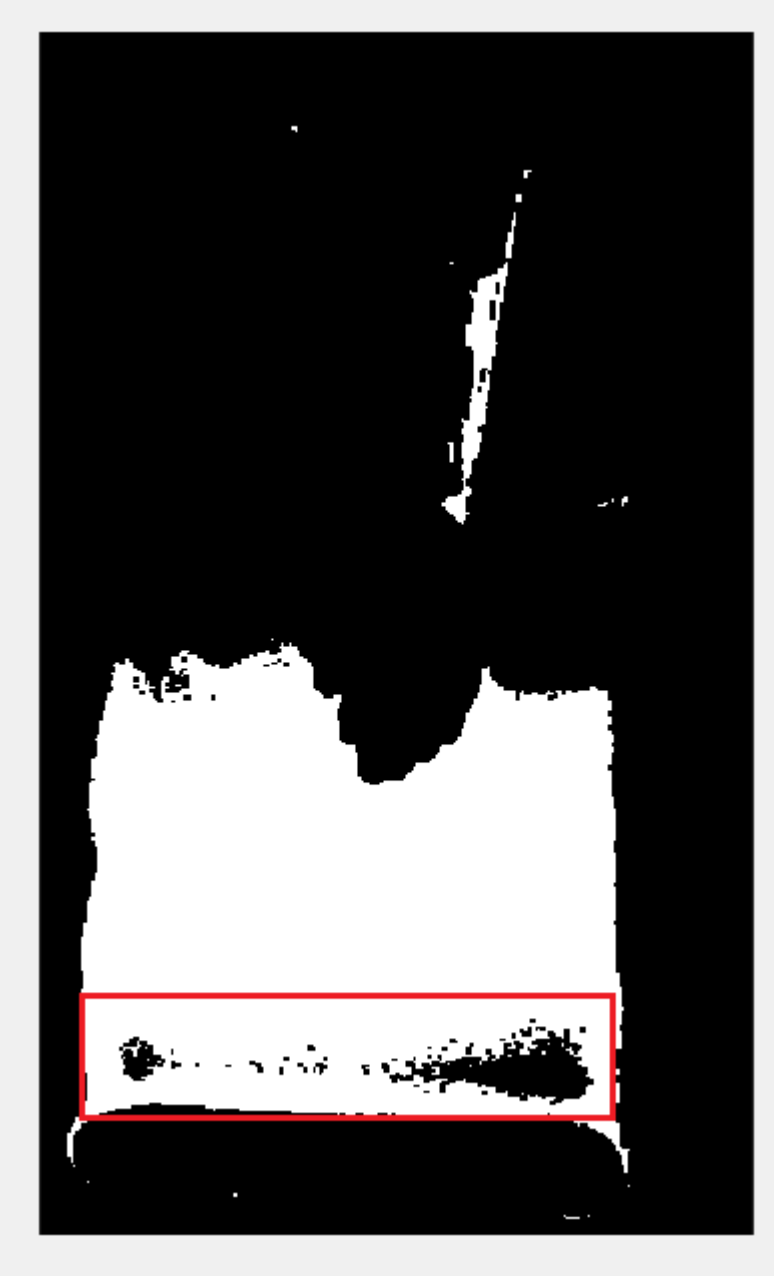
Drawing11. Removing small (<10000 pix) objects

Next, erosion (thinning of the boundaries) is carried out to avoid any closed contours in the fastener area. This problem occurs if shadows appear from various objects. A similar example is shown in Figure 12. Therefore, secure the rubber so that no shadow falls on it.



Drawing12. An example of a photo with a shadow. Part 2 – b&w image before dilatation

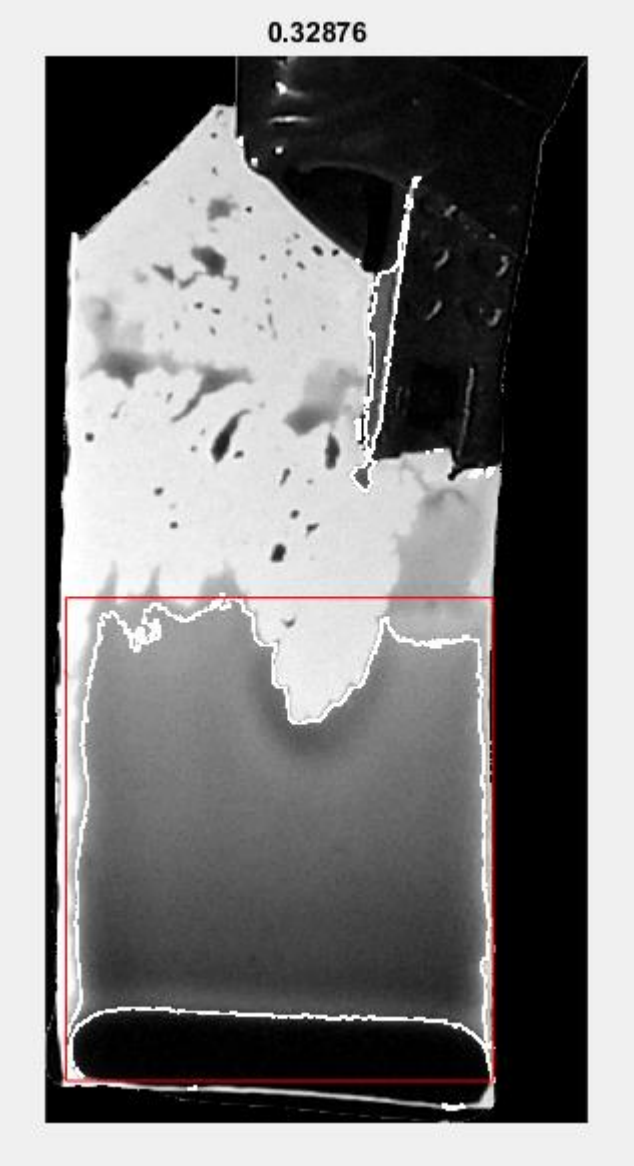
In Figure 12, the largest object is highlighted with a red rectangle. The shadow that appears on the sample from the fastener is perceived as part of this very fastener. And it may turn out that a single object of fasteners, background and shadow is perceived as a liquid (a single object of the largest size). This is a clear example of why dilation is needed. The result of applying dilation to the image in Figure 11 is shown in Figure 13



Drawing13. Applying dilatation and filling holes in objects

All that remains is to fill in the holes in the existing image (dark areas in the red rectangle in Figure 13).

We select the largest of the available objects and determine its area. At the same time, we pass the image through a high-pass filter, find the boundaries of the liquid and display all this on the screen to check the correctness of the choice of the threshold value (Figure 14).



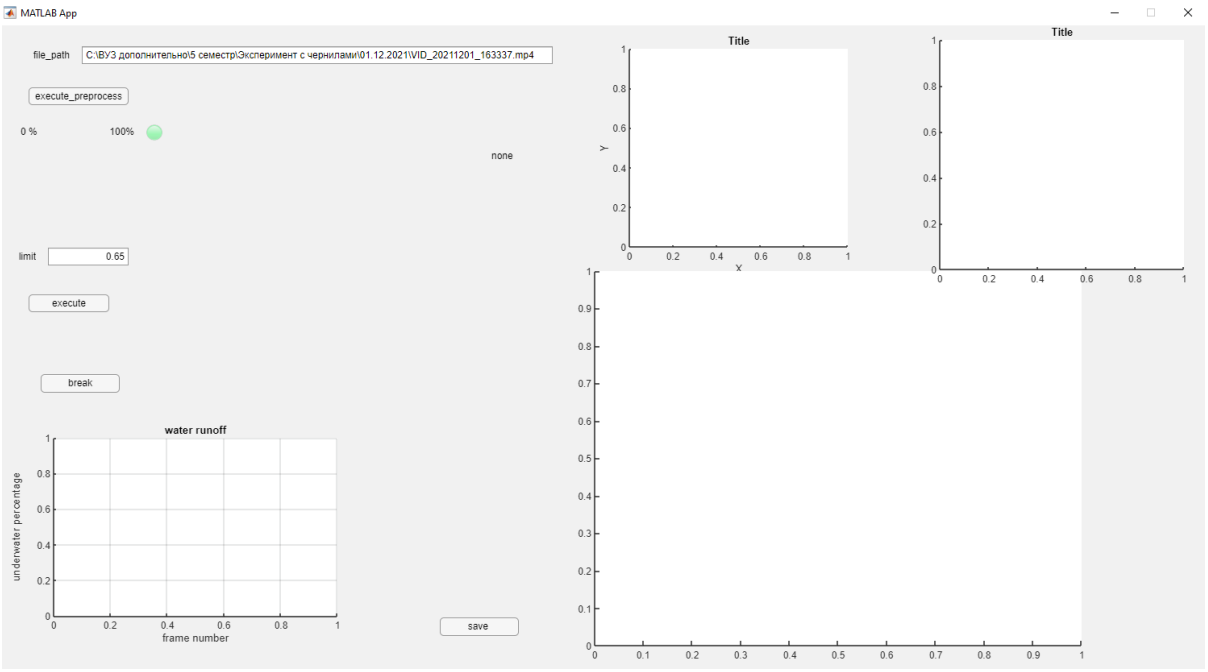
Drawing14. Obtaining the final image and quantitative characteristics of the area ratio

The area value displays the ratio of the area of the liquid to the area of the rubber together with the fastener. So the resulting graph will in any case need to be normalized to the first (maximum value) amount of liquid. This would still have to be done, because when the sample is lifted, not all of it is immersed in water. After saving, normalize the area ratio value. The normalization principle will be shown below.

The code was then translated into a matlab application to make it easier to work with.

Work plan in the application

The application at the time of writing the coursework looks like this:



Drawing15. Possible type of application

The lower left graph just shows the dependence of the filled area (unnormalized) in real time. 3 images are also displayed: on the top left - a gray image, on the top right - a binary image with rubber highlighted, bottom - the presence of water in the rubber area with an indication of where the liquid boundary is highlighted.

The first thing to do is add the full path of the file, including the name and extension. Make sure the slashes are correct. Correct "\". Then the video is preprocessed (part 1 of the code). This process is performed 1 time. In the future, you will no longer be able to press the button. To process another video, close the window and open the application again

As preprocessing progresses, the percentage completed is displayed below. When preprocessing is finished, the green button will light up. After this, you should proceed directly to the search for the threshold.

Set the threshold in the range from 0.16 to 1 (less than is meaningless, more than 1 is not allowed and there will be an error). After clicking the execute button, it is blocked and processing begins.

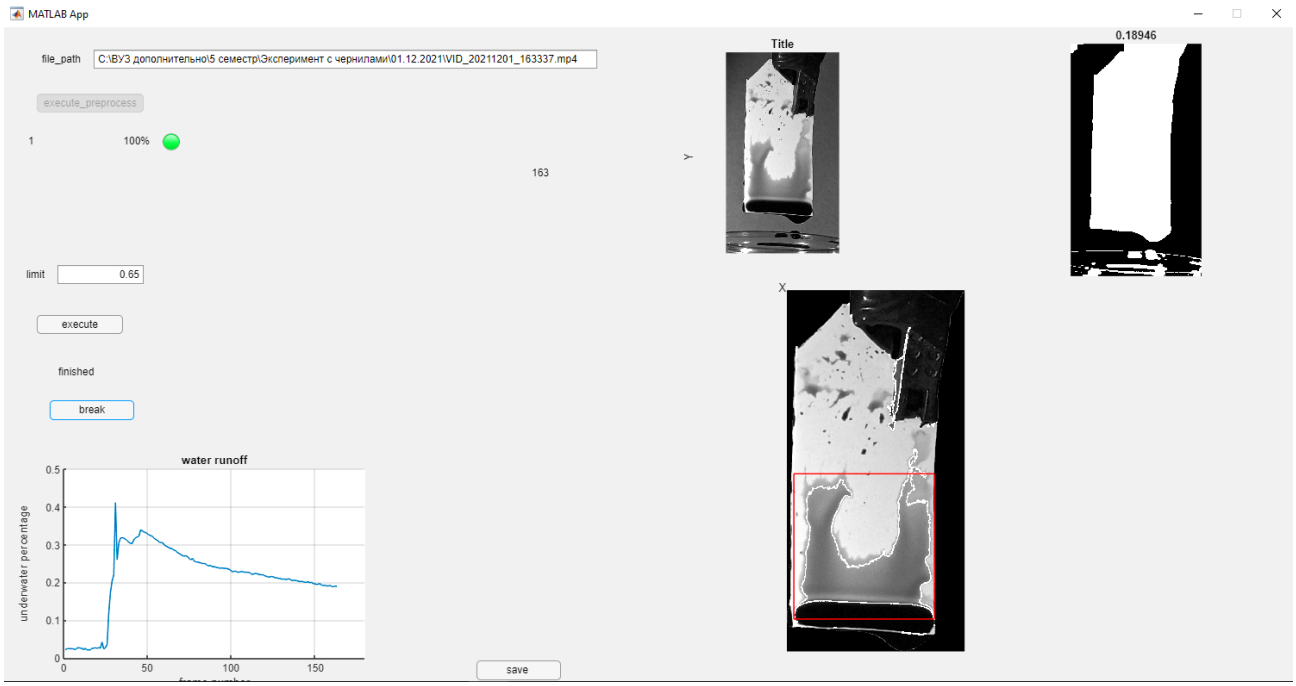
How well the processing is done can be seen in the large picture. The red rectangle defines the area where the liquid is located. The other 2 pictures are needed to assure the user where exactly the problem occurs (if it occurs, of course). The left image is a black

and white image of the video frame, the right is the position of the rubber along with the mount.

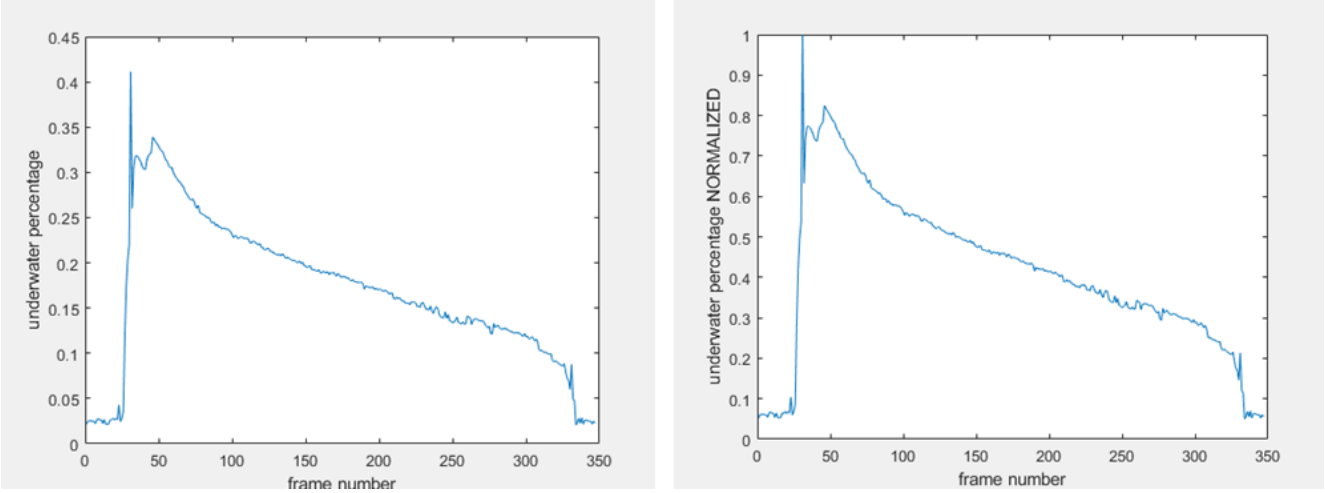
If the way the liquid is released suits everyone, then nothing else needs to be done. If something goes wrong and the threshold is selected incorrectly, then press the “break” button. The program execution stops and you select a new threshold.

If you need to stop the processing process and not process the video completely while saving the current results, then also click “break” and then “save”. The area dependence and black and white images will be output as variables in Matlab. It should be noted that area data is erased only when the “execute” button is pressed again.

Example of the program:



Drawing16. Example of how the program works



Drawing17. Graph of the dependence of the area occupied by water on the frame number. On the left – unnormalized (displayed on the graph in real time), on the right – normalized

It is important to ensure that the normalization element is physically justified. A situation may arise when the maximum value does not correspond to the area occupied by the liquid. The general principle is this: the maximum correctly determined area of the liquid is sought (in all experiments performed this is the maximum value of the array). It corresponds to the moment when the sample was just lifted from the water. Before the moment of lifting, some noise may appear, which theoretically can be greater in amplitude than the maximum area value. However, in all experiments performed they were insignificant. All transformations for normalization and division (if there are several experiments in one video) of the area data array are performed by the user outside the application.

The graph shows the general nature of the dependence. Around frame 40, the sample is lifted out of the water. The maximum value corresponds to the maximum value of the filled area. Then until frame 60 the data is invalid because the program misses some areas where there is liquid because they are too dark. Around frame 60-70, these areas disappear, and then a clear relationship is observed. It is similar to a hyperbola, however, in the region of frame 200, a process associated with the evaporation of water from the sample begins, as evidenced by jumps in the area values. Around frame 330, the sample is lowered into the water.

Conclusion

In this course work, an analysis was made of whether adding 5% ink to distilled water affects the drainage time. To do this, the existing setup was supplemented for better contrast of object boundaries and a series of experiments was carried out. It has been shown that the appearance of ink has virtually no effect on runoff. The error was no more than 2 seconds. A code was written in MatLab for automatic analysis of video of colored liquid draining from a sample. The code has been verified to work properly on a large number of recorded experiments.

List of references used and information materials

[1] Poluektova K.D. Master's thesis “The influence of discharge activity and moisture on the hydrophobicity of the surface of high-voltage insulators” 2021.

List of equipment used, including equipment Science Park of St. Petersburg State University

MATLAB software package

Applications

Execution codes when pressing buttons (MATLAB)

```
 Function execute_preprocessButtonValueChanged(app, event)
app.execute_preprocessButton.Enable ='off';
video = VideoReader(app.file_pathEditField.Value);
 % Video properties:
width = video.Width;
height = video.Height;
frameRate = video.FrameRate;
numOfFrames = video.NumberOfFrames;
 % *Following are all the constants that must be changed manually before starting the program:
cut_start=1;%write manually how many frames to remove at the beginning
cut_end=numOfFrames;%write manually how many frames to remove at the end
cropsize=[0, 0, width, height];%cut frames, specify [xmin ymin width height]
 % left upper corner of the sample is xmin, ymin
app.frames_gray = cell(1,cut_end - cut_start);%numOfFrames-(cut_start+cut_end)); % empty array to
make the loop run faster
tic
 fori=1:cut_end - cut_start%numOfFrames-(cut_start+cut_end) %50000 frames (30 minutes of 25 fps
video) will process 10 minutes, that is 100 frames in 1 second
k=i+cut_start;%removes the first frames
app.Label.Text =num2str(round(i/(cut_end-cut_start),2));
drawnow;
frames_crop=imcrop(read(video, k),cropsize);%frame cropping
app.fon_green{i} = imbinarize(frames_crop(:,:,2)-frames_crop(:,:,1));%finds green background
app.frames_gray{i} = imadjust(rgb2gray(frames_crop));%makes black and white instead of rgb
 end
app.Lamp.Enable ='on';
 end


function executeButtonValueChanged(app, event)
app.executeButton.Enable ='off';
app.areas = [];
app.Label_3.Text ='';
por = app.limitEditField.Value;
fori=1:length(app.frames_gray)%length(frames_gray)
ifapp.breakButton.Value
app.breakButton.Value = false;
app.executeButton.Enable ='on';
app.executeButton.Value = false;
app.Label_3.Text ='finished';
 break;
 end
app.noneLabel.Text =int2str(i);
drawnow;
arr = [];
arr1 = [];
drawnow;
imshow(app.frames_gray{i},'Parent',app.UIAxes2)
title(num2str(i));
```

```
h=fspecial('laplacian', .3);%element for high filter
BWdfill = ~app.fon_green{i};

feats = regionprops(BWdfill,'BoundingBox');
S = regionprops(BWdfill,'Area');
arr = zeros(1,length(S));
 forj = 1 : length(S)
arr(j) = S(j).Area;
 end
k = find(arr == max(arr));%search for element with maximum area
cropsize_new = feats(k).BoundingBox;
imshow(BWdfill,'Parent',app.UIAxes2_3);
 %rectangle('Position', feats(k).BoundingBox,'EdgeColor','r');
app.frames_gray{i} = app.frames_gray{i}.*uint8(~app.fon_green{i});
plate = imcrop(app.frames_gray{i},cropsize_new);%crop the initial image

ID = plate;
circ_se = strel('disk',3);%element for dilatation and erosion
ID1= imbinarize(ID,0.2);%all tires
ID2 = imbinarize(ID,por);% rubber without water
ID11 = ID1-ID2;%water without rubber
ID11=bwareaopen(ID11,10000);% removal of small objects
ID11 = imerode(ID11,circ_se);%erosion
ID11=imfill(ID11,'holes');%filling holes
bord = imfilter(ID11,h,'replicate');%finding the liquid boundary
bord = uint8(imdilate(bord,circ_se));% border line extension
bb = (256*bord+ID);% image with border marked
imshow(bb,'Parent',app.UIAxes2_2);
feats1 = regionprops(ID11,'BoundingBox');
S1 = regionprops(ID11,'Area');
 iflength(S1) == 0
app.areas(i) = 0;
 else%finding an object with maximum area
arr1 = zeros(1,length(S1));
 forj = 1 : length(S1)
arr1(j) = S1(j).Area;
 end
m = find(arr1 == max(arr1));
app.areas(i) = S1(m).Area/S(k).Area;
rectangle('Position',feats1(m).BoundingBox,'EdgeColor','r','Parent',app.UIAxes2_2);
title(num2str(app.areas(i)));
 end
plot(app.UIAxes,app.areas);

 end
app.Label_3.Text ='finished';

 end


function breakButtonValueChanged(app, event)
app.breakButton.Value = true;
 end

function saveButtonValueChanged(app, event)
```

```
assign("base",'percentage',app.areas)
assign("base",'picts',app.frames_gray)
 end
```