Contents

fragment streszczenia: The scope of this work is process of creating a device that automatically assigns broken eggs one of the three following classes: intact yolk, damaged yolk, clear line.
The implementation of such device will be preceded with research that compares various methods of recognizing the camera image, taking into account their performance in terms of computation resources, and in terms of correctness of the recognition

1 Introduction

1.1 Research scope

In 2014 creation of camera based module for chicken egg processing machines was requested by OVO-TECH company (the client).
The module is supposed to increase quality of egg-mass based products made with use of client machines.

During last 4 years multiple attempts were made by client to separate whites from yolks of freshly broken eggs in a mechanic way.
Obtained solutions proved to be working good enough for some companies, while other rejected them as imperfect and declared will to buy clients devices in the future, provided that adequate improvements will be made.

The scope of this thesis is overviewing the possibilities of resolving client needs with image processing mechanisms that will optically detect state of object (batch of intermediate product) and judge whether or not it should be discarded from the processing plant.
Moreover, a process of creating a prototype device using tested method is described.

Problems such as: - choosing image preprocessing methods - finding optimal parameters for used image alternations - overfitting issues - real-device solution implementation, - technical issues concerning the device choice, setup and performance had been taken into account during the research procedure.

Figure 1.1.1: Clients egg braking machines operating over the world

Figure 1.1 Clients egg braking machines operating over the world

1.2 Host machine

The machine that will be extended with the module prototype prepared in this thesis is OVO-TECH RZ-1 model.

Applications	**Egg breaker used to produce fresh liquid egg. Enable to separate white from yolks.**
How it works	The machine imitates work of people. Egg shells are cut and opened with use of special knives. Hygienic standard of broken eggs is very high.
Destination	Bakeries, confectioneries, egg processing companies.
Capacity	max 3600 eggs/h (separating function off) 1800 eggs/h (separating function on)
Power input	0.37 kW, 1 × 230 V, 50~60 Hz
Dimensions	900 × 1600 × 1100(h) mm
Minimal workspace	2000 × 1500 mm
Weight	ca. 140 kg
Operated by	1–3 people
Delivery time	5 weeks
Note	The machine includes quality control module disallowing weak yolks get into final product. The machine is equipped with yolks-whites separator, and also has 2 modes: fast and precisely.

Figure 1.2.1: Rz-1 unit baisic parameters

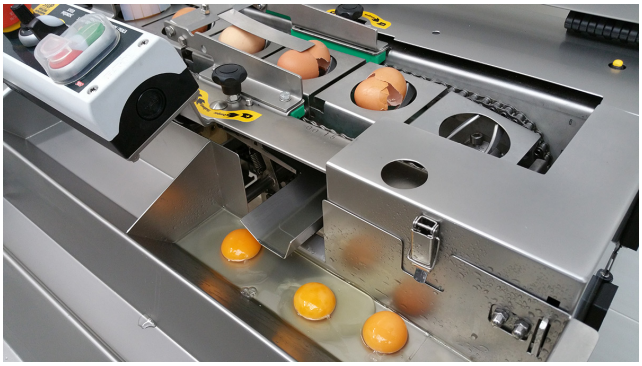

Figure 1.2.2: Egg cracking module of RZ-1 unit

The machine operates in a following way:

1. Eggs are directed into one-line flow, and mechanically aligned

2. Cracking module cuts the incoming egg surface from below with two knives that are aligned in direction of egg focal radius

3. The knives imitate human hands work by opening the previously notched egg

4. The cracked eggs are accumulated in a movable utensil for optical assessment. If the egg yolk appears to be broken, it is removed by an operator from the processing plant

5. Cracked eggs are transported by sliding to white and yolk separating module

6. Separation is done in a mechanic way while eggs slide over specificly shaped gap that only egg white fills in, while the egg yolks remain intact

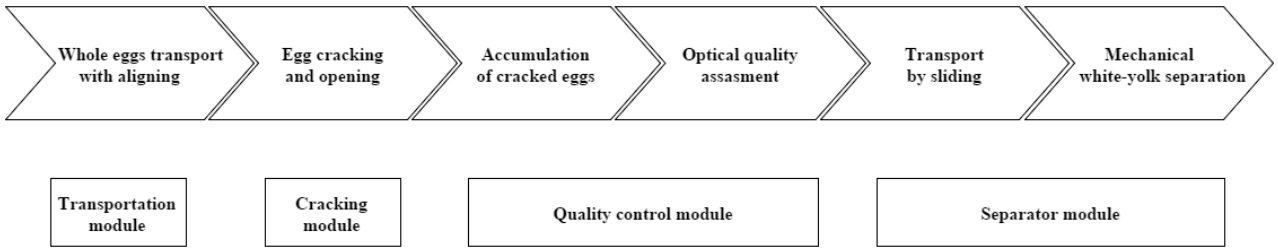

Figure 1.2.3: RZ-1 Egg processing - process and plant structure

RZ-1 operates in two modes: - Precision mode - Fast mode

Precision mode is used widely with bad quality eggs to extend amount of time for optical assessment in (5) phase.

More advanced OVO-TECH models such us RZ-3, RZ-6 and RZ-8 will be also considered for further research.

1.3 Problem description

If an egg processed by rz-1 machine is old or the chicken was feed improperly the yolk might be either broken already inside the egg, or might break during the cracking phase due to cracker imperfections such us being calibrated for different size or weight eggs.
A mixture of damaged yolk and white is created in a result, and itâĂŹs not possible to separate it into two initial components using the rz-1 separation module.

Batch of such mixture should be removed from the processing plant, since it would contaminate pure egg white product otherwise.

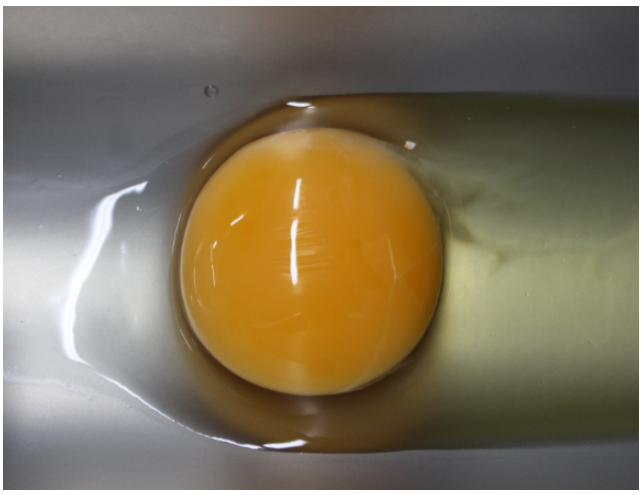

Figure 1.3.1: Properly cracked egg

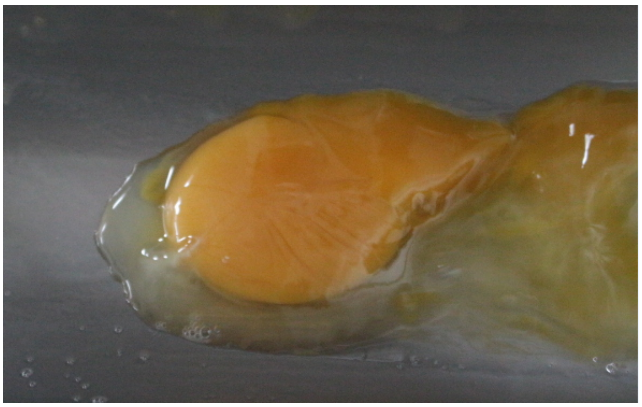

Figure 1.3.2: Damaged yolk in improperly cracked egg

1.4 Demand for improvements

Nine companies were inquired whether or not would they invest in improved Ovo-tech machine, provided that egg white will be visually clean of yolk parts.
Seven of them replied positively that they would seriously consider such offer since their processing plants would benefit in terms of product quality or manpower cost on such improvement. Remaining 2 companies did not answer the question.

One should consider why better separation is a requirement for those companies.
Following reasons were presented:

- Confectionaries and bakeries utilize egg white foam as an ingredient for their products. Egg white foam is obtained by aeration (also known as beating / foaming / whipping) process. Comprehensive process description was provided by one of the bakers:

"When air is incorporated into a liquid or viscous solution, the solution entraps the air bubbles, forming a foam. If the foam is stabilized by proteins, it leavens a food, increasing its height and reducing its density. The viscosity of all egg products is ideal for incorporating air cells during the whipping or beating process. As whipping or beating progresses, air bubbles decrease in size and increase in number, all the time surrounded by egg proteins. Liquid egg products have low air-liquid interfacial tension; thus, when eggs are beaten or whipped, the proteins denature, or simply, they unfold. This exposes two oppositely charged ends of the protein molecule: the hydrophobic, or water-hating end, and the hydrophilic, or water-loving ends. The proteins align themselves between the air and water, securing the air bubbles with their hydrophilic chains pointing into the water and dangling their hydrophobic chains in the air. During baking, these proteins bond with each other, forming a delicate, yet reinforced network.

Egg whites do this much better than yolks because of the unique proteins found in whites. In fact, even though the term foam technically refers to any system where there are entrapped air bubbles, in the food industry, when discussing egg white products, the term tends to be exclusive to egg whites foams. This is because egg whites, unlike any other natural food ingredient, are able to create the largest possible food foam, a foam six to eight times greater in volume than unwhipped, non-aerated liquid egg white

Whole eggs and egg yolks can also increase the volume of foods, including certain baked goods and dairy desserts such as ice cream and custard, but just not as much as egg whites alone. Visually, whipped yolks may double or triple in volume, while whipped whole eggs produce less volume than either yolks or whites whipped separately. They are also less thick than yolks alone." [1]

- Ovo-tech observed that a simple dependence exists: the older the egg is, the more likely

its yolk is degenerated inside the egg shell. Thus eliminating the batches containing fuzzy yolk should decrease chances of accidentally processing the contaminated eggs, while the risk of egg contamination generally increases with egg age.

- One of the Ovo-tech clients is a biotechnology company that manufacture medicine out of genetically modified eggs.
The company purchased some ovo-tech machines, but is nevertheless displeased with their separation ratio.
This company currently is undergoing a procedure of obtaining US Food Drug Administration (FDA) approval for wide distribution of their egg-based medicine.
Executives of this client suspect that using machines with better egg separation will increase their chance for positive outcome of above process.
A non-disclosure agreement was signed between this client and ovo-tech, thus further expanding this topic in scope of this thesis is not possible.

2 Problem analysis

2.1 Establishing framework for product cleanness assessment

The requirement is to create a system able to detect all badly-cracked eggs.
After request for clarification, client made 60+ pictures of product during machine operation and classified them manually, thus requirement is less ambiguous.

Client decided that intact egg yolks should be removed from processing line if:

1. intact yolk is surrounded by homogeneous mixture of another damaged yolk and white. This is the most strict (in terms of cleanness) case, that generates less strict cases:

2. intact yolk surrounded by damaged yolk blobs

3. fuzzy (damaged) yolk that still keeps its shape without mixing with white

4. damaged egg blobs surrounded with white

5. egg-white homogeneous mixture

6. egg-white nonhomogeneous mixture

Two cases are accepted as proper product:

7. intact yolk surrounded with egg white

8. clean egg white also, there is no need to do anything when:

9. transportation part is empty (machine is not operating or the operators are preparing next batch of eggs).

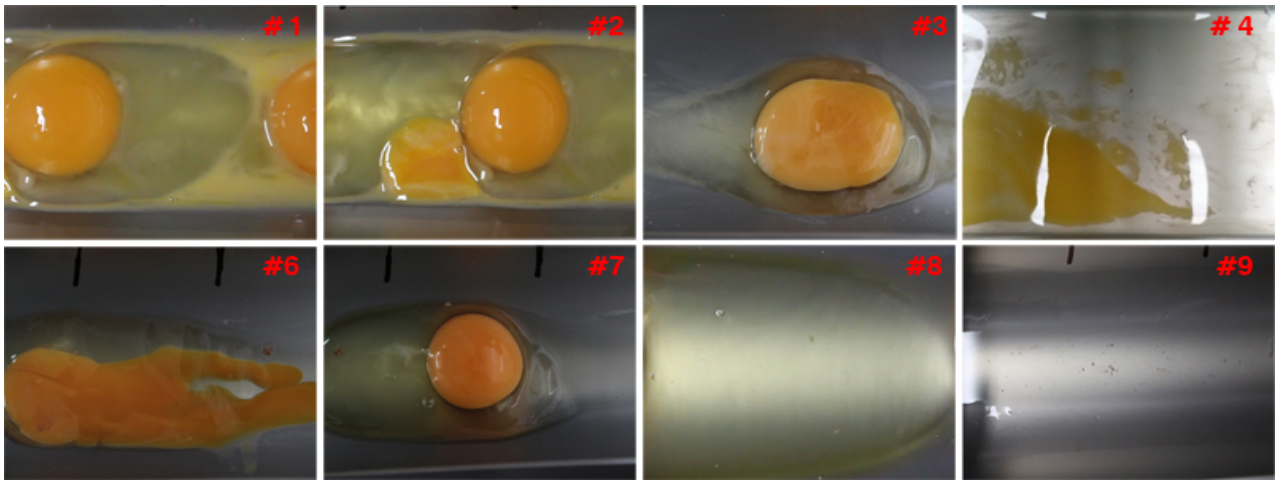

Figure 2.1.1: Cracked egg product cases. Source: own materials

Source: own materials

Also, a problem is constituted by chalaza - a usually white element that is attached to egg yolk for in-egg suspension.
Despite the fact, that client decided that chalaza position and amount are irrelevant to him, its appearance may increase difficulty of the recognition problem if identified as parts of damaged egg yolk.

Figure 2.1.2: Chalaza attached to egg. Source: own materials

2.2 Techniques of automatic egg quality assessment

Different techniques for automatic, non-destructive egg quality grading have been investigated. Among them, the following are worth mentioning: - Intelligent systems based on visible-infrared transmittance spectroscopy[2] - Fourier frequency analysis of a vibration-based response on impact force[3] - Ultrasonic, magnetic resonance, electric resistance, electric nose[4] All methods listed above require very specific equipment, are prone to overfitting and take in to account assumption that does not has to be satisfied for problem considered in this thesis: the egg shell should not be damaged.

Since in this very special case, eggs are already broken, simplified and different methods can be considered. Initially, two ideas were researched by author of this thesis: - Proximity sensor utilization - Using laser / narrow light beam Both methods would identify the intact

yolk basing on its height and wouldnâĂŹt require much processing (an Arduino Uno or other ATMega based circuits might be sufficient).

Nevertheless, both of them had to be rejected, since intact yolks presence in the tested batch is not sufficient for batch accepting âĂŞ absence of egg yolk parts is also required (see subchapter 2.1).

Finally, using images of the batch obtained from camera and processing it was chosen as a most promising option for further research.

3 Batch state detection system

3.1 System design

Firstly the preprocessing will be done to eliminate unimportant data (reduce the dimensionality), reduce noises and extract interesting image areas.

Secondly the image will be segmented to extract egg yolk (whole and damaged).

Thirdly two options will be tested:

- The undamaged egg yolks will be detected and deleted from the image. Than the amount of remaining yolk parts will be counted. If it will exceed the threshold value, the signal for removing the product batch will be sent.
- The extracted egg yolk will be graded in terms of shape similarity to a single, intact yolk

The best obtained combination of preprocessing, segmentation and grading parameters will be used in a final algorithm that will be implemented in Raspberry Pi 2 device with camera module, and than tested in the Rz-1 processing plant.

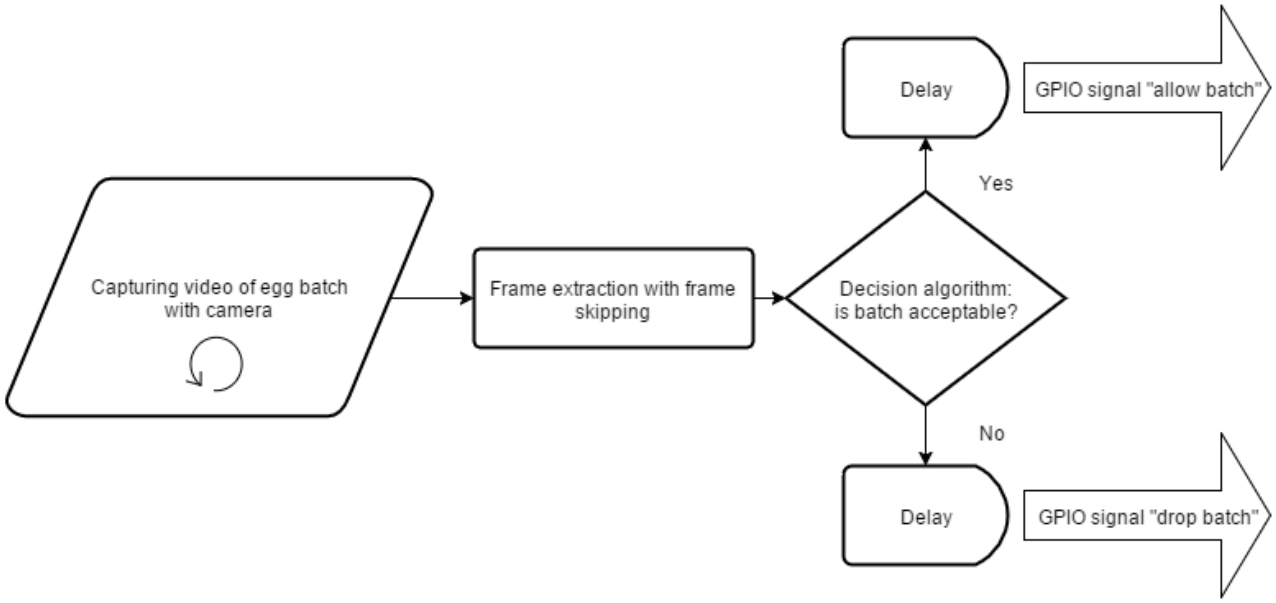

Figure 3.1.1: System design

3.2 Detecting batch state with camera image

When it comes to assessment of camera image, 4 approaches are widely considered:

1. Very easy scanning with photo sensors or very low resolution cameras

2. Computer graphics preprocessing and segmentation accompanied with assessment of a parameter (number of pixels, number of feature points, percentage of one region that fills another bigger region and similar)

3. Machine learning approach

4. Neural networks which are used as a special case of either machine learning process or a part of it

First case was considered, since it could be implemented on Arduino Uno, other ATMega based platform or real-time systems working on PLC controllers such as LOGO! 7, which are often used in factories due to their reliability (apart from Stuxnet-class worms, that has been firstly observed in 2010, no other security threads are believed to exist).
Unfortunately, the problem is more complex than simply deciding whether the product is present or not; or whether the product is yellow or not, thus more advanced processing is required.

Second case will be expanded in the next subchapter and has been chosen as main method that will be implemented and tested in this thesis.

Last two approaches will be tested if the 2nd approach will provide to be insufficient.
It is worth mentioning, that for (3) Support Vector Machines was considered, since it classifies data to exactly two classes (i.e. proper and improper egg mass).

Also, Haar Features Cascade was tested, but data amount obtained for testing was insufficient, and the methods popular implementations are focused on locating the particular object on image rather than on comparing two pattern cases.
(4) Case was considered and the author build a multilayered perceptron network working with MNIST database in order to better understand the method.
The method application will be researched further on beyond the scope of this work.

3.3 Detection algorithm

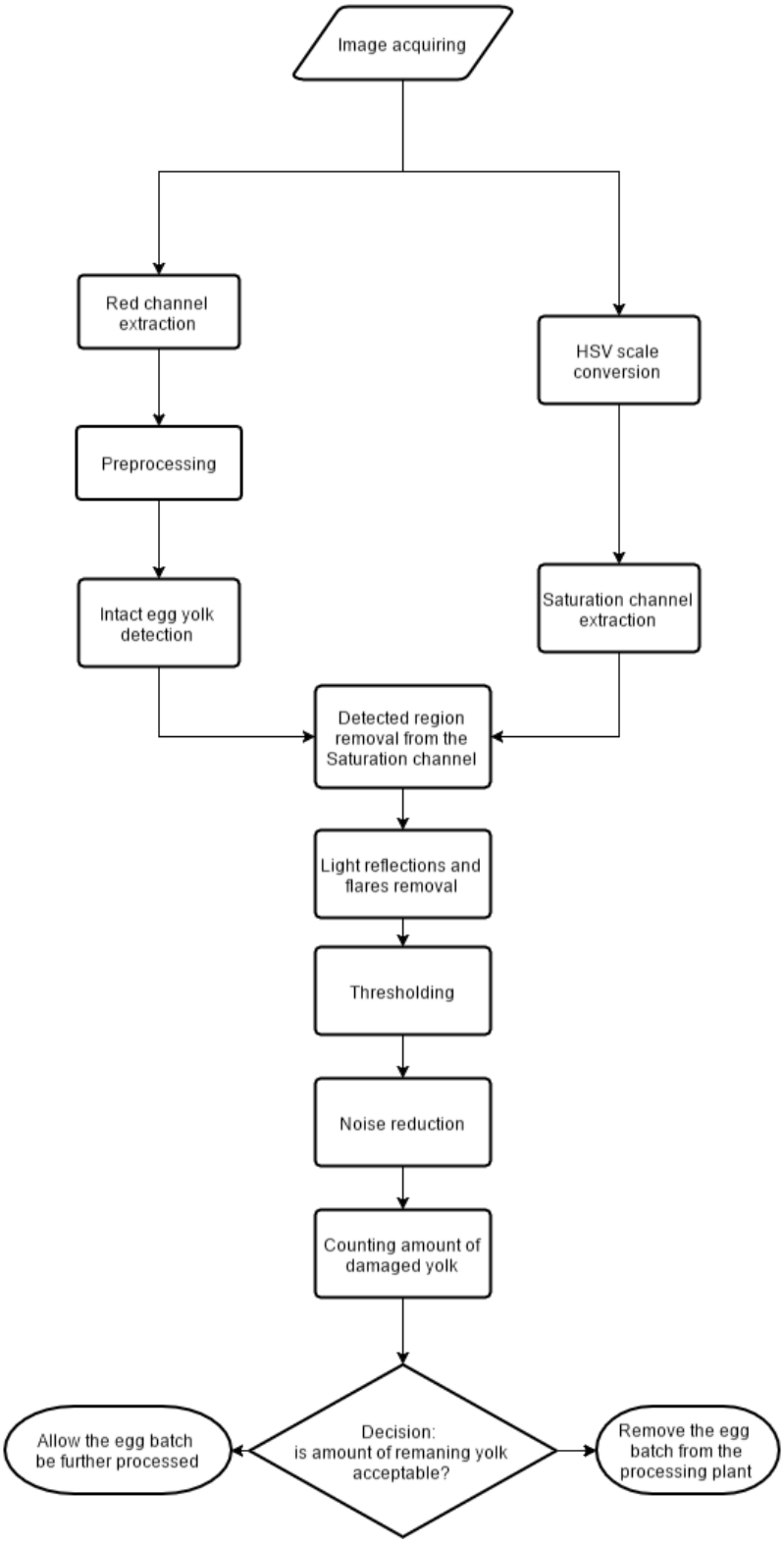

Figure 3.3.1: Detection Algorithm

3.4 Algorithm programming

Decision have been made, to utilise OpenCV computer vision library for batch state recognition.
OepnCV is designed for computational efficiency, with strong focus on real-time applications.
It has been written in optimised C and takes advantages of multicore processors.
It provides simple-to-use interfece that enables building sophisticated computer vision solutions.
The library features more than 500 functions that process images, a general Machine Leraning Library (including Multilayered Perceptron Networks).
It is used since 1999 by major institutions such as Stanford University and is mantained and developed by specialists from companies such us IBM, Microsoft, Intel, Sony, Siemens, and Google.[5]

First recognition attempts were done by the author of thesis using Android 4.3 running on Samsung N7100 Galaxy Note II smarpthon, utilising its build in camera.
The initial code was written with use of Java distribution of OpenCV library, as well as Android SDK.
Sadly, the performance of developed pre-prototype was very poor and a single frame was processed in more than 3 seconds.
Another version was build in a way that most resource consuming operations (such us Hough Circle Transform and all array operations) were rewritten in C language.
For that purpouse original C version of OpenCV was utilised, as well as NDK (Native Development Kit for Android).

Nevertheless, author of this thesis, decided that despite Android aviability for embedded platforms, it is no a platform suitable for factory tasks, due to big overhead, long booting and variety of functions that will not be utilised.

Raspbian Linux "Jessie" distribution was picked as the adequate one to host recognition algorithm accopmanied with Python 2.7 language.
Python is well supported on Raspian, it produces small amount of code, and can be executed fast in terms of computational time if the speciffic C-written functions are called.[6]
For author, a case occured, that 39-lines code snippet that recognise nested regions of image, written in C took only 6 code lines after rewritting it in Python.

4 Egg yolk detection

4.1 Methods overview

An egg yolk, when intact and fresh can be approximated with sphere which projects as circle-like shape on 2D image.
Methods presented below enable to filter out the unnecessary noise, simplify the image and detect the yolk shape.
While not all of them were used in the final detection algorithm, they were giving promising results on the early stage prototypes.

Following methods treat images as 1, 8, or 24-bit encoded matrices, thus both terms are used synonymously.
x and y denote pixel position.

4.2 Circle Hough Transform (CHT)

Circle Hough Transform is an algorithm used for retrieving 3 parameters defining a circle-like shape on images[7] : - (xc, yc) - position of shapes center - r - its radius

To understand the way CHT works, simplified case is considered:
The image is 1-bit encoded; the radius r and position (xp,yp) of a point lying on the searched circle are known.
Potential circles space is created by treating points on a circle of radius r centered in (xp,yp) as centers for potential circles.
For every potential circle, number of intersecting points is counted.
The potential circle with largest number of intersections is chosen as the best fitting one.

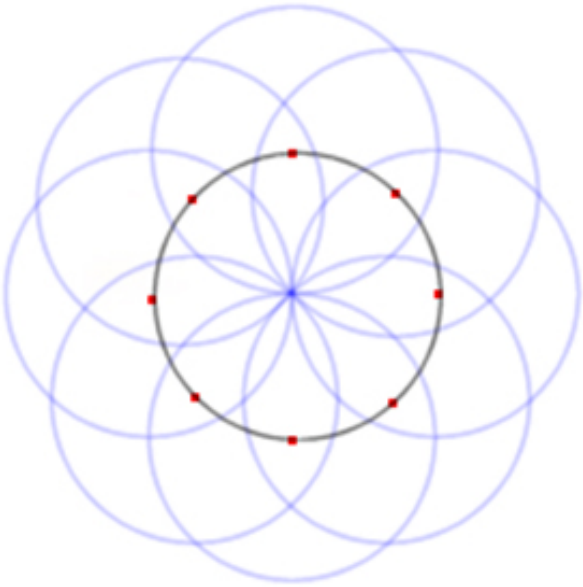

Figure 4.2.1: Potential circles space[8]

In the full cases, such procedure is repeated for every image point, and the intersections numbers are stored in accumulation matrix.[9]
To reduce computation time and false detection rate, minimum distance between image centers is given as a parameter min-dist.
Threshold value param-2 is defined as minimum number of intersections to treat circle candidate as detected circle.

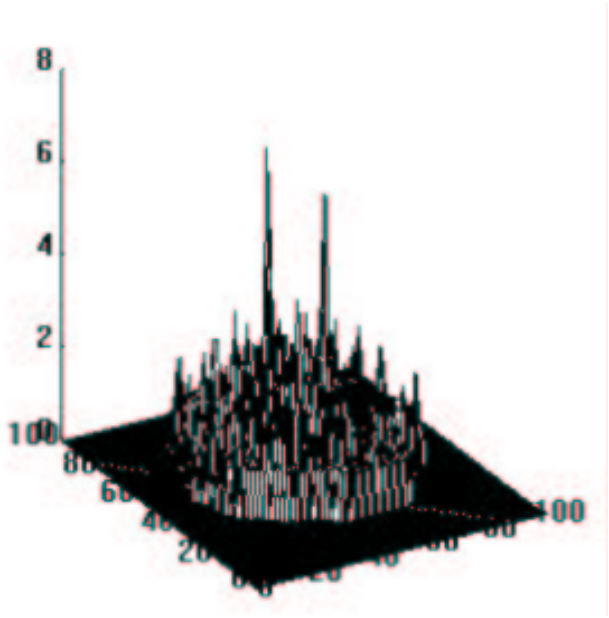

Figure 4.2.2: Accumulation matrix with two good circle candidates[9]

If r is unknown, min-radius and max-radius parameters are defined, and the algorithm repeats the procedure for all possible integer r in range of above parameters.
OpenCV utilizes Canny edge detector for finding the potential intersection points, thus operates not only on 1-bit images, but also in 8-bit grayscale and 24-bit bgr images and provides smaller number of false positives.[10]
param-1 is introduced as higher threshold for Canny edge detector, while the lower threshold is set to 0.5 * param-1[11] .

4.3 Erosion and Dilatation

Erosion and dilatation are two morphological filters that enable to filter out unnecessary noise from the image.
Combined, they leave the core information (biggest blobs) intact or amplified, while small elements are deleted.
It is easily applied to binary images:
A ROI (region of interest, a rectangular part of image) moves through image pixel by pixel and is compared to previously defined kernel (binary matrix, usually symmetric with respect to both diagonals).

```
0 0 0 1 0 0 0
0 0 1 1 1 0 0
0 1 1 1 1 1 0
1 1 1 1 1 1 1
0 1 1 1 1 1 0
0 0 1 1 1 0 0
0 0 0 1 0 0 0
```

Figure 4.3.1: A diamond-shaped kernel[12]

The consecutive pixels of on ROI that matches kernel elements with value 0 remain intact.

Pixels of ROI that matches kernel 1-valued elements changes to:
1-ns if at least one of them is 1 for Dilatation
0-s if at least one of them is 0 for Erosion.

Erosion gives the effect of âĂŸshrinkingâĂŹ the objects (and if they are small enough, they are deleted).
Dilatation makes the objects bigger.

Figure 4.3.2: Orignal image, eroded image, dilated image.[13]

When these methods are used multiple times, and convoluted, they filter out the noise. Perimeter Determination and Skeletonization are widely used contour detection functions which utilize erosion and dilatation[13].

4.4 Gaussian blur

Gaussian blur is obtained by convoluting pixel with gauss function values.

It reduces noise, detail and filters out the high frequencies what makes it a low pass filter.[14]

Gaussian blur drastically reduces number of false-positive Hough circles detections. It nevertheless should be coupled with sharpening to reduce false-negative cases.[15]

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Figure 4.4.1: Gauss function in two dimension[16]

sigma, ksize.width, ksize.height parameters are introducted.
Ksize stands for size of precomputed kernel, that is convoluted with ROIs on the image.

The term âĂŸconvolutionâĂŹ might be misleading and for the scope of image processing tasks is defined as picewise multipling two matrices and than summing the obtained values:

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} * \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} = (1*a)+(2*b)+(3*c)+(4*d)+(5*e)+(6*f)+(7*g)+(8*h)+(9*i).$$

Figure 4.4.2: Convolution of 3x3 matrices[17]

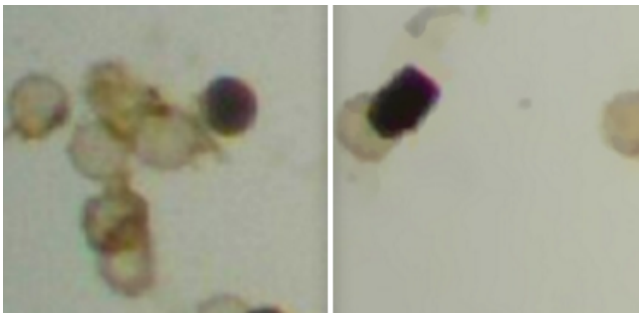

Figure 4.4.3: Microscope image before and after applying gaussian filter[15]

4.5 Channeling and HSV conversion

lorem ipsum

4.6 Using the methods

```
r_c = cv2.GaussianBlur(r, (kernel,kernel), 0)
val,s_c = cv2.threshold(s,thresh,255,cv2.THRESH_BINARY)
circ_no = 0
circles = cv2.HoughCircles(r_c,cv2.HOUGH_GRADIENT,1,dist,param1=par1,
                           param2=par2,minRadius=90,maxRadius=230)
```

4.7 Finding optimal parameter values

lorem ipsum

4.8 Overfitting

lorem ipsum

5 Contaminated product recognition

5.1 General idea

lorem ipsum

5.2 Thresholding

Thresholding is a simple case of image segmentation.
It can be used to convert one channel (in most cases grayscale) image to binary image.
The basic binary thresholding mode sets every pixel of destination matrix to:

- 0 if the source matrix value is smaller than threshold (integer parameter)
- 1 if the source matrix value is bigger than threshold.

The truncate version sets every pixel of destination matrix to

- threshold if the source matrix value is smaller than threshold, or
- leaves it unaltered otherwise.

Various modes and modifications are used depending on coding schema and colors distribution.

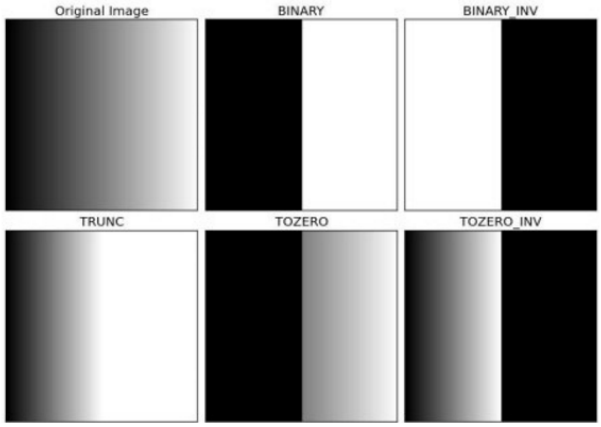

Figure 5.2.1: Different Thresholding modes generated with Metaplotlib

Finding optimal threshold value finding is a time consuming process and it often gives satisfactory results only for small set of images.
Therefore Otsu Binarization algorithm for automatically selecting threshold is used.

Moreover using one threshold value for whole image may not work properly if the lighting conditions differ in different image areas.[18]
Adaptive Thresholding utilize different threshold values for different ROIs.
Two modes are often used: Adaptive Mean and Adaptive Gaussian.
First compute threshold value as mean of neighborhood values.
Second uses weighted sum neighborhood values.

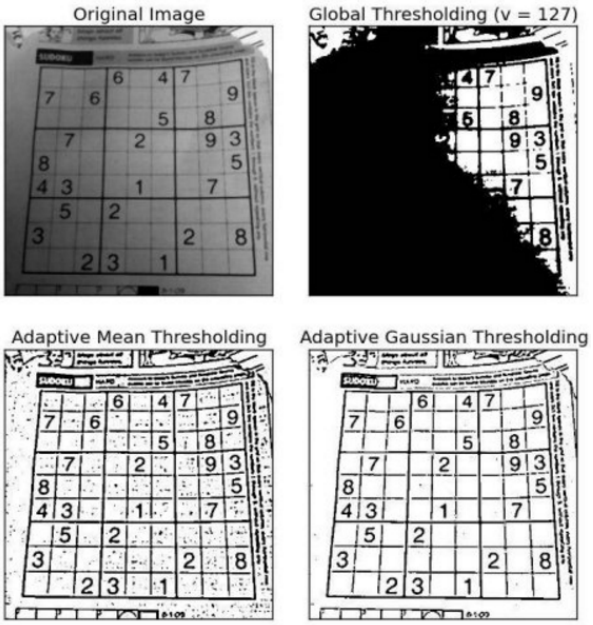

Figure 5.2.2: Threshold computing methods12[18]

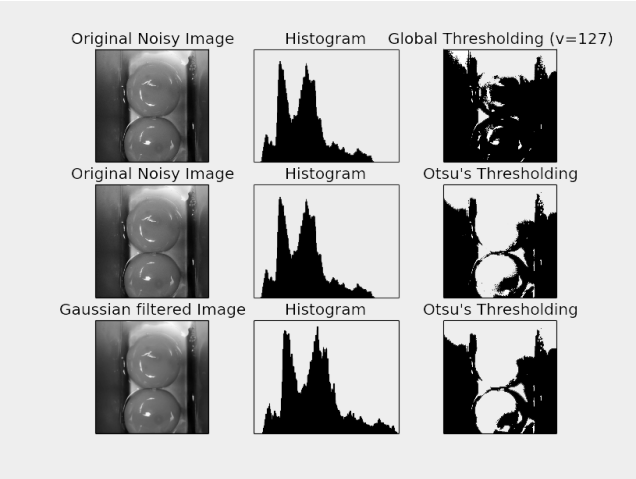

Figure 5.2.3: Different thresholding applied Fig. generated with Metaplotlib

For more general cases (i.e. multichannel images, assigning more colors) clustering method known as k-means is widely used.[19]

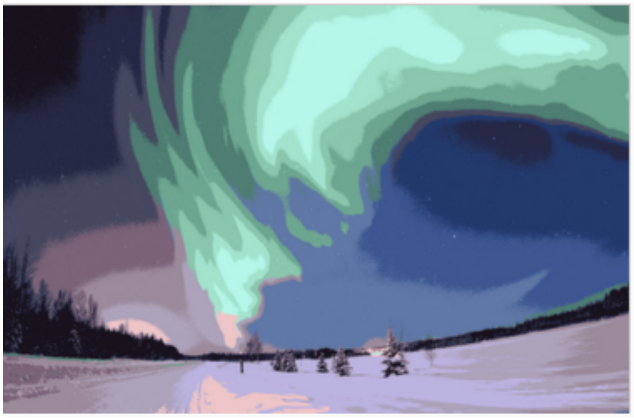

Figure 5.2.4: K-means clustering method for k = 16[20]

5.3 Using the methods

```
s_c = cv2.morphologyEx(s_c, cv2.MORPH_OPEN, cv2.getStructuringElement(cv2
```

6 Choosing detection system parameters

6.1 Manual parameters finding

6.2 Brute force testing

lorem ipsum

7 Phisical system implementation

7.1 Raspberry Pi

lorem ipsum

8 Further research and development

Work on the solution developed in this thesis will be continued further on by the author and OVO-TECH company.
The following alternative approaches and improvements in detection system will be taken into consideration

8.1 Operating Systems

Using alternative system operating on Rasperry Pi 2 platform might improve image processing performance. Therefore, more frames per second could be analized, more filters can be applied and higher resolution pictures could be used. Among over 20 avaliable ARM systems, presented below are most promising:

- Minibian - a Raspbian image that is resource optimised.
  Its current Raspian "Jessie" based version boots in 14secs, use only 29MB RAM and takse 451MB (compared to 1,2GB in original version).
  GUI, aplication clients, office suite and other non-necessary packages have been removed.
  The image is targeted for embedded or server applications (NAS, Web server, electronic applications).

- Arch Linux ARM

- Gentoo

- SliTaz

- Windows 10 IoT Core - embedded solution developed by Microsoft. Guarantees extensive technical support.

Largest reliability and performance benefits are expected from using realtime-capable systems.
Following solutions are avaliable or developed for Raspberry Pi:

- RISC OS

- Xenomai

- PREEMPT_RT kernel patch and High Resolution Timers[21]

- ChibiOS/RT[22]

- RODOS - an Open Source kernel project developed by the German Aerospace Center and Prof. Montenegro's University[23]

Nevertheless advanced image processing, GPU access, utulisation of existing graphics libraries might be not avaliablie or require amount of effort unproportional to those benefits.
Also, external real-time clock (such as AD9850 Pulse generator) might be required.

8.2 Platforms

- Arduino Uno R3 accompanied with DS1307 Real Time Clock - considered, but for now the image processing is to extensive in terms of image processing.
  Having extremly cheap clones (4-20$ on retail market) that fetures often stronger ATMega units is an advantage altough.

- Intel Galileo Gen 2 - based on Intel Quark 32-bit SoC X1000 processor, provides pins-compatible with shields designed for the Arduino Uno R3, what makes communication with other OVO-TECH machine modules easier and allows use of the mentioned above RTC.

- PandaBoard - the OMAP4430 features 1 GHz dual-core ARM Cortex-A9 MPCore CPU (a big improvement compared to Cortex-A7 CPU in Raspberry Pi 2), 304 MHz GPU, and 1 GiB DDR2 SDRAM.
  What is interesting, it provides a real-time clock, what makes it a candidate for real-time system (see section 8.1).

- BeagleBoard xM - based on dual, 1,5GHz clocked ARM Cortex-A15 + dual ARM M4, 532 MHz GPu, often used with RISC-OS, having cheaper clones with more RAM.
  It is considered a stronger Raspberry Pi alternative.

- Nvidia Jetson TK1 - build on quad ARM Cortex-A15, NVIDIA Kepler 192-cores CUDA GPU, 2 GiB RAM, optimised for OpenGL and OpenCV, equiped with SATA connector.
  TK1 is the stronggest avaliable unit, what makes it also the most expensive one.

8.3 Server and logging

Gathering the data about egg quality, recognition performance a d effectiveness is a possibility that has to be addressed.
Raspberry Pi 2 platform is capable of working as a server and thus provide system manufacturer a remote access to the its operations.

8.4 Machine learning

9 Conclusions

All the Web. sources have been checked for availability in November 2015.
Web links are not included if articles are to be immediately found with search engines, which is consistent to MLN referencing convention.
For modified web documents, date of last modification is shown.

References

[1] Berry D. *Egg product functional properties* American Egg Board Web. 2013

[2] Mehdizadeha S., Minaeib S., Hancockc N. H. ,Torshizid M. *Information Processing in Agriculture*, Volume 1, Issue 2 Print. 2014

[3] Deng X., Wang Q., Chen H., Xie H. *Eggshell crack detection using a wavelet-based support vector machine* Comput Electron Agric, Print. 2010

[4] Ketelaerea B.,Bamelisa F., Kempsa1 B., Decuyperea1 E., Baerdemaekera J. *Non-destructive measurements of the egg quality* World's Poultry Science Journal, Volume 60 / Issue 03, Cambridge University Press, Print. 2004

[5] Bradski G. ,Kaehler A. *Learning OpenCV: Computer Vision with the OpenCV Library* O'Reilly Media Inc., Print 2009

[6] Gorelick M., Ozsvald I. *High Performance Python: Practical Performant Programming for Humans* O'Reilly Media Inc., Print 2014

[7] Opencv Dev Team *Hough Circle Transform* OpenCV 2.4.12.0 Documentation, Web. 2014

[8] Milbourne A. *Computers Counting Craters* Birkbeck College, Web. 2012

[9] Rhody, Harvey *Hough Circle Transform* Chester F. Carlson Center for Imaging Science Rochester Institute of Technology, Web. 2005

[10] Kapur S., Thakkar N. *Mastering OpenCV Android Application Programming* Pack Publishing, Print. 2015

[11] Opencv Dev Team *Feature Detection* OpenCV 2.4.12.0 Documentation, Web. 2014

[12] Opencv Dev Team *Morphology Fundamentals: Dilation and Erosion* MathWorks Documentation, The MathWorks, Inc. Web. 2014

[13] Opencv Dev Team *Eroding and Dilating* OpenCV 2.4.12.0 Documentation, Opencv Dev Team, Web. 2014

[14] Shapiro L. G., Stockman G. C *Computer Vision* Prentice Hall, Print. 2001

[15] Khvedchenya E. *How to detect circles in noisy images* Computer Vision Talks, Web. 2014

[16] Nixon M. S., Aguado A. S. *Feature Extraction and Image Processing* Academic Press, Print. 2008

[17] Lecarme O., Delvare K. *The Book of GIMP: A Complete Guide to Nearly Everything* No Starch Press, Print. 2013

[18] Opencv Dev Team *Image Thresholding.* OpenCV 2.4.12.0 Documentation, Web. 2014

[19] Barghout L., Sheynin J. *Real-world scene perception and perceptual organization: Lessons from Computer Vision.* Journal of Vision, Print. 2013

[20] Barghout L., Sheynin J. *Image segmentation* Wikipedia, Wikimedia Foundation Inc, Web. 2015

[21] Gupta V. *Patchwork Gaurantee spinlocks implicit barrier for PREEMPT_COUNT* The Linux Kernel Archives, Web. 2013

[22] Bate S. *ChibiOS/RT on the Raspberry Pi* ChibiOS EmbeddedWare Web. 2015

[23] Montenegro S., Dannemann F. *Real Time Kernel Design for Dependability* DASIA 2009 DAta Systems In Aerospace, Paper. 2009