Level measurements with computer vision - comparison of traditional and machine learning computer vision methods

Eirik Døble, Sindre Haugseter, Christian Mikkelsen, Jørgen Bang Sneisen, Nils-Olav Skeie, Ole Magnus Brastein*

Department of Electrical Engineering, Information Technology and Cybernetics
University of South-Eastern Norway, N-3918 Porsgrunn,
*(ole.m.brastein@usn.no)

Abstract

In this work, modern machine learning methods are compared against traditional image processing techniques, for the purpose of estimating the level of coffee beans in a transparent tank fitted to a coffee machine. Measurements using both approaches are compared against manual level measurements. The resulting algorithm are analysed for repeatability under scene variations, such as orientation of the tank with respect to the camera and the distribution of coffee beans. *Keywords: Level measurement, computer vision, image segmentation, ResNet34*

1 Introduction

1.1 Background

Level measurements is important for a large number of applications in both industry, science and the commercial sector (Bentley, 2005). Popular measurement technologies include guided radar, ultrasonic, capacitance and flotation based sensors principles (Bentley, 2005). In many cases, it is desirable to apply a non-intrusive sensor principle. One possible solution which has received significant scientific interest in recent years is the use of digital cameras together with advanced, typically machine learning (ML) based, algorithms (Goodfellow et al., 2016). This technology has a large range of possible applications, including the non-intrusive level measurement of substances in a partially transparent tank.

In this work, the system of interest is a coffee machine that has been outfitted with an industrial robotic arm. The goal of the project, originating from Bouvet Consulting in Porsgrunn, is to create an *AI barista*. One aspect of this project is to measure the level of coffee beans remaining in the tank of the coffee-machine. Since the machine is a common-of-the-shelf model, there is no level measurement sensor built in. However, as is typically the case for such devices, there is a transparent inspection window which allows users to visually estimate the level of coffee beans in the tank. The facility for visual inspection, together with the obvious need to ensure safe human consumption of the produced coffee, makes the use of vision based sensor technology particularly interesting.

1.2 Previous work

There have been several published works on using computer vision for level estimation for liquids, which is arguably easaier then the granular coffee beans studied in this work. Zepel et.al. used open-source libraries (Numpy, OpenCV, and PySerial) for liquid level monitoring and control in common continuously stirred tank reactor processes. They used Canny Edge detection in order to locate strong horizontal edges to detect the liquid-air interface, and perform decisions to control the pump(s) for manipulating the liquid level. They found that the method gives acceptable results when using computer-vision as part of an autonomous platform to monitor experimental factors and make control decisions. (Zepel et al., 2020).

In (Eppel and Kachman, 2014) a general computer vision method for the recognition of liquid surfaces and liquid levels in various transparent containers is presented. They concluded that making a general recognition method for liquid systems is possible and that it can be achieved with good accuracy in various cases. The best indication of the liquid surface was found to be the relative intensity change, the edge density change and the gradient direction relative to the curve normal.

2 Methods

2.1 Machine learning vs traditional computer vision algorithms

The field of computer vision has experienced a paradigm shift over the last two decades (Goodfellow et al., 2016). The application of machine learning (ML), and in particular multilayer artificial neural networks (ANN), known as *deep learning*, has revolutionised what is possible to achieve with computer vision (Goodfellow et al., 2016), leading to the use of computer vision in many new application. While machine learning certainly has had a profound effect on the computer vision field, there are still much use for non-learning algorithms as well. In many applications, the traditional image processing methods may indeed be advantageous because they tend to be faster w.r.t. execution time.

Before presenting the methods of interest in this work it is instructive to discuss the fundamental difference

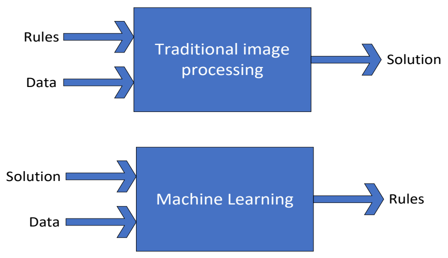


Figure 1. Comparing traditional image processing with machine learning

between the two approaches to image analysis. Before the advent of ML, image analysis was conducted by constructing algorithms using various developed standard techniques, such as filtering and gradient computations (Bradski and Kaehler, 2008). Most analysis tools uses a series of such steps to compute a result. The common denominator for these techniques is that the algorithms consist of a set of rules that describe how an image should be processed to produce an analysis results or solution (Bradski and Kaehler, 2008). In some applications, engineering these rules turns out to be very difficult. Consider the textbook case of distinguishing images of cats and dogs, it is difficult to imagine manually constructing a set of rules that can compute the probability of an image containing a particular animal based solely on the pixel values.

As shown in Fig. 1, machine learning turns the approach to image analysis, and indeed data analysis in general, around. Rather then *engineering* the rules needed to produce a solution, relevant *input data* is coupled by a *desired solution* (Goodfellow et al., 2016; Kuhn and Johnson, 2013). Subsequently, algorithms are used to identify *patterns* between input data and the desired solution. The identified patterns can be considered as *machine generated rules* which in turn is applied to new images in order to compute the desired result (Goodfellow et al., 2016; Kuhn and Johnson, 2013). Naturally, rules generated by a computer by pattern recognition is not necessarily similar in formulation to the rules or instruction steps used in traditional computer vision algorithms, but their use is the same; computing a solution or analysis result for new images of the underlying system of interest.

2.2 Machine learning using fastai

The ML framework of choice in this project is *Fastai* which was created by the Fast.ai organization (Howard and Gugger, 2020). The motivation for developing the framework is to provide a practical approach to machine learning, where the idea is to start learning by doing practical work instead of first requiring a deeper theoretical knowledge of the intricacies of ML. The fastai framework is based on PyTorch (Paszke et al., 2019) and gives the PyTorch library an extra layer of functionality using the API's by offering high-level API's which makes it easier to get started with machine learning.

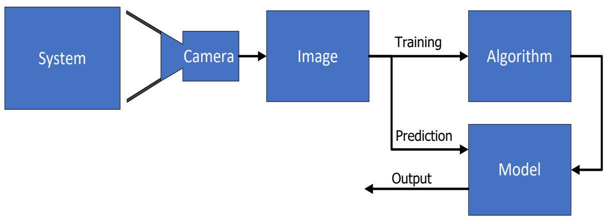


Figure 2. Machine learning and prediction with images

2.2.1 Image classification and segmentation

A fundamental goal of many computer vision analysis algorithms is to *label* the content of an image (Goodfellow et al., 2016). Two distinctly different approaches towards applying labels to various parts of an image are known as *classification* and *segmentation* (Bradski and Kaehler, 2008). Both methods use an algorithm to train a *model* from the images. After a model has been trained, new images is sent to the model and an output is given, as shown in Fig. 2.

In *classification* objects in the image are analysed and labelled as belonging to one of many pre-determined classes, often together with an estimated probability accuracy of the class label being correct (Bradski and Kaehler, 2008; Goodfellow et al., 2016). A class can be, for example, a car, a cat, a house, etc. When training a classification model, a large number of images are pre-labelled and sorted according to the class of object they contain, typically arranged in different folder in the filesystem. The output of an image classifier is a region, typically rectangular, in the input image together with a label that classifies the content of that region.

In contrast, image *segmentation* does not detect discrete objects, but rather seeks to determine non-uniform regions in the image which belong to a particular pre-trained class (Bradski and Kaehler, 2008). As such, segmentation can arguably be considered a type of classification, but instead of classifying the object in the image, *each single pixe*l is labelled depending on what class it most likely belongs to. When training a segmentation model, each pixel in the training image is labelled by a class id, typically using different shade of grey in an overlaying image known as a *mask* (He et al., 2016; Deng et al., 2009). By coupling a mask with a training image, the machine learning algorithm learns the pattern that connects image content with segment class. When the trained segmentation model is applied to a new image it will assign a class label to every pixel, such that neighbouring pixels of the same class form an image segment. The segmented image can then be further analysed to locate objects and boundaries like lines and curves in an image.

2.2.2 Estimating tank level from an image

In this work, image segmentation is used to estimate the level of coffee beans in a transparent tank. After applying the segmentation model on a new image, the method outputs a tensor of same dimensions as the input image, where each tensor element constitutes a class label for the pixel in the corresponding image.

All the training images used in this project are tagged by manually creating masks with three classes: COFFEE, EMPTY_TANK and BACKGROUND. When a new image is segmented using the resulting trained model, the output is a tensor array of the same dimensions as the input image, containing the identified classification of each pixel.This tensor is used to calculate the level in the coffee bean container using:

$$Level\,[\%] = \frac{N_{\text{coffee}}}{N_{\text{coffee}} + N_{\text{tank}}} \tag{1}$$

where N_{coffee} and N_{tank} is the number of pixels after image segmentation classified as belonging to the segments COFFEE and EMPTY_TANK, respectively. The underlying assumption of this approach is that the camera is positioned such that the 2D projection of the coffee tank onto the image sensor produces an image, where the region consisting of coffee beans relative to that of the complete tank, e.g., the empty tank plus the coffee beans, is proportional to the coffee volume of interest. While this assumption is true for the approximate rectangular tank used for this work, a more complex geometry may require a second regression model to be trained in order to estimate the actual volume from the projected 2D regions in an image (Goodfellow et al., 2016; Kuhn and Johnson, 2013).

2.2.3 Transfer learning

Transfer learning (TL) was first introduced in (Bozinovski and Fulgosi, 1976) where they describe a mathematical and geometrical model of TL (Bozinovski, 2020). TL is a method in machine learning where knowledge gained in one task is exploited to improve generalization in different but related task (Goodfellow et al., 2016). For example, knowledge gained from recognizing a car could be used when learning to recognize a truck. TL is a popular approach in computer vision and natural language processing tasks because it can train neural networks with comparatively little data with shorter training time than when training from scratch. In most real-world problems it is difficult to obtain a large number of labelled data points for training of complex models. Hence, a pre-trained model is beneficial. In Computer vision, it is common for the neural network to first find edges in the first layers, general shapes in the middle layers, and finally task-specific characteristics for the last layers. In TL, the first and middle layers are transferred to the new model, while only the last layers must be re-trained. The main advantages of TL is shorter training time, less training data, and in most cases better performance (Bozinovski, 2020).

2.2.4 ResNet

The ML model of choice in this work is a Residual Neural Network (ResNet), which is a continuation of a convolutional neural network (CNN) that has become a popular model for computer vision in the recent years (He et al., 2016). To further improve on the CNN method, the ResNet adds skip connection or shortcuts to jump over some layers. ResNet models typically implements double- or triple- layer skip connections. A weight matrix can be used to find the skip weights in models known as HighwayNets. The skip connections help to avoids the troublesome vanishing gradient problem, which occurs in CNN which preventing further training.

The ResNet model used for this project is ResNet-34 which consist of a network with 34 layers (He et al., 2016). The ResNet-34 model was pre-trained using the ImageNet dataset which consists of millions of pre-labeled images (Deng et al., 2009). Larger networks are better at complex problems but are easy to overfit, takes longer to train and use more memory than smaller networks. Hance, ResNet-34 was chosen because its performance to accuracy trade-off was satisfactory for the problem. Other neural networks like AlexNet, GoogLeNet, DenseNet and SqueezeNet was tested but based on initial experimentation the ResNet-34 model appears to perform adequately for the task.

2.2.5 Model training

The ResNet-34 model was repeatedly re-trained with an increasing number of training images, until a satisfactory result was reached at only 93 images. The relatively low number of training images needed to achieve satisfactory performance shows the strength of using transfer learning on a pre-trained model.

A common method for increasing the variation in the training set artificially is to apply randomised transformations to training set. Examples of transformations are: changing the image size, flipping, rotating, or adding Gaussian blur. By increasing the variation of images in the training set the resulting model will be able to handle a greater variance in images used as inputs when predicting results.

2.3 Traditional approach using OpenCV

The traditional approach to computer vision, i.e., prior to the advent of ML methods, is to construct the algorithms using a sequence of processing steps, typically using a selection of standard computations such as computing gradients, filtering, global or local thresholding, and morphological transformations (Bradski and Kaehler, 2008). These operators are applied to produce new images, typically of the same dimensions in height and width but not necessary the same bit-depth. Input images are typically RGB encoded bitmaps of bit-depth 8, but many other formats exist. Often, building the software for image capture and conversion into the required format requires considerable work (Bradski and Kaehler, 2008). A sequence of capturing and processing steps is often denominated as a *pipeline* in the computer vision field.

One of the main advantages of traditional computer vision algorithms is that the *image operators* , i.e., mathematical processing steps, that is used to build up higher level algorithms are well known and understood (Bradski and Kaehler, 2008). Hence, there exist highly optimised implementations of these steps. Arguably, the most popular library of such optimised computation methods is the *Open-Source Computer Vision Library (OpenCV)* (Bradski and Kaehler, 2008). As an open-source library, the framework consist of contributions from many of the worlds leading experts in computer vision. Since OpenCV is widely used, its methods and syntax is commonly understood and recognised by software engineers in the field, which helps to speed up the development of new applications (Bradski and Kaehler, 2008). OpenCV supports the programming languages C ++, Python, Java and Matlab, and all major operating systems, e.g., Windows, Linux, Mac OS, and Android.

In this work, OpenCV is used to analyse the images of the coffee bean tank and subsequently provide an estimate of the level of coffee beans. Two promising methods of interest towards this goal is the use of segmentation by *binary thresholding* or edge detection using the *Canny Edge Detection* algorithm. Note that OpenCV also contains several ML based approaches, but in this work it is the non-learning methods in OpenCV that are of primary interest.

2.3.1 Binary threshold

OpenCV supports numerous thresholding functions (Bradski and Kaehler, 2008), including a simple *binary threshold* with or without inversion, i.e., pixels above threshold can be defined as either high (white) or low (black) in the output image.

The simplest thresholding method compares every pixel in the frame against the pre-determined threshold l_{coffee}. In the output image, each pixel value is set depending on being above or below the threshold. If the intensity of the pixel $f_{x,y}$ is greater than that given threshold value l_{coffee}, the output image pixel $\tilde{f}_{x,y}$ will be set to black, e.g., a greyscale value 0 and white or greyscale 255 otherwise, according to:

$$\tilde{f}_{x,y} = \begin{cases} 0, & f_{x,y} > l_{\text{coffee}} \\ 255, & otherwise \end{cases} \tag{2}$$

The application of a threshold assumes that the region of interest can be distinguished from the background by the intensity of the pixel alone. One difficulty with applying this method is choosing an appropriate threshold value, since this value is highly dependant on the scene and lighting conditions.

While the application of the threshold computation is straight forward, building an algorithm around thresholding requires the application of several additional image operators to pre-process the image. In this work, the image is first converted to greyscale by averaging the RGB values of each pixel. Next, a Gaussian blur step is used to smooth the image and reduce the influence of noise on the thresholded image, before finally applying the threshold step. Further, as illustrated in Fig. 3, the

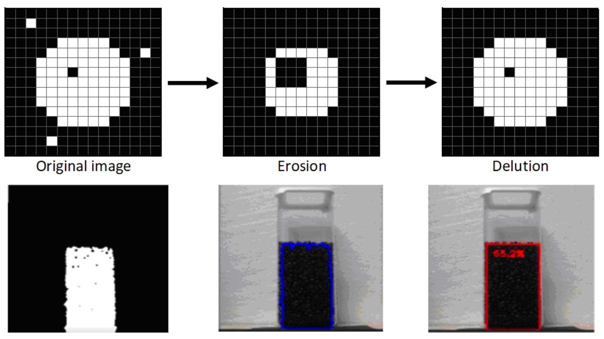


Figure 3. Binary threshold post processing steps.

output of the threshold step is post-processed by the morphological transformations *erosion* and *dilation*, which together forms the operation known as *opening*, to further reduce image noise. The output of the morphological transformation is a binary image where, assuming appropriate threshold value, the coffee bean area is marked as white. The last two steps of the algorithm is to find the contour of the thresholded region and finally fit a rectangle around it. Since the width of the tank is known apriori, the coffee bean level can be estimated from the height vs width ratio of this identified rectangle. A weakens of this approach is that it neglects the possibility of uneven distribution of coffee beans. However, this shortcoming could easily be remedied in future work by either fitting a polygon or multiple rectangle slices across the width direction.

2.3.2 Canny edge detection

Canny Edge Detection (CED) is an algorithm that can be used to detect edges in an image (Bradski and Kaehler, 2008). CED is a multi-step algorithm which consist of the following steps:

1. Noise reduction with a Gaussian filter
2. Finding the intensity gradient compares a pixel's value against its neighbouring pixels and outputs a new image where the larger difference in intensity equates to a higher pixel value
3. Non-maximum Suppression is used to thin the edges
4. Hysteresis thresholding is used to connect neighbouring pixels into a consistent edge

The CED algorithm was tested for the purpose of identifying the edges of the tank and the edge between the two mediums: coffee beans and air. However, the binary threshold algorithm was found to provide more consistent results, hence in the sequel only the binary threshold method is discussed further .

3 Experimental setup

An experimental rig, consist of a wooden plate where a Logitech C922 camera and the coffee tank are mounted, was constructed in order to get repeatability in the pictures w.r.t. camera angle. The camera angle was chosen such that the flat top of the coffee container would not disturb

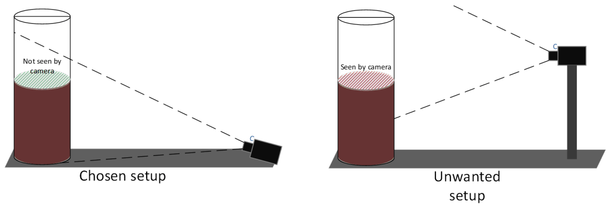


Figure 4. Experimental setup

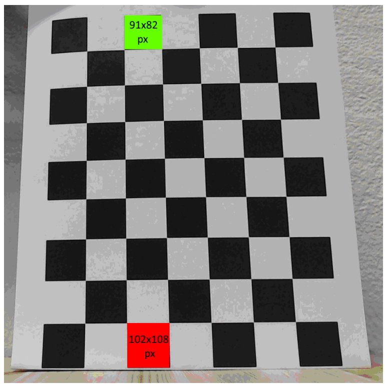


Figure 5. Image shows a A4 25x25mm calibration chessrboard.

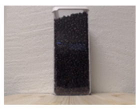
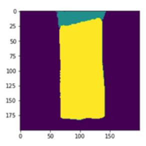
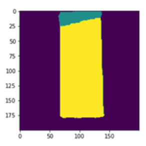
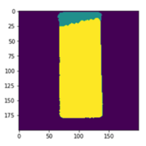
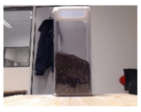
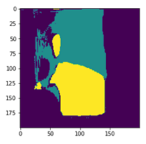
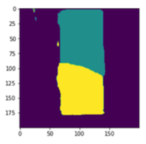
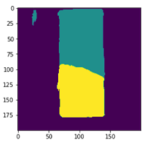
Image	20 epochs	40 epochs	100 epochs
			
			

Figure 6. Training with increasing number of epochs. Segments are labeled as coffee (yellow), empty tank (cyan), and background (purple).

the measurement, as shown in Fig. 4. This is because of the method used for measuring level is effectively measuring the area that is in the cameras field of view (FOV).

To get enough variety in the background of the pictures the rig was positioned with different walls in the background, and some black paper was used to show that black does not automatically mean 'coffee' for the ML model. The ideal scenarios are a plain light-coloured background with little disturbances, even lighting, and little to no reflection.

3.1 Perspective distortion

Due to the position of the optical sensor in relation to the tank, as shown in Fig. 4, the captured images are somewhat affected by perspective distortion, which will influence the measured area. As shown in Fig. 5 a rectangle of a given size will occupy a bigger pixel area the further down in the image that it is projected. Observe also that, due to a slight lens distortion effect, the rectangles are not completely square. Compensating for these factors would be recommended for future work and is achievable by standard OpenCV methods.

4 Results and discussion

4.1 Model training

Once the fasai framework is configured and the training images loaded, the model training process consist of repeatedly calling the *fit_one_cycle* method, which advances the model training one iteration, or epoch as is the denomination used in the ML literature. The method returns the loss and accuracy metric in the form of a table for the current training stage. Since segmentation is used to estimate the coffee level, the loss function compares pixel gradients in the training pictures and masks, as discussed in Sec. 2.2.2. To determine the ML model's performance loss, the predicted output is compared with the target value, and the deviation between these determines the loss values, where a large deviation gives a high loss value.

The amount of training needed varies from the complexity of the task. Some examples of different situations are shown in Fig. 6. In the first row, an ideal scene configuration with even lighting, little reflection and no noise in the background is shown. Here the machine learning model finds satisfactory results already after just 20 epochs. A more challenging scene is shown on the second row, where a dark jacket is introduced in the background. With only 20 epochs of training, the ML model struggles to differentiate the dark jacket from the container with coffee. At 40 epochs the prediction is much better, and at 100 epochs of training the prediction is close to perfect. This result shows that more complex scene configurations are more challenging to segment and therefore require a more intensively trained ML model. Further, this results shows that it is important to have a variety of scene configurations in the training data since the ML model can only be expected to accurately estimate the image segmentation of images that are similar to the training set used to build the model.

4.2 Optimal scene conditions

The ideal scene configuration is a plain light-colored background with little disturbances providing good contrast with the coffee beans, even lighting, and little or now reflection from the transparent tank. In this situation both OpenCV and fastai detects the area with coffee beans with acceptable accuracy. The mean error from ten different predictions gives fastai an error at 0,7% and OpenCV 3,0% as the results in Fig. 7 and Table 1show. The estimated level accuracy for both methods is considered well within acceptable range for the purpose of a coffee robot.

Note that both methods failed to estimate the level to 0% if the tank is empty, due to how the level is computed

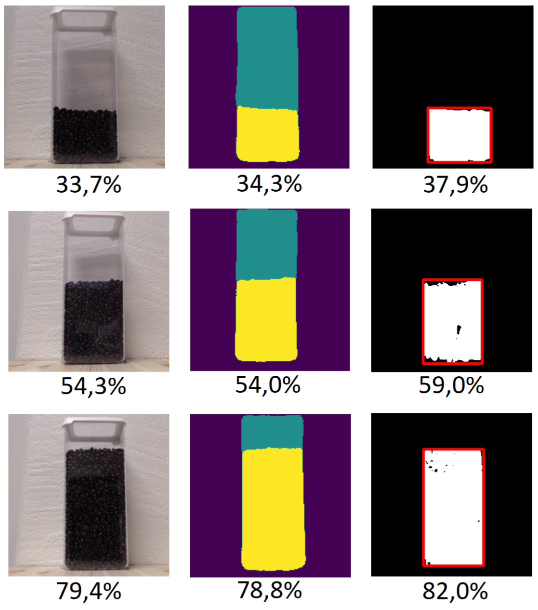


Figure 7. Results of applying both methods under optimal conditions (left: raw image, middle: ML result, right: threshold result).

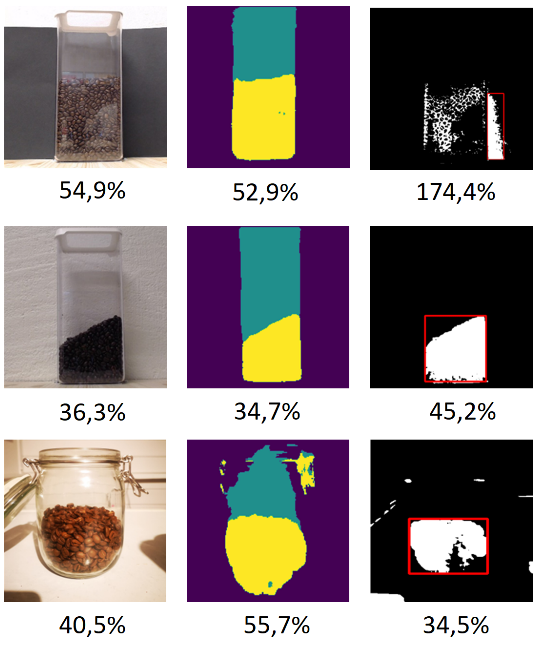


Figure 8. Results of both methods under challenging conditions (left: raw image, middle: ML result, right: threshold result).

from the image analysis results.

4.3 Challenging scene conditions

Figure 8 shows some examples of more challenging scenarios. In the first row a black sheet of paper was added to the background to test if the models could differentiate dark background from the coffee beans. The results show that fastai still segments the coffee accurately, except for a small strip with a lot of reflection on top of the coffee beans. In contrast, the binary threshold method failed completely due to the pre-determined threshold value no longer being suitable for the captured image. The dark background caused the camera to automatically turn up the light sensitivity, which made the coffee too light to be within the threshold. For future work all automatic functions on the camera, like light sensitivity, should be turned off to secure better control over the images.

The second row shows a near ideal scenario, but with the coffee slanted. Here, Fastai also obtains an accurate level estimate while the threshold based method significantly overestimate the level. Since the ML method post processing counts all the marked pixels, while the threshold based method draws a rectangle around the extreme points in the largest contour and find the height from a width to height ratio, the latter is unable to account for an uneven coffee distribution and will therefore always overestimate the level in such conditions. A solution for future work would be to draw a polygon around the coffee instead of a rectangle.

The images in the third row of Fig. 8 contains a completely different container than used in in the training data. One of the problems with this scene is that the top surface of the coffee is visible, which violates the assumption discussed in Sec. 4. The coffee beans in the picture are also a much lighter colour which further complicates the choice of threshold value. The fastai model, despite the rather large deviation of this particular scene configuration compared with the training images, successfully segmented out the coffee beans. Only some areas around the lid were misclassified. The estimation error of 15% is largely caused by the image violating the assumption of camera angle w.r.t. the coffee surface in the jar being visible. While the threshold method apparently produced a prediction error of only 5%, this result is simply random and must be rejected as false. Since the ratio between the height and the width of the container is different from the assumed tank geometry, the post processing of threshold based method cannot produce reliable results without adapting the assumptions to the new container.

A common challenge encountered in many of the experiments performed in this work is the sensitivity to reflection on the container. The reflections affect the results

Table 1. Comparing analysis results under ideal conditions.

Ref [%]	ML [%]	Err [%]	OCV [%]	Err [%]
15.2	14.7	0.5	14.6	0.6
25.2	26.4	1.2	26.7	1.5
33.7	34.3	0.6	37.9	4.2
44.3	44.3	0.0	45.8	1.5
54.3	54.0	0.3	59.0	4.7
62.1	62.1	0.0	65.1	3.0
72.0	71.0	1.0	74.3	2.3
79.4	78.8	0.6	82.0	2.6
97.2	96.3	0.9	102.3	5.1
Mean error		0.7		3.0

on both fastai and the threshold method, resulting in inaccurate level calculations. In fastai the reflections will in some cases be interpreted as the container instead of the coffee beans behind the reflection. In the threshold method the reflection brightness makes the algorithm neglect part of the coffee area as it is brighter than the coffee beans not covered by a reflection.

4.3.1 Adapting to changes in the image scene

As discussed in Sec. 4.3, performing level estimation on an image which deviates from the reference images can cause the level prediction to fail or miscalculate the level. These disturbances and deviations are typically changes in the container geometry, challenging light conditions, colour schemes, reflections, shadows, or new structures in the background.

The fastai model is only capable of accurate segmentation on images that are similar to the training images. However, it can be retrained for new deviations or disturbances, which is done by adding images containing the new conditions to the training set. Next the model is trained through several epochs to prodiuce a new model. The existing training data should be preserved and is used as a starting point when a model is trained with new conditions. This process of adding new data to the training set and subsequently re-training the model is arguably similar to process of creating a new model using transfer learning on the general ResNet-34 model.

In the threshold based method, the expected scene is more closely tied to the chosen method parameters, i.e., the tank geometry and threshold values, which must be set for each specific scenario. Since the threshold value is more highly dependant on light conditions and colour, it is necessary to manually set new threshold values by trial and error. An alternative for future work could be to detect the threshold automaticity by further utilising apriori knowledge of the scene, such as the expected approximately rectangular shape of the coffee area, or indeed the expected location of the coffee beans in the projected image.

The difference in manual labour needed when updating the methods for new conditions or deviations, shows the versatility and adaptability of the fastai training methods compared to the more traditional threshold based method. When compensating for disturbances fastai will retain all previous scenarios, while the threshold method is configured for using specific settings for each scenario.

4.4 Repeatability under experimental variation

To test the *repeatability* of the computer vision based level estimation methods, two experiments were preformed. First, the tank is rotated such that the images are captured from 10 different angles in the range 0 to 90 deg. Next, the beans were repeatedly removed and replaced in the tank to test different distributions of coffee beans. Both experiments used the exact same amount of coffee beans, and consist of 10 repeated images captured with no more than one changing experimental variable for each experiment.

Table 2. Repeatability under altered viewing angle.

	OpenCV [%]	Fastai [%]
Average	59.6	56.4
Std.dev.	4.59	2.04
Max.dev	12.1	3.01

Table 3. Repeatability under altered coffee distribution.

	OpenCV [%]	Fastai [%]
Average	59.6	58.03
Std.dev.	3.39	2.39
Max.dev	8.44	5.67

4.4.1 Rotating tank - altered viewing angle

The results of the first experiment, shown in Table 2, shows that the standard deviation of the thresholding based method is more than twice that of the ML based method. This can be explained by how the threshold method post process the binary output image under the assumption of a known tank geometry height/width ratio. When viewing the tank at an angle, the assumed tank geometry differs significantly from the observed image projections, hence the estimated level is incorrect. In contrast, the ML method segments the tank directly, thus capturing the effective projected width of the tank from any angle, thereby producing accurate estimates level estimates also under varying viewing angles.

4.4.2 Refilling tank - altering distribution of coffee beans in tank

The results of the second repeatability experiment, shown in Table 3, shows that both methods are somewhat robust against the distribution of coffee in the tank. As in the previous experiment, the ML method shows lower estimation errors, but only marginally so for the second experiment. The error in the threshold method is mostly driven by the assumption of a rectangular coffee region, i.e., the use of a fitted rectangle around the detected thresholded region. If an alternative geometry is fitted, e.g., a polygon, the threshold method would likely have similar robustness to coffee distribution as is found for the ML method. Again, the ML method, by use of image segmentation to obtain a detailed shape of the coffee in the tank, produces accurate estimates also under varying distributions of coffee within the tank.

4.5 Timing

An important consideration in any method that utilities ML is the computation time needed to obtain a results. The largest allowable computation time is often denominated as a *hard real-time requirement*. In this work, the real-time requirement is the minimum amount of time it

Table 4. Comparing computation time of both methods.

	OpenCV [s]	Fastai [s]	Relative [x]
Average	0.019	6.35	337
Fastest	0.013	2.75	208
Slowest	0.030	11.7	388

would take to brew a cup of coffee, since the coffee level only needs to be calculated when the level changes.

To verify that the proposed solution meets this requirement, an experiment was done where 40 level predictions was executed with both binary threshold in OpenCV and the re-trained ResNet model in fastai to compare the time usage for each prediction. The experiment used a collection of randomly selected images and gave the results shown in in Table 4. As expected, the threshold method including pre- and post-processing steps vastly outperforms the ML method in terms of computation time, being on average 337 times faster. This can be explained by the relative simplicity of the computations needed in the binary threshold method, compared with the mathematical complexity of evaluating a 34-layer neural net.

However, from manual observation of the AI barista robot, just the task of moving the coffee cup from the machine to the customer is found to take around 12 seconds, hence a maximum computation time of 11.7 seconds for the ML method is more then sufficient w.r.t. the real-.time requirement.

5 Conclusions

The goal of this work was to test the feasibility of using computer vision, both machine learning and more traditional rule-based computations, for non-intrusive coffee bean level estimation in a transparent tank. Further, both approaches was compared on merits of accuracy and computational speed. Both methods are found to be suitable for the specific application in the AI barista project.

Based on the results and analysis presented in this work, it can be concluded that computer vision with machine learning is superior to traditional image processing in terms of accuracy, robustness against scene configuration, and user-friendliness. The traditional methods still produce good level estimation accuracy, but requires significantly more assumption w.r.t. the scene configuration. However, the traditional approach is computationally much faster and therefore less resource demanding. If assuming an ideal scene configuration, e.g., good lighting and without disturbances, the traditional method may be preferable in applications where the real-time requirements are more challenging than for a coffee machine. Over all, the level estimation accuracy and repeatability of both methods are found to be acceptable, with some suggested improvements to improve robustness of the threshold based approach, for the implementation in a coffee machine, but in another application, there may be a higher demand for correcting distortion and the robustness against stacking errors.

In future work, lens and perspective distortion due to camera physics and position should be compensated for. To increase the threshold methods robustness against slanted coffee distributions, the post processing step of fitting a rectangle to the obtained threshold region should be modified to instead fit a more flexible geometric shape, e.g., a polygon or a set of thinner rectangle slices that together make up the full tank width.

References

John P Bentley. *Principles of measurement systems*. Pearson education, 2005.

Stevo Bozinovski. Reminder of the first paper on transfer learning in neural networks, 1976. *Informatica*, 44(3), 2020.

Stevoand Bozinovski and Ante Fulgosi. The influence of pattern similarity and transfer learning upon the training of a base perceptron b2. *Proceedings of Symposium Informatica*, (3-121-5), 1976.

Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. " O'Reilly Media, Inc.", 2008.

J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi:10.1109/CVPR.2009.5206848.

Sagi Eppel and Tal Kachman. Computer vision-based recognition of liquid surfaces and phase boundaries in transparent vessels, with emphasis on chemistry applications. *arXiv preprint arXiv:1404.7174*, 2014.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*, volume 1. MIT press Cambridge, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Jeremy Howard and Sylvain Gugger. Fastai: A layered api for deep learning. *Information*, 11(2):108, Feb 2020. ISSN 2078-2489. doi:10.3390/info11020108.

Max Kuhn and Kjell Johnson. *Applied predictive modeling*, volume 26. Springer, 2013.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*, 2019.

Tara Zepel, Veronica Lai, Lars PE Yunker, and Jason E Hein. Automated liquid-level monitoring and control using computer vision. *ChemRxiv Preprint*, 10, 2020.