FinalProject_MilkDetection

by Peetimon Arunwiriyahkit

Submission date: 10-Dec-2022 11:04PM (UTC+0700)
Submission ID: 1970701968
File name: FinalProject_MilkDetection.pdf (38.36M)
Word count: 8172
Character count: 43912

Mahidol University
Department of Biomedical Engineering
Faculty of Engineering

PROJECT REPORT ON

Milk detection on food label

PRESENTED TO

Assoc. Prof. Dr. Panrasee Ritthipravat

PREPARED BY

Chawanrat	Wisitphongphiboon	6213454	EGBI
Thanyatorn	Leethamchayo	6213459	EGBI
Peetimon	Arunwiriyahkit	6213462	EGBI

EGBI443 - Image Processing in Medicine
1st Semester of 20202 Academic Year

Table of Contents

Introduction

There are a tremendous amount of people that are vulnerable to milk. The symptoms can be divided into two main types which are milk allergy and lactose, protein in milk, intolerance. According to Sudsa-ard et al.(2014), around 50 - 60 percent of adults in Thailand and 65-70% of adults worldwide have a symptom of lactose intolerance as they cannot digest lactose after drinking milk. This symptom usually occurs in adults rather than children because the human body generates less lactase, an enzyme that digests lactose when we grow older. Furthermore, 1.9% of the world's population has a milk allergy and cannot drink milk at all. More than 40% of people worldwide are unable to digest lactose after they reach adulthood.

The aim of this project is to help Thai people who are vulnerable to milk find products that contain milk so they can avoid consuming them accidentally. Our main focus is Thai snacks which have both Thai and English texts on their label.

To develop the algorithm, we utilized many tools to help people recognize certain words from a food label i.e. image processing, OCR and PyQt5. In this study, the term "image processing" refers to digital image processing, which uses computer algorithms to execute operations on images and improve the image quality or extract usable information from it. Image processing techniques come in a wide variety of usage. The field of computer vision known as text recognition from photographs is both implementable and quite practical in real life. In this project, we can create an application to read text from labels on food packaging in both Thai and English by utilizing OpenCV and a text recognition library. This project consists of 3 main parts. First is preprocessing the image of 30 snacks' labels to remove noise and rotate the image to be easily for text recognition to improve the results greatly by using the OpenCV algorithm which is an important part of this project. Next is utilizing a text detector to recognize the image's text sections by using Optical character recognition(OCR), in this case, pytesseract. Last is the User Interface to apply the algorithm into the platform that is easily used by using PyQt5 in Python. The block diagram of the system is shown below.

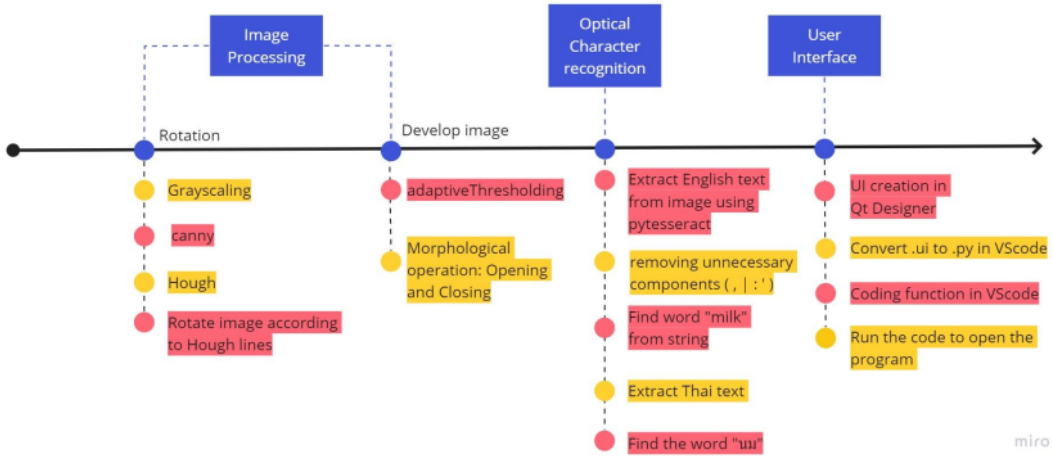


Figure 1: Block Diagram of the overview system

Coding Structure

Image processing

Image processing is the first part of the coding structure. The goal of this step is to develop a raw image taken from the user's camera because, most of the time, a raw image is blurred, tilted, and contains noises. That makes the optical character recognition or OCR algorithm unable to detect text from an image. This step attempts to solve all those problems by reducing as much noise as possible and making the lines of an alphabet connect simultaneously. So, the OCR is able to interpret certain alphabets or words.

Note: This section will explain the image processing step-by-step with only some parts of the full codes. You can see the full version of the codes in the appendix.

Grayscaling

First, the images are imported into the program by the user. We start by turning into grayscale using the OpenCV function,

```
cv2.cvtColor(img, cv2.COLOR_RGB2GRAY).
```

This function transforms the image with 3 channels into 2 channels, making it easier to be processed in the next step.

Figure 2: Comparison between raw image (left) and grayscale image (right)

Rotation

Next, The first issue that needs to be fixed is that the image is tilted making the OCR, in this case: Pytesseract, unable to extract text from the image. We fix this problem by using Hough Transform. Hough transform is a technique that can recognize shapes, i.e. lines and curves, within an image. In this project, we used Hough transform to detect straight lines from text or frames of the snack's label and rotate the image according to those lines. Before using the Hough transform, we applied Canny edge detection (Fig. 3) using

```
canimg = cv.Canny(gray_img, 60, 200)
```

to extract only the edge so the Hough transform is able to easily detect the straight lines. Then, the Hough transform is applied using,

```
lines = cv2.HoughLines(canimg, 1, np.pi/180.0, 250, np.array([]))
```

The detected lines are shown in blue color in the figure below. Next, the image is tilted by using the angles of the detected lines.

Figure 3: Image after applying Canny edge detection (left) and Hough transform (right)

The angles, which are now in the radian unit, are transformed into degrees by using the formula,

```
180*theta/np.pi - 90
```

Finally, we choose the angle that is detected most by using the statistic 'mode',

```
deg_mode = mode(deg)
```

and tilt the image using ndimage from Scipy library,

```
ndimage.rotate(img_nr, mode(deg)).
```

However, the detected lines sometimes are not aligned horizontally making the image tilted in the wrong direction. If the detected angles are more than 20 degrees, we subtract them to make the tilt angle not exceed 20 degrees as shown in fig. 4.

Figure 4: Rotated image from Hough transform

Thresholding

After rotating the image, it still contains shades and some areas are still too bright as you can see in the figure above, this occurs from the lighting when the image was taken. We solve this problem by applying thresholding to a grayscale image. However, simple thresholding is not enough. When applying simple thresholding, in this case,

```
cv2.threshold(thresh, 100, 255, cv2.THRESH_ BINARY
```

the thresholding is unable to simplify too-light and too-dark areas and requires some variable that differs for each image. To overcome that problem, we used adaptive thresholding,

```
cv2.adaptiveThreshold(gray, 255, cv.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.
```

```
THRESH_BINARY, 201, 9
```

instead. When applying adaptive thresholding, the image becomes better and the shady and light areas are eliminated as you can see in the figure below. This is because adaptive thresholding calculates different thresholding values for each small region, unlike the simple threshold method that the same value is applied to all of the images. To add, we use gaussian to be the calculation method since it weights the sum of the region area in the image and delivers satisfying results.

Figure 5: Comparison between simple threshold (left) and adaptive thresholding (right)

Morphological operation

However, the result from adaptive thresholding sometimes adds more noise to images. We dealt with that by using a morphological method, opening and closing. The kernel used for the morphological operation is called the rectangular kernel. The morphological kernel contains only the value of 1. The example of a morphological kernel of 5x5 is structured from the function

```
cv.getStructuring Element (cv.MORPH_RECT,(5,5))
```

and the result is

```
array([[1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1]], dtype=uint8)
```

Next, we applied the kernel to opening and closing. Opening is to apply erosion and, then, dilation. It fills in black points (reducing white spots) making black text easier to interpret. On the other hand, closing applies dilation followed by erosion. It fills in white areas (reduces black spots) and makes the background smoother. The codes for opening and closing are

```
cv2.morphologyEx(threshadap, cv2.MORPH_OPEN, kernel)
```

and

```
cv2.morphologyEx(closing, cv.MORPH_CLOSE, kernel)
```

After processing the image with morphological transformation, the image becomes clearer as you can see from the fig. 6 below.

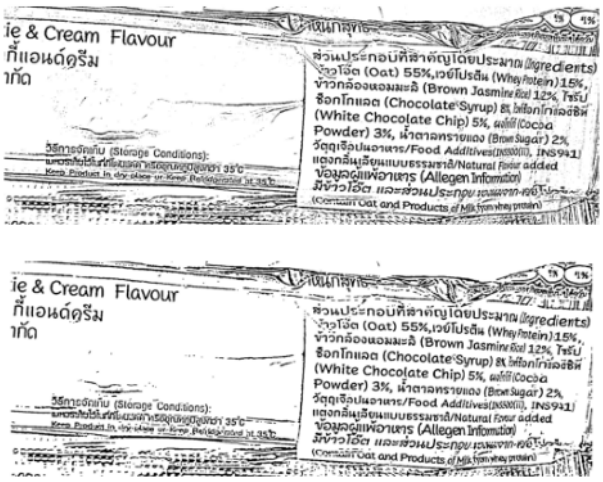


Figure 6: Image from adaptive thresholding (upper) and image after morphological method (lower)

Optical character recognition (OCR)

Optical character recognition (OCR) is a technology that converts text or handwriting from images or documents into machine-readable text. OCR commonly consists of 3 main processes: pre-processing, recognition of character, and post-processing. Pre-processing is the process that removes noises and adjusts the contrast by converting the color image into the black and white image. The area of white is considered the background of the image while the black area is the character that further needs to be processed. Those characters are analyzed through 2 algorithms which are pattern matching and feature detection. After that, the system converts extracted data into ASCII that the system of computer can read for further application.

Using OCR is advantageous to this project since it can extract text from any document, in this case, the food label. There are various OCR engines and one of the most accurate currently available is Tesseract. It supports more than 120 languages including Thai. However, the output might have poor quality if the input is not well-developed or preprocessed. Sometimes the input needs to be processed manually before using Tesseract, for example, dark borders have to be removed, or else it will be misread as a character. The image has to be adjusted if it is somehow rotated or skewed and the low-frequency change in brightness must be processed through a high-pass filter to avoid the determination of some areas by the system's algorithm.

Therefore, the image processing methods mentioned in the section above have to be integrated to enhance the input image before processing with Tesseract and obtain positive outputs with high accuracy. The output after applying tesseract OCR is shown in fig. 7.

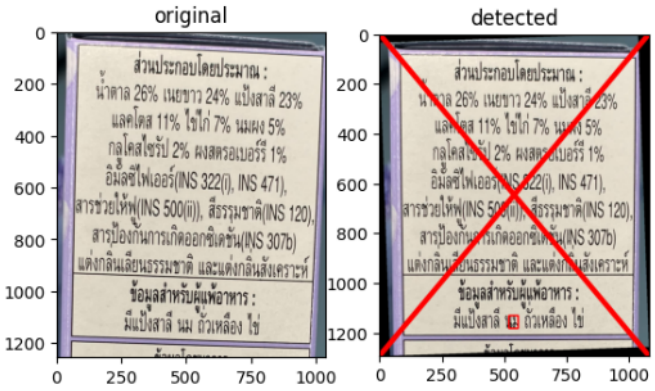


Figure 7: Image after applying tesseract

To apply Tesseract, we mainly used the command

```
Pytesseract.image_to_data
```

The output of the command is in the format of a dictionary with the keys:

```
dict_keys(['level', 'page_num', 'block_num', 'par_num', 'line_num',
   'word_num', 'left', 'top', 'width', 'height', 'conf', 'text'])
```

In this project, the keys that we mainly used are left, top, width, and height, which store the coordinates of the text in the image. We also used the key ‘text’ to visualize the detected words.

The process of milk detection is done by finding both English and Thai words: ‘milk’ and ‘นม’. For English, we start by importing the tesseract library with

```
import pytesseract
```

and applying it to the rotated image by using

```
data = pytesseract.image_to_data(imgoutput_type= pytesseract .Output.
                                 DICT).
```

The text extracted from an image is stored in the variable ‘data’ and we can visualize the text by taking a look at the key data ‘text’. The example of extracted text is shown below (the raw image for the detection is fig. 8),

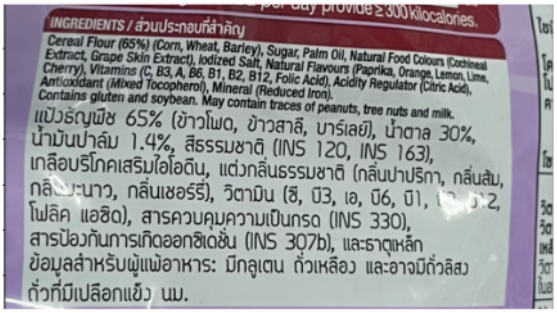


Figure 8: Raw image from the user

```
['', '', '', '', '   ', '', '', '', ' ', ' ', '', ' ', ' ', '', '', '', '
', '', '', '', 'INGREDIENTS', '/', '#2UUSnauinAny', '', 'Cereal', 'Flour',
'(65%)', '(Corn', 'Wheat', 'Barley)', 'Sugar', 'Palm', 'Oil', 'Natural',
'Food', 'Colours', '(Cochineal', '', 'Cherry)', 'Via', '8366', 'BY', 'Be',
'Bia', 'Fl', 'a', 'hy', 'Repuator', 'aeegume', '', 'Contains', 'gluten',
'and', 'soybean', 'May', 'cota', 'race', 'Bae', 'tree', 'nuts', 'and',
'milk.', '', 'WWIStYWs', '65%', '(FraIwa', 'd19a14', 'ursias)', 'uaena',
'30%', '', 'UuwuUUIaU', '1.4%', 'SSSU8IG', '(INS', '120', 'INS', '163)',
'', 'indauSinatasuloladu', 'tcondusssuerd', '(nauUwUSm1', 'nauay', '', '',
'', 'N@IUN', 'NAULsass)', 'Jardu', '(3', '03', '1a', 'U6', 'O1i>', '32',
'', '', '', ' ', '', '', '', 'q', '', 'S', '', 'i', '', '&', '', 'z', '',
'', '', 'AongWaenuT)', 'uv.', '', '', '', '']
```

Pytesseract successfully detect the word ‘milk’ as highlighted but it still contains ‘.’ so python is unable to regcognise the word. We solve this problem by replacing all unnecessary strings ‘ ,|:’,’’ with nothing by using the command from the library re,

```
re.sub(',|:', '', word)
```

Now that the data in the text becomes cleaner, we can let the program located the word ‘milk’ using the command

```
wordrecog = [ i for i, text in enumerate(data["text"]) if text.lower() ==
                                    detectmilk].
```

All of the words are converted into lowercase by using `lower()` command. For the Thai language, we need to find the word ‘นม’. However, sometimes Pytesseract is unable to detect the Thai word since the words are not separated by space like English words. We solve this problem by locating a character ‘น’ that is followed by the character ‘ม’.

After we get the word ‘milk’ or ‘นม’, we created the visualization by building the frame around the detected words. We drew 4 lines to create a frame using the coordinates from the key data: width, height, left, and top. Then, we drew the x symbol on top of the image to show that this label contains milk and is not safe for consumption as shown in fig. 9. On the other hand, if there is no milk on the label, the program returns an image with the ✓ symbol to let the users know that this snack is safe for them as shown in fig. 10.

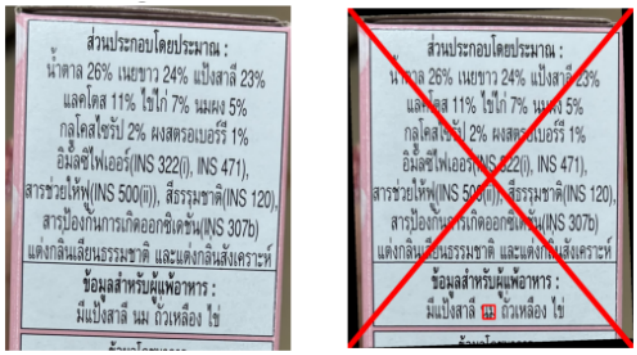


Figure 9: Comparing raw image (left) and output of image containing milk (right)

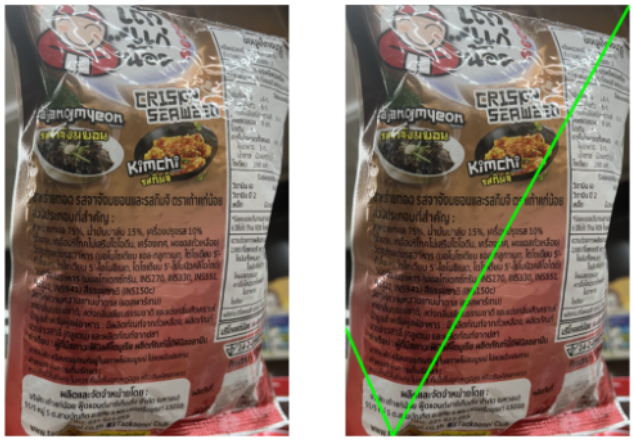


Figure 10: Comparing image raw image (left) and output of image not containing milk (right)

However, the program sometimes cannot detect the word from rotated image because the image is not yet developed. In this case, we developed the rotated image like what we mentioned in the previous section and let the program recognize the word again.

User interface

The user interface or UI is the platform where human-computer contact and communication in a device. It also refers to the manner in which a user engages with a website or application. We use Python as the main platform to develop image processing for milk allergy from snack labels and name the application “Milk Detector”. Despite the fact that Python contains several other graphical user interfaces (GUI) frameworks, as stated by Nederkoorn (2022), PyQt5 is the most well-known used by both UI developers and Python programmers . PyQt is a library that alters the Qt GUI framework which is written in C++ to be programmable in Python or create a whole GUI in Python from scratch. It allows the user to create modern UI and is fully cross-platform, including Linux, macOS, Windows, and even mobile operating systems like iOS and Android.

UI creation

Starts with using the Qt designer program to create the appearance of the application which is easy to work with. The creator can simply add widgets by dragging and dropping them from the toolbox on the left side into the working space. The widget itself can be adjusted, for example, size, alignment, and color by editing the property on the right side as shown in Figure 11.

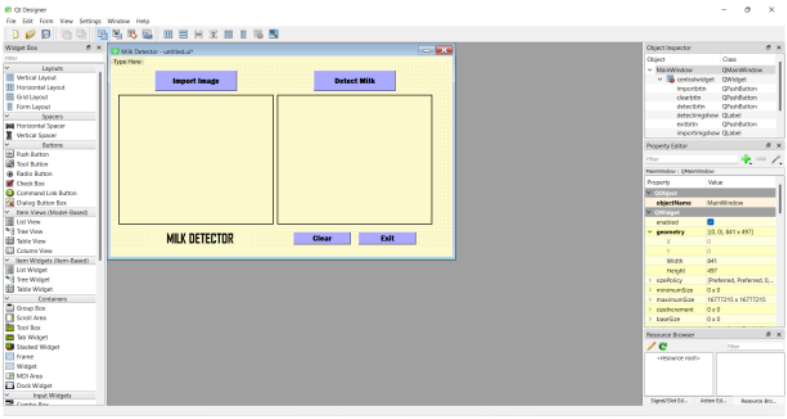


Figure 11: Qt Designer program

During the creation of UI, the creator can use the preview function to see what the real application would look like as shown in figure 12. For the Milk Detector, the main components are the title of the application, buttons, and blank spaces where the image will be displayed.

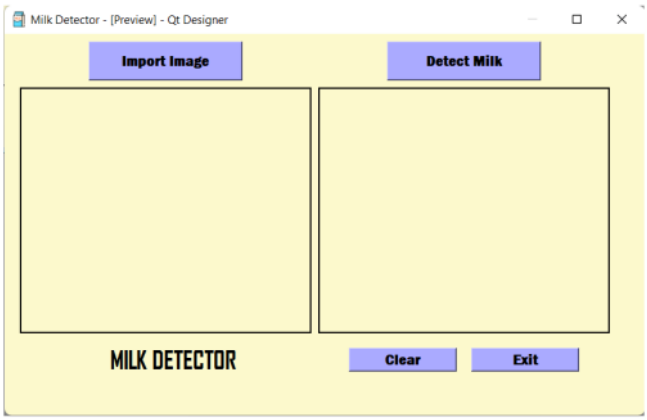


Figure 12: The preview of GUI created in Qt Designer

After the result is satisfactory enough, we save the file in .ui file name extension and open it in any code editor program. The code of .ui which is written in C++ is in figure 13.

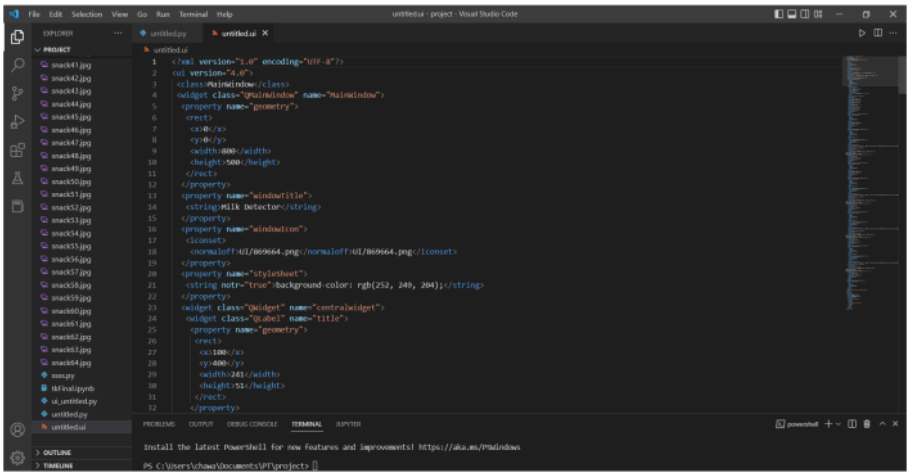


Figure 13: The screenshot of .ui file in visual studio code

Type pyuic5 -x (file name).ui -o (file name).py in TERMINAL to convert from C++ to programmable Python as shown in figure 14. And the result of the conversion is shown in figure 15.

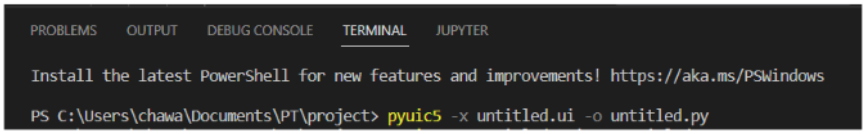


Figure 14: Command code in terminal window

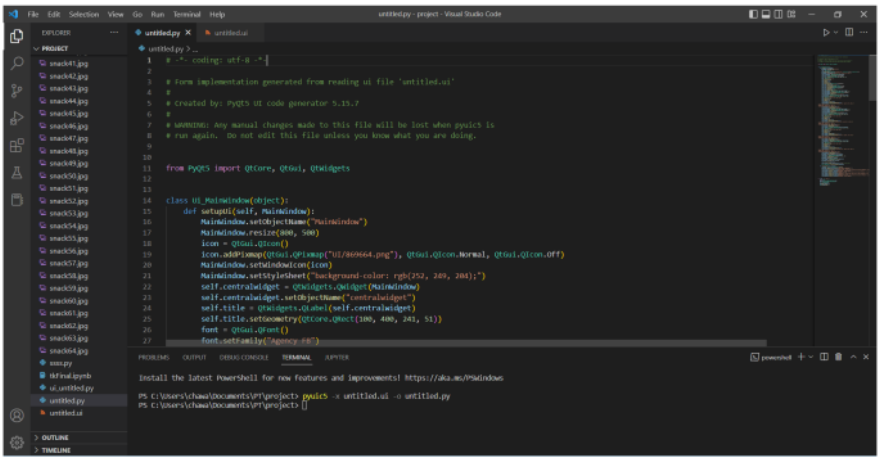


Figure 15: The screenshot of .py file in visual studio code

After changing from .ui to .py, now, we can add custom-coding functions in the file and make the program work since some function is yet built in the Qt Designer. There is only 1 function in the Milk Detector that already works from the Qt Designer which is the “Exit” button. Therefore, need to add 3 more functions according to the UI which are import image, detect milk word, and clear function.

The function to import the image from File Explorer in the computer and show images in the blank space widget on the left sight is shown below.


```
        def browseImage(self):
                options = QtWidgets.QFileDialog.Options()
                options |= QtWidgets.QFileDialog.DontUseNativeDialog
                self.fileName, _ =
QtWidgets.QFileDialog.getOpenFileName(options=options)

                if self.fileName:
                        pattern = ".(jpg|png|jpeg|bmp|jpe|tiff)$"
                        if re.search(pattern,self.fileName):
                                self.setImage(self.fileName)

        def putImageinspace(self,fileName):
                self.importimgshow.setPixmap(QPixmap(fileName))
                self.detectbttn.setEnabled(True)
```

The function to detect the “milk” and “นม” is the same and already explained in the image processing and the OCR part. Furthermore, the detected image will be saved as ‘detectedmilk.jpg’ in the same folder as the original image. And the detected image will be called and shown in the right blank box of the main window of UI. Eventually, new small window will pop up to show whether the snack contain milk as an ingredient or not. The code for these function is shown below.

```
    def detectmilk(self):
            # read non-rotated img
            img_nr = cv2.imread(self.fileName)
            img_nr = cv2.cvtColor(img_nr, cv2.COLOR_RGB2BGR)

            # ---------- Rotate image --------------

            # change image to grayscale
            gray_ori = cv2.cvtColor(img_nr, cv2.COLOR_BGR2GRAY)
            # Apply canny
            canimg = cv2.Canny(gray_ori, 60, 200)
            # Apply Hough
            lines = cv2.HoughLines(canimg, 1, np.pi/180.0, 250, np.array([]))
            # rotate image
            if np.all(lines != None):
                    deg = []
                    for line in lines:
                            theta = line[0,1]
                            deg.append(180*theta/np.pi - 90)

                    deg_mode = mode(deg)
                    # rotation angle in degree
                    if deg_mode >= 20 and deg_mode <= 50:
                            img = ndimage.rotate(img_nr, mode(deg) - 45)
                    elif deg_mode <= -20 and deg_mode >= -50:
                            img = ndimage.rotate(img_nr, mode(deg) - 45 + 90)
                    elif deg_mode < -50 and deg_mode >= -90:
                            img = ndimage.rotate(img_nr, mode(deg) + 90)
                    elif deg_mode > 50 and deg_mode <= 90:
                            img = ndimage.rotate(img_nr, mode(deg) - 90)
                    else:
                            img = ndimage.rotate(img_nr, mode(deg))

            # ------------- Develop image -------------

            # convert rotated image to grayscale
            gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

            # dev image
            threshadap = cv2.adaptiveThreshold(gray, 255,
```

```
cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY, 201, 9)

                # opening
                kernel = cv2.getStructuringElement(cv2.MORPH_RECT,(3,3))
                opening = cv2.morphologyEx(threshadap, cv2.MORPH_OPEN, kernel)

                # closing
                kernel = cv2.getStructuringElement(cv2.MORPH_RECT,(3,3))
                closing = cv2.morphologyEx(opening, cv2.MORPH_CLOSE, kernel)
                dev = closing

                #-------------- Detect English --------------
                new_img = img.copy()
                p1 = p2 = p3 = p4 = 0

                pytesseract.pytesseract.tesseract_cmd = 'C:/Program
Files/Tesseract-OCR/tesseract.exe'

                # Extract the English word
                detectmilk = 'milk'
                data = pytesseract.image_to_data(img,
output_type=pytesseract.Output.DICT)
                for i, word in enumerate(data["text"]):
                        data["text"][i] = re.sub(',|:', '', word)
                wordrecog_en = [ i for i, text in enumerate(data["text"]) if
text.lower() == detectmilk]

                # try enhaced image it real image does not work
                if wordrecog_en == 0:
                        detectmilk = 'milk'
                        data = pytesseract.image_to_data(dev,
output_type=pytesseract.Output.DICT)
                        for i, word in enumerate(data["text"]):
                                data["text"][i] = re.sub(',|:', '', word)
                        wordrecog_en = [ i for i, text in enumerate(data["text"]) if
text.lower() == detectmilk]

                for word in wordrecog_en:
                        # get top, left position and width, height of extracted word
                        w = data["width"][word]
                        h = data["height"][word]
                        l = data["left"][word]
                        t = data["top"][word]
                        # define 4 coordinates of box
                        x1 = (l, t)
                        x2 = (w + l, t)
                        x3 = (w + l, h + t)
```

```
                        x4 = (l, h + t)
                        # draw 4 lines to create the box
                        new_img = cv2.line(new_img, x1, x2, color=(255, 0, 0),
thickness=5)
                        new_img = cv2.line(new_img, x2, x3, color=(255, 0, 0),
thickness=5)
                        new_img = cv2.line(new_img, x3, x4, color=(255, 0, 0),
thickness=5)
                        new_img = cv2.line(new_img, x4, x1, color=(255, 0, 0),
thickness=5)

                # ---------- Detect Thai --------------

                # reuse parameters
                x1 = x2 = x3 = x4 = 0
                h, w = gray.shape
                pytesseract.pytesseract.tesseract_cmd = 'C:/Program
Files/Tesseract-OCR/tesseract.exe'
                # Extract Thai string
                data_th = pytesseract.image_to_data(img,
output_type=pytesseract.Output.DICT, lang="tha")
                # detect both 'น' that follows by 'ม' and 'นม'
                wordrecog_m = [ i for i, text in enumerate(data_th['text']) if text ==
'น']
                wordrecog_k = [ i for i, text in enumerate(data_th['text']) if text ==
'ม']
                wordrecog_th = []
                wordrecog_th = [ i for i, text in enumerate(data_th['text']) if text
== 'นม']
                for i,m in enumerate(wordrecog_m):
                        for j,k in enumerate(wordrecog_k):
                                if k-m == 1:
                                        wordrecog_th.append(m)

                # try enhaced image if real image does not work
                if wordrecog_th == []:
                        data_th = pytesseract.image_to_data(dev,
output_type=pytesseract.Output.DICT, lang="tha")
                        wordrecog_m = [ i for i, text in enumerate(data_th['text']) if
text == 'น']
                        wordrecog_k = [ i for i, text in enumerate(data_th['text']) if
text == 'ม']
                        wordrecog_th = []
                        wordrecog_th = [ i for i, text in enumerate(data_th['text'])
if text == 'นม']
                        for i,m in enumerate(wordrecog_m):
```

```
                                for j,k in enumerate(wordrecog_k):
                                        if k-m == 1:
                                                wordrecog_th.append(m)

                # Draw X if the snack contains milk
                if len(wordrecog_th) + len(wordrecog_en) != 0:
                        new_img = cv2.line(new_img, (0,0), (w,h), color=(255, 0, 0),
thickness=20)
                        new_img = cv2.line(new_img, (0,h), (w,0), color=(255, 0, 0),
thickness=20)
                # Draw ✓ if the snack does not contain milk
                else:
                        new_img = cv2.line(new_img, (0,np.int(h*3/4)),
(np.int(w/6),h), color=(0, 255, 0), thickness=20)
                        new_img = cv2.line(new_img, (np.int(w/6),h), (w,0), color=(0,
255, 0), thickness=20)

                # Draw box around detected word
                for word in wordrecog_th:
                        # get top, left position and width, height of extracted word
                        w = data_th["width"][word] + data_th["width"][word+1]
                        h = data_th["height"][word]
                        l = data_th["left"][word]
                        t = data_th["top"][word]
                        # define 4 coordinates of the box
                        x1 = (l, t)
                        x2 = (w + l, t)
                        x3 = (w + l, t + h)
                        x4 = (l, h + t)
                        # draw 4 lines to create the box
                        new_img = cv2.line(new_img, x1, x2, color=(255, 0, 0),
thickness=5)
                        new_img = cv2.line(new_img, x2, x3, color=(255, 0, 0),
thickness=5)
                        new_img = cv2.line(new_img, x3, x4, color=(255, 0, 0),
thickness=5)
                        new_img = cv2.line(new_img, x4, x1, color=(255, 0, 0),
thickness=5)
                plt.imsave("detectedmilk.jpg", new_img)

                pixmap = QPixmap("detectedmilk.jpg")
                self.detectimgshow.setPixmap(pixmap)
                self.detectbttn.setEnabled(True)

                #Create window to show output in text whether it contains milk or not
                messagebox = QMessageBox()
                #if it has milk as an ingredient, show "CONTAIN MILK"
```

```
            if len(wordrecog_th) + len(wordrecog_en) != 0:
                    messagebox.setText("CONTAIN MILK")
            #if it not has milk as an ingredient, show "NOT CONTAIN MILK"
            else:
                    messagebox.setText("NOT CONTAIN MILK")
            messagebox.setWindowTitle("SHOW OUTPUT")
            messagebox.setStandardButtons(QMessageBox.Close)
            x = messagebox.exec_()
```

And the last function is clear the output image is shown below

```
        def clearimage(self):
                self.detectimgshow.clear()
```

More importantly, the application will not be working if we do not create connections between widgets and the function together. The code down below used to connect those connections. After that, run the code to start the application and test the system.

```
                #if clicked "Import Image" button, the function "getImage" will
work and image will be imported
                self.Importbttn.clicked.connect(self.browseImage)
                #if clicked "Detect Milk" button, the function "detectmilk" will work
and detected image will shown
                self.detectbttn.clicked.connect(self.detectmilk)
                #set the "Detect Milk" button to be false and be true after imported
the image
                self.detectbttn.setEnabled(False)
                #if click "Clear", the detected image will be clear out
                self.clearbttn.clicked.connect(self.clearimage)
                #if click "Exit" button, the application will close
                self.exitbttn.clicked.connect(MainWindow.close) # type: ignore
```

Milk detector application

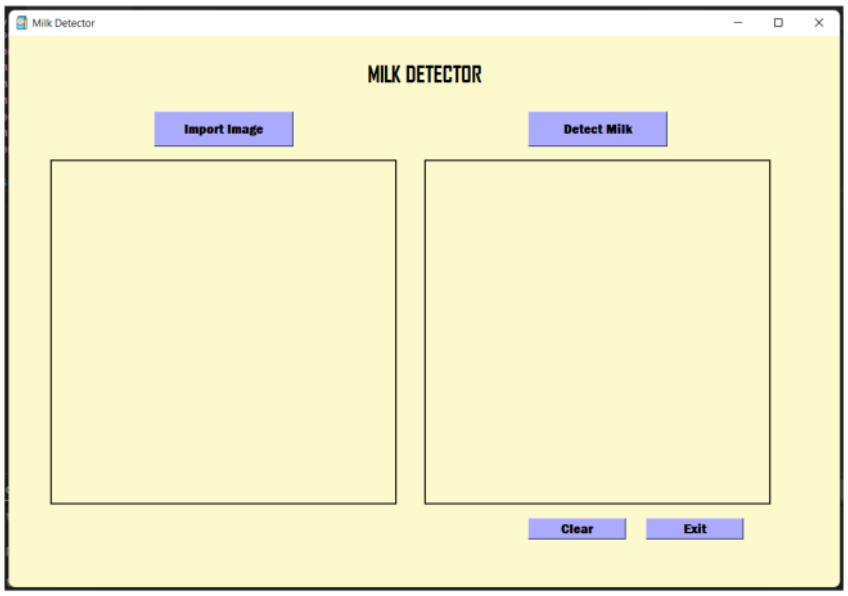


Figure 16: The appearance of the Milk Detector application

The figure above shows the final appearance of the Milk Detector application that is built by using PyQt5. The name of the program is Milk Detector and there is a milk logo shown next to it. For the contents of the program, the title is shown at the top. Inferior to it is the import and detect button, blank spaces, and clear and exit button, respectively. Moreover, tooltips were also added to each button to explain how it works as shown in figure 17.

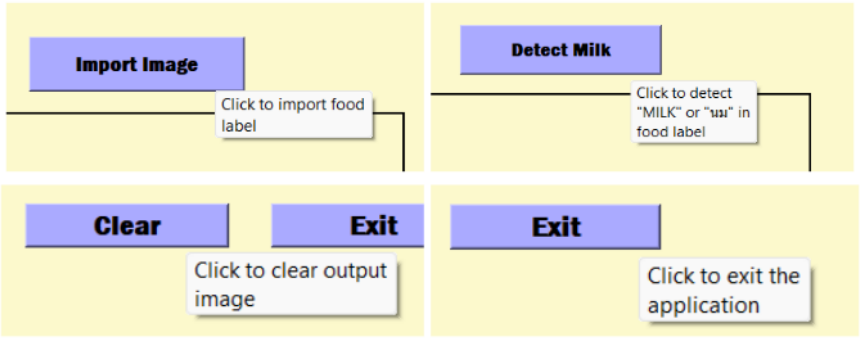


Figure 17: The tooltips for each button

A simple widget, blank space, is created to display the image of the snack's label. Once the user clicks the “Import Image” button and selected desired image, the image will be displayed on the left widget as shown in figure 18.

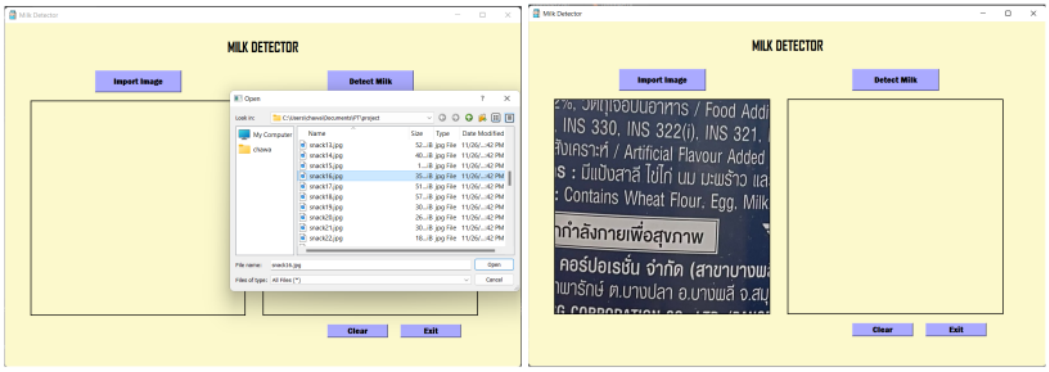


Figure 18: Import image and image display inside the widget

After clicking “Detect Milk”, the widget will show the image of whether it has milk as an ingredient or not. If the snack has milk as a component, the image will display on the right widget with the ‘x’ symbol, and a new widget containing the sentence “contain milk” will pop up as shown in figure 19.

Figure 19: the result of an image of a dairy product after clicking ‘detect’ button.

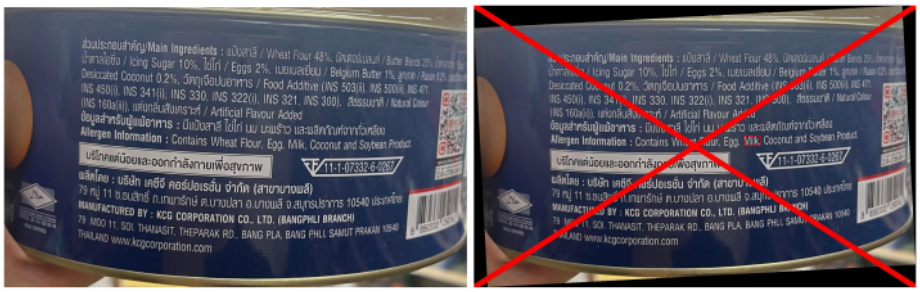


Figure 20: The original image and final actual image

On the other hand, the image with the '✓' symbol will display on the right widget with the new widget containing the sentence "not contain milk" as shown in figure 23 below.

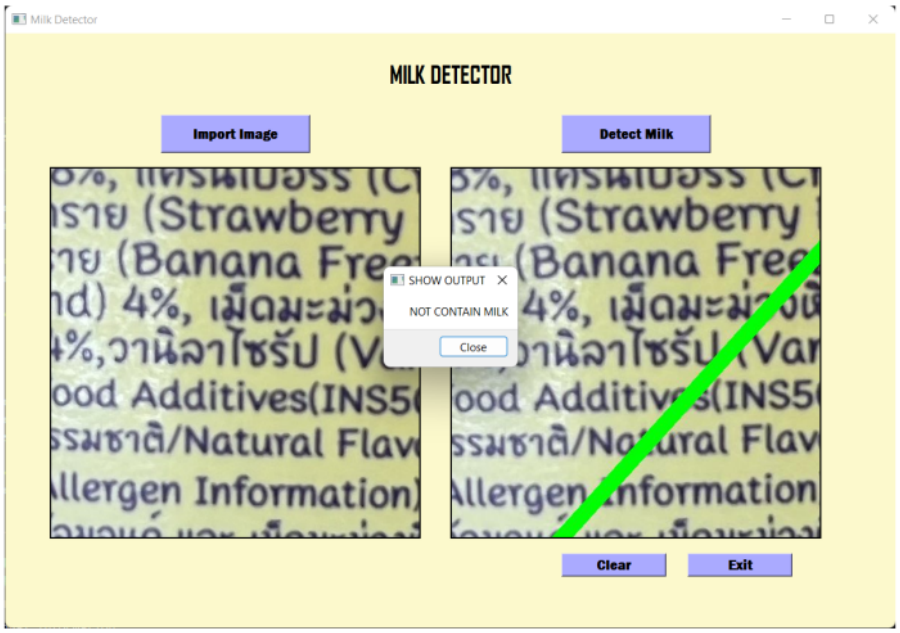


Figure 21: The result of an image of a non-dairy product after clicking 'detect' button.

Figure 22: The original image

Figure 23: The actual final image

Evaluation

Dataset

Dataset contains a total of 86 images of 86 different snack labels. The images were taken by 3 different phone cameras from 7-11 convenience stores and contain both English and Thai languages. The folder of the dataset can be accessed through the google drive link: https://drive.google.com/drive/folders/1J192l0_9e9zZiOKaNDQgjBWvs3n7P5l8?usp=sharing

Accuracy

From a total of 86 images, there are 63 images that the system can predict correctly whether the snack contains milk or not with 8 True Negatives. However, the other 23 images are predicted incorrectly with 1 False Positive (predicted non-dairy products as containing milk) and 22 False Negatives (predicted dairy products as not containing milk).

Therefore, the accuracy of the model is **73.26%**.

Conclusion and Discussion

Conclusion

It is crucial for people to choose the goods they use with care when it concerns their health. In order to lessen the effects of any allergies or ongoing medical conditions. Additionally, the number of health issues brought on by nutrition is growing daily. In order to avoid any health issues, individuals should thoroughly inspect the parts and ingredients of the things they buy. This study aims to identify the nutrients that can cause specific health issues associated with the intake of dairy products. An application that is user-friendly and simple to use in everyday life has been created for this aim using image processing-based techniques. However, the application has an error in showing the detected image due to the format of the QImage function. With the time limitation and lack of knowledge, we could not find the right format to show the image yet, we hoped that this error will be fixed in future development.

OCR technologies are used to address a variety of issues. The ability of this technology to quickly transfer hundreds of pages of data to computers or databases is the main factor behind its expanding use. By using OCR technology in this study, it is hoped to promote persons' consumption of nutritious foods. Using image processing techniques on the images on the items, the content of dairy products could be analyzed. We have successfully completed to analyzed all the 35 snack products which will be detected by the word “milk” in English and “นม” in Thai and displayed in the User Interface. After testing all 86 image data, we can get the correct result for 63 images which makes the accuracy percentage **73.26%**.

Discussion

There are some false positives and negatives caused by extracting text incorrectly. This might occur because the character is too small, too close to each other, or the image is too blurred. The problem that we found is mostly from Thai language detection because there are many fonts used on the food label and it is sometimes difficult for the program to differentiate between the character ‘น’ ‘ป’ ‘ม’ and ‘บ’. The examples of wrong detections are shown below.

Figure 25: The program is unable to detect the word ‘นม’ and ‘milk’

Figure 26: The wrong detection of the word ‘นม’

Future Improvement

This project was created specifically for dairy products, especially milk, however, it may be used for various items. It is applicable to all products with intricate and expansive databases. Making a user profile that is more in-depth is another technique to enhance the application. The parameters of the patient's assay results can be used to create the user profile, enabling practically all parameters to receive nutritional counseling. Making recommendations for food allergy.

Another good example of applying image processing to detect food allergies is using a barcode. Using image processing techniques on the barcode images on the snack items, the content of dairy products could be analyzed. The application may request the user to submit a list of items that they are allergic to and should avoid eating. It then does a data analysis using the barcode image as the product's parameter and notifies the user whether or not he or she can consume the product. As a result, people with various health issues might avoid the symptoms that they might otherwise experience when consuming dairy products. This not only helps the user to prevent food allergies and misconsuming, but it also helps protect the user's personal information and user-friendly application.

Another technique is to apply machine learning to detect the pattern of the algorithm to increase the accuracy and precision of the system. In addition, a better database and data acquisition system for food labels should be improved to be easier to detect food allergies.

Appendix

Output

Examples of milk detection from food labels (35 examples)

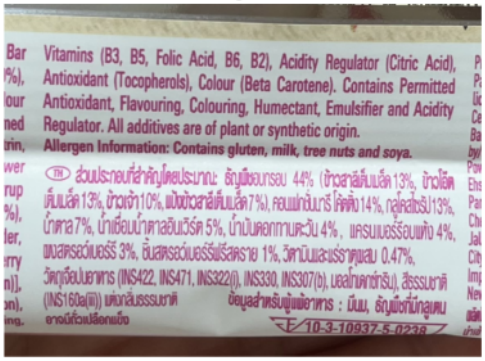
Original

Output

Original

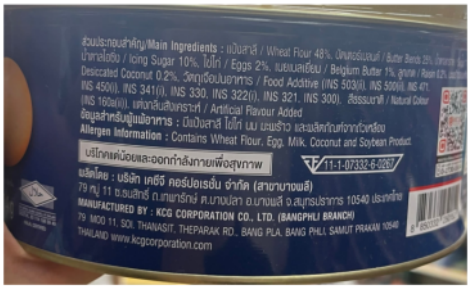
Output

Original



Output

Original

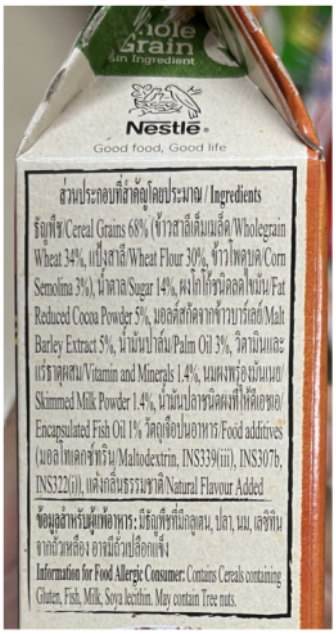


Output

Original

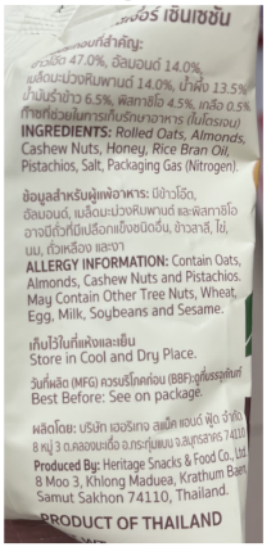
Output

Original

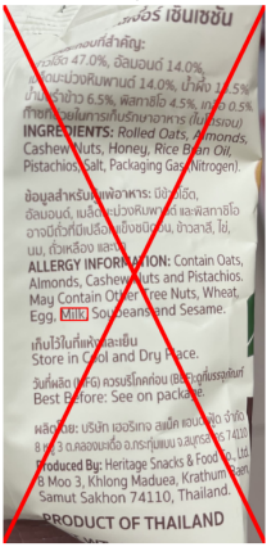


Output

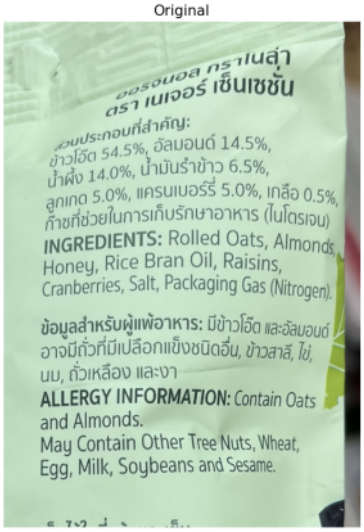
Original



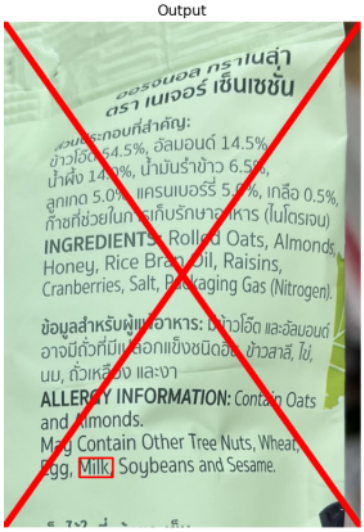
Output



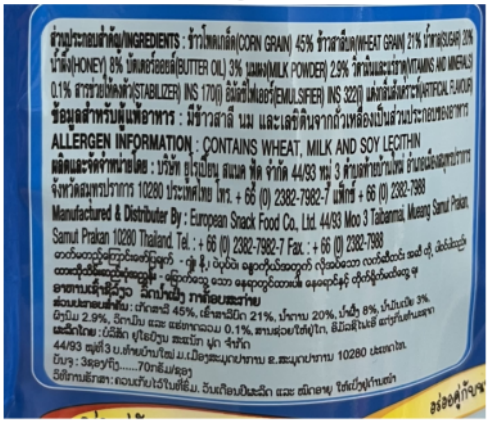
Original



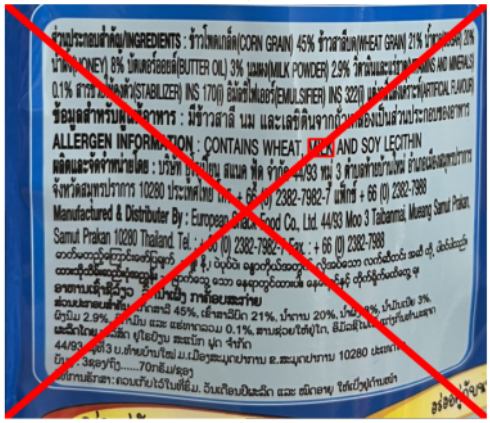
Output



Original



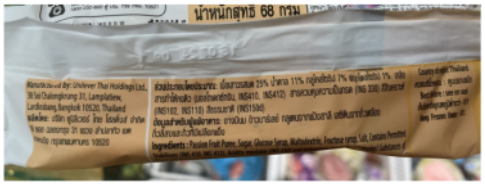
Output



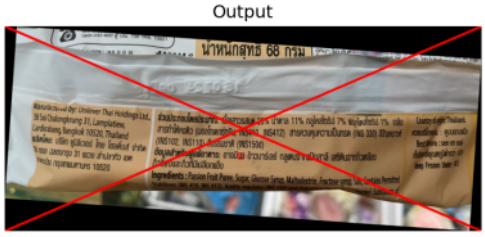
Original

Output

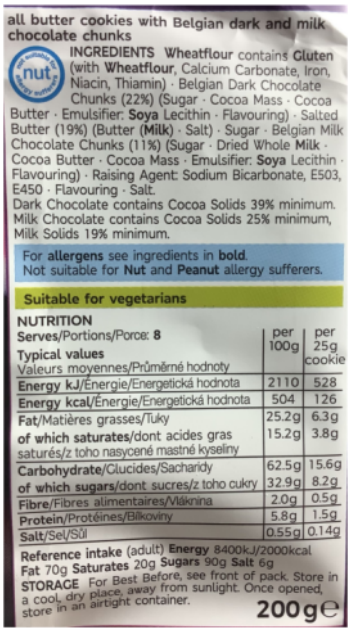
Original



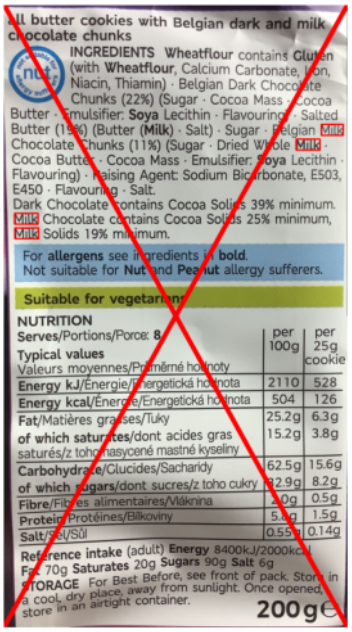
Output



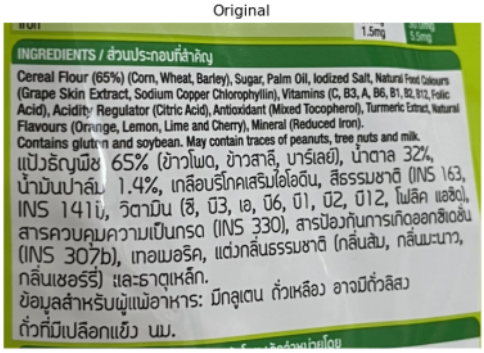
Original



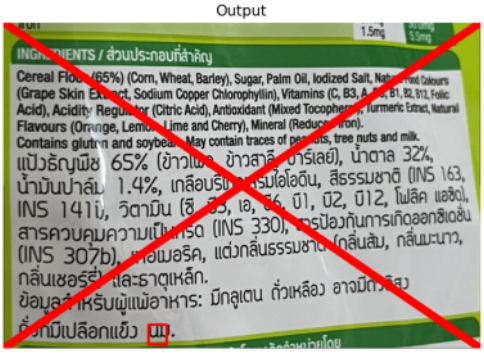
Output



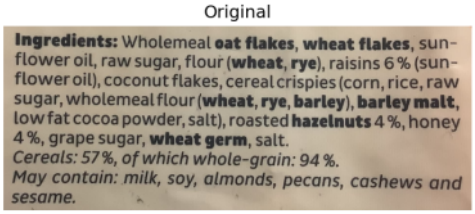
Original



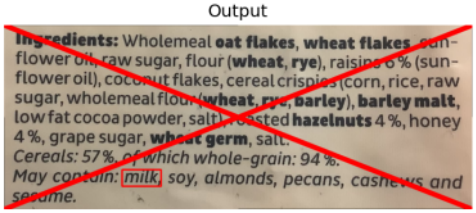
Output



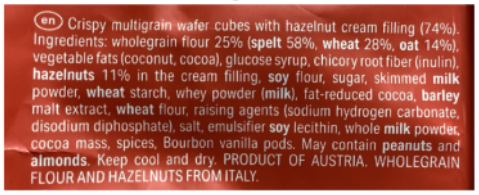
Original



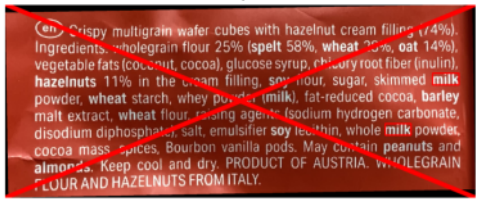
Output



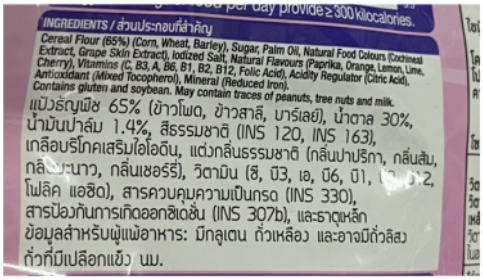
Original



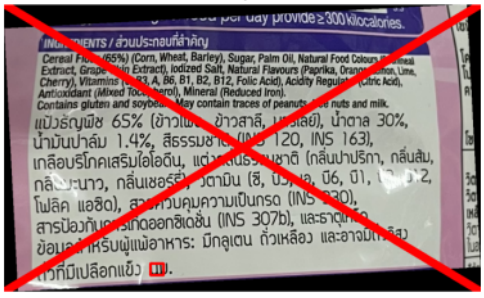
Output



Original



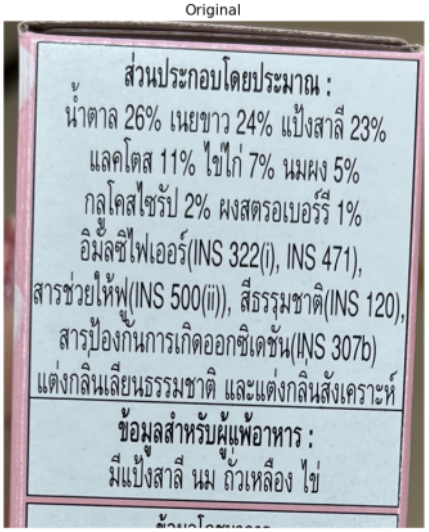
Output



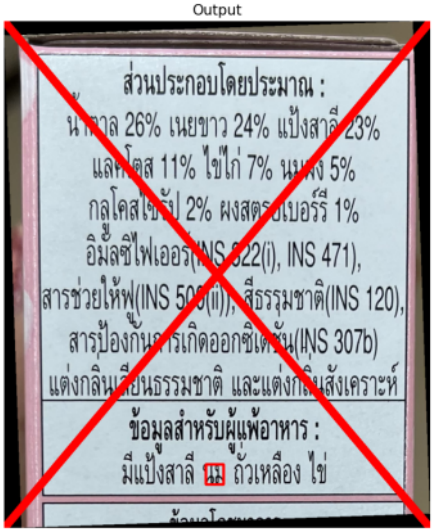
Original

Output

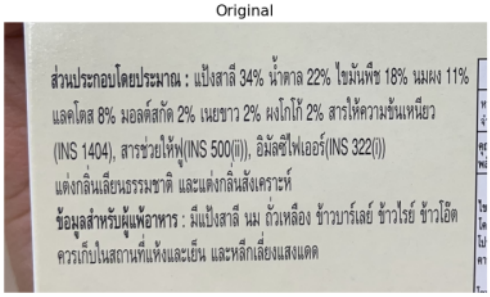
Original



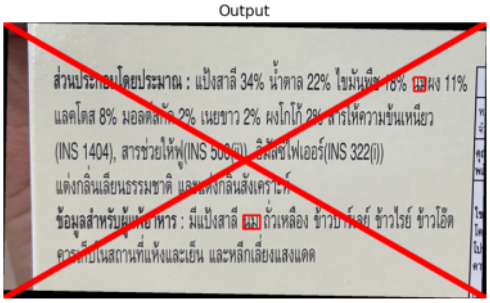
Output



Original

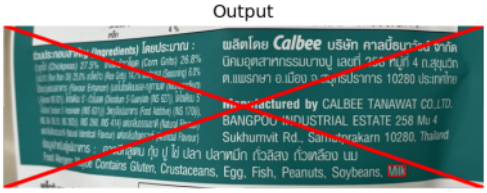


Output



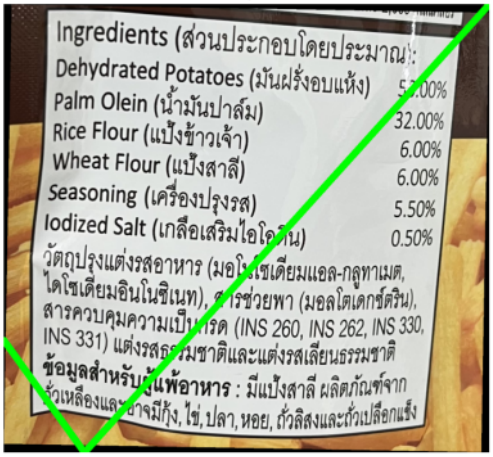
Original

Output



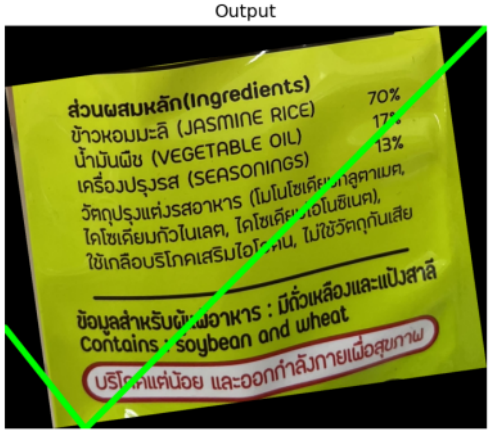
Original

Output

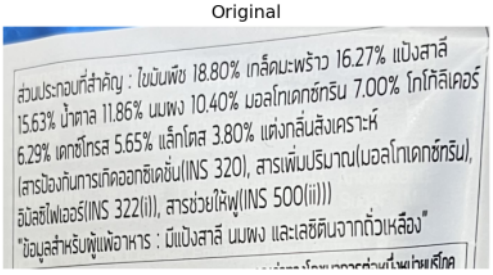


Original

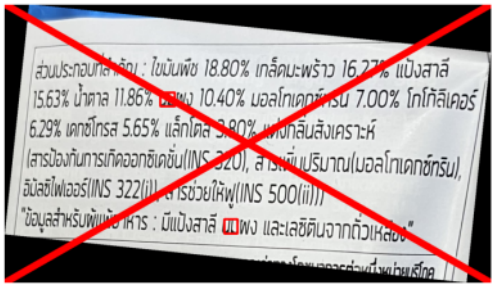
Output



Original

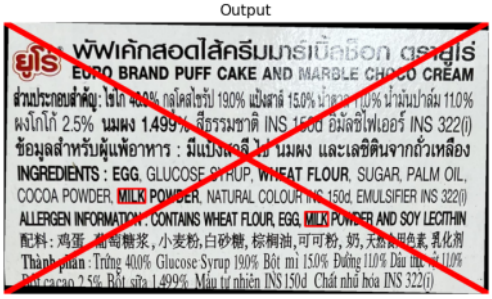


Output



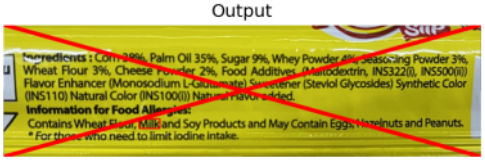
Original

Output



Original

Output

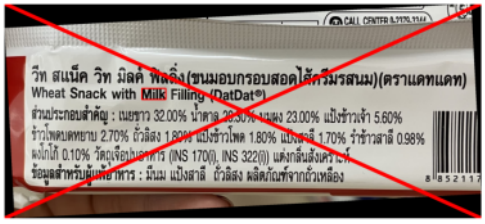


Original

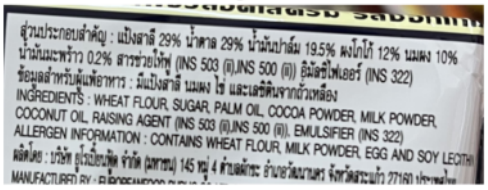
Output

Original

Output

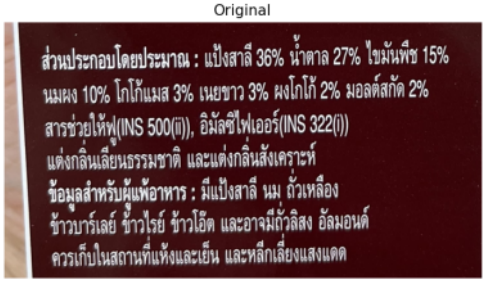


Original

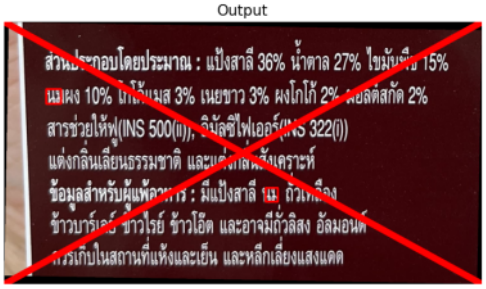


Output

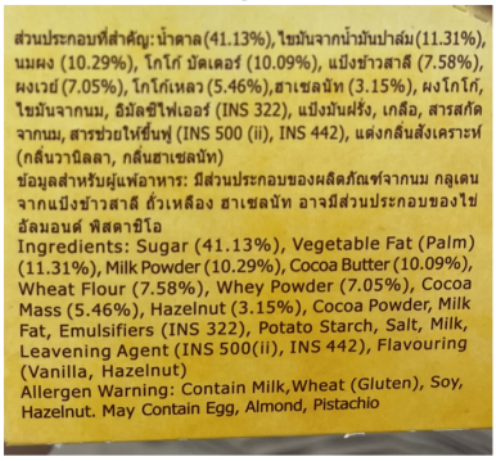
Original



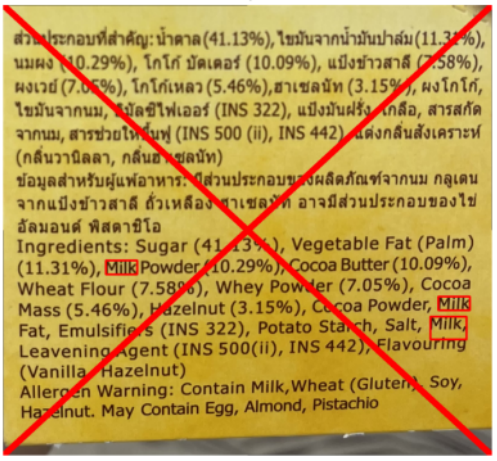
Output



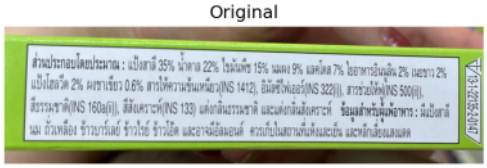
Original



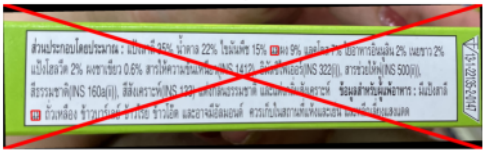
Output



Original

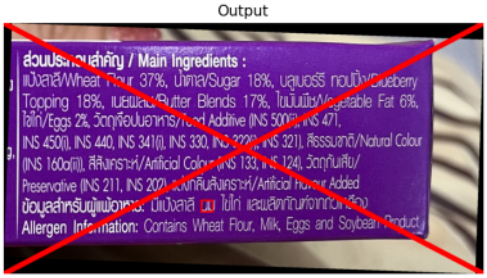


Output

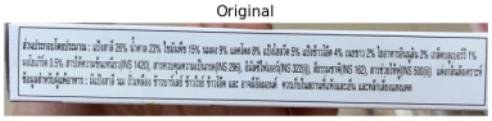


Original

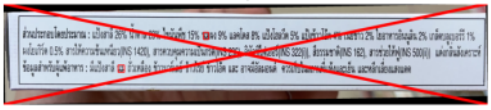
Output



Original

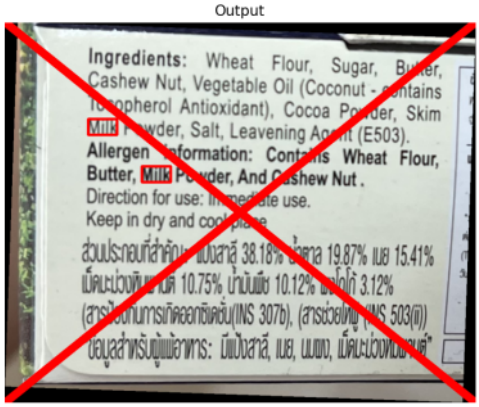


Output



Original

Output



Source code without UI

```
import pytesseract
import cv2
import matplotlib.pylab as plt
import re
import numpy as np
from scipy import ndimage
from statistics import mode
import sys

pytesseract.pytesseract.tesseract_cmd = 'C:/Program Files/Tesseract-OCR/tesseract.exe'

# read non-rotated img
img_nr = cv2.imread('snack2.jpg')
img_nr = cv2.cvtColor(img_nr, cv2.COLOR_RGB2BGR)

# ------------------------------ Rotate image ---------------------------------------

# change image to grayscale
gray_ori = cv2.cvtColor(img_nr, cv2.COLOR_BGR2GRAY)

# Apply canny
canimg = cv2.Canny(gray_ori, 60, 200)

# Apply Hough
lines = cv2.HoughLines(canimg, 1, np.pi/180.0, 250, np.array([]))

# rotate image
if np.all(lines != None):
    deg = []
    # convert radian to degree
    for line in lines:
        theta = line[0,1]
        deg.append(180*theta/np.pi - 90)
    # Find the angle that are detected most frequently
    deg_mode = mode(deg)

    # rotate the image according to the tilted angle
    if deg_mode >= 20 and deg_mode <= 50:
        img = ndimage.rotate(img_nr, mode(deg) - 45)
    elif deg_mode <= -20 and deg_mode >= -50:
        img = ndimage.rotate(img_nr, mode(deg) - 45 + 90)
    elif deg_mode < -50 and deg_mode >= -90:
        img = ndimage.rotate(img_nr, mode(deg) + 90)
    elif deg_mode > 50 and deg_mode <= 90:
        img = ndimage.rotate(img_nr, mode(deg) - 90)
```

```
    else:
        img = ndimage.rotate(img_nr, mode(deg))

# ------------------------------ Develop image ---------------------------------

# convert rotated image to grayscale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# dev image
threshadap = cv2.adaptiveThreshold(gray, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
cv2.THRESH_BINARY, 201, 9)

# opening
kernel = cv2.getStructuringElement(cv2.MORPH_RECT,(3,3))
opening = cv2.morphologyEx(threshadap, cv2.MORPH_OPEN, kernel)

# closing
kernel = cv2.getStructuringElement(cv2.MORPH_RECT,(3,3))
closing = cv2.morphologyEx(opening, cv2.MORPH_CLOSE, kernel)
dev = closing

#------------------------------ Detect English ---------------------------------

new_img = img.copy()
p1 = p2 = p3 = p4 = 0

# Extract the English word
pytesseract.pytesseract.tesseract_cmd = 'C:/Program Files/Tesseract-OCR/tesseract.exe'
detectmilk = 'milk'
data = pytesseract.image_to_data(img, output_type=pytesseract.Output.DICT)
# Delete unnecessary string
for i, word in enumerate(data["text"]):
    data["text"][i] = re.sub(',|:', '', word)
# Detect 'milk' from string
wordrecog_en = [ i for i, text in enumerate(data["text"]) if text.lower() ==
detectmilk]

# Try enhaced image if raw image does not work
if wordrecog_en == 0:
    detectmilk = 'milk'
    data = pytesseract.image_to_data(dev, output_type=pytesseract.Output.DICT)
    for i, word in enumerate(data["text"]):
        data["text"][i] = re.sub(',|:', '', word)
    wordrecog_en = [ i for i, word in enumerate(data["text"]) if word.lower() ==
detectmilk]

# Draw box around detected word
```

```
for word in wordrecog_en:
    # get top, left position and width, height of extracted word
    w = data["width"][word]
    h = data["height"][word]
    l = data["left"][word]
    t = data["top"][word]
    # define 4 coordinates of box
    x1 = (l, t)
    x2 = (w + l, t)
    x3 = (w + l, t + h)
    x4 = (l, h + t)
    # draw 4 lines to create the box
    new_img = cv2.line(new_img, x1, x2, color=(255, 0, 0), thickness=5)
    new_img = cv2.line(new_img, x2, x3, color=(255, 0, 0), thickness=5)
    new_img = cv2.line(new_img, x3, x4, color=(255, 0, 0), thickness=5)
    new_img = cv2.line(new_img, x4, x1, color=(255, 0, 0), thickness=5)

# Draw X on top of the image if milk is detected
if len(wordrecog_en) != 0:
    h, w, c = img.shape
    new_img = cv2.line(new_img, (0,0), (w,h), color=(255, 0, 0), thickness=20)
    new_img = cv2.line(new_img, (0,h), (w,0), color=(255, 0, 0), thickness=20)
    # If 'milk' is already found, there is no need to detect 'นม'. The program stops
here.
    plt.imsave("detectedmilk.jpg", new_img)
    fig = plt.figure("Milk Detection")
    plt.subplot(1,2,1), plt.imshow(img_nr), plt.title('Original'), plt.axis('off')
    plt.subplot(1,2,2), plt.imshow(new_img), plt.title('Output'), plt.axis('off')
    plt.show()
    print("already found milk, stop here")
    sys.exit(0)

# ---------------------------------- Detect Thai ----------------------------------

# reuse parameters
p1 = p2 = p3 = p4 = 0
h, w = gray.shape

pytesseract.pytesseract.tesseract_cmd = 'C:/Program Files/Tesseract-OCR/tesseract.exe'
# detect Thai word 'นม'
data_th = pytesseract.image_to_data(img, output_type=pytesseract.Output.DICT,
lang="tha")
# detect both 'น' that follows by 'ม' and 'นม'
wordrecog_m = [ i for i, text in enumerate(data_th['text']) if text == 'น']
wordrecog_k = [ i for i, text in enumerate(data_th['text']) if text == 'ม']
wordrecog_th = []
wordrecog_th = [ i for i, text in enumerate(data_th['text']) if text == 'นม']
```

```
for i,m in enumerate(wordrecog_m):
    for j,k in enumerate(wordrecog_k):
        if k-m == 1:
            wordrecog_th.append(m)

# try enhaced image if real image does not work
if wordrecog_th == 0:
    data_th = pytesseract.image_to_data(dev, output_type=pytesseract.Output.DICT,
lang="tha")
    wordrecog_m = [ i for i, text in enumerate(data_th['text']) if text == 'น']
    wordrecog_k = [ i for i, text in enumerate(data_th['text']) if text == 'ม']
    wordrecog_th = []
    wordrecog_th = [ i for i, text in enumerate(data_th['text']) if text == 'นม']
    for i,m in enumerate(wordrecog_m):
        for j,k in enumerate(wordrecog_k):
            if k-m == 1:
                wordrecog_th.append(m)

# Draw X if the snack contains milk
if len(wordrecog_th) + len(wordrecog_en) != 0:
    new_img = cv2.line(new_img, (0,0), (w,h), color=(255, 0, 0), thickness=20)
    new_img = cv2.line(new_img, (0,h), (w,0), color=(255, 0, 0), thickness=20)
# Draw ✓ if the snack does not contain milk
else:
    new_img = cv2.line(new_img, (0,np.int(h*3/4)), (np.int(w/6),h), color=(0, 255, 0),
thickness=20)
    new_img = cv2.line(new_img, (np.int(w/6),h), (w,0), color=(0, 255, 0),
thickness=20)

# Draw box around detected word
for word in wordrecog_th:
    # get top, left position and width, height of extracted word
    w = data_th["width"][word] + data_th["width"][word+1]
    h = data_th["height"][word]
    l = data_th["left"][word]
    t = data_th["top"][word]
    # define 4 coordinates of the box
    x1 = (l, t)
    x2 = (w + l, t)
    x3 = (w + l, t + h)
    x4 = (l, h + t)
    # draw 4 lines to create the box
    new_img = cv2.line(new_img, x1, x2, color=(255, 0, 0), thickness=5)
    new_img = cv2.line(new_img, x2, x3, color=(255, 0, 0), thickness=5)
    new_img = cv2.line(new_img, x3, x4, color=(255, 0, 0), thickness=5)
    new_img = cv2.line(new_img, x4, x1, color=(255, 0, 0), thickness=5)
```

```
plt.imsave("detectedmilk.jpg", new_img)

# ------------------------------ Show image ----------------------------------------

fig = plt.figure("Milk Detection")
plt.subplot(1,2,1), plt.imshow(img_nr), plt.title('Original'), plt.axis('off')
plt.subplot(1,2,2), plt.imshow(new_img), plt.title('Output'), plt.axis('off')
plt.show()
```

Source code with UI

However, the image is still for the reason aforementioned.

```
from PyQt5 import QtCore, QtGui, QtWidgets
import os , re , cv2 , pytesseract
from PyQt5.QtGui import  QIcon , QPixmap, QImage
import numpy as np
import matplotlib.pylab as plt
from PIL import Image, ImageFont, ImageDraw
from itertools import chain
from scipy import ndimage
import statistics
from statistics import mode
import sys

class Ui_MainWindow(object):
        def __init__(self):
                pytesseract.pytesseract.tesseract_cmd = 'C:\Program
Files\Tesseract-OCR\\tesseract.exe'

        def setupUi(self, MainWindow):
                MainWindow.setObjectName("MainWindow")
                MainWindow.resize(1200, 800)
                icon = QtGui.QIcon()
                icon.addPixmap(QtGui.QPixmap("UI/869664.png"), QtGui.QIcon.Normal,
QtGui.QIcon.Off)
                MainWindow.setWindowIcon(icon)
                MainWindow.setStyleSheet("background-color: rgb(252, 249, 204);")
                self.centralwidget = QtWidgets.QWidget(MainWindow)
                self.centralwidget.setObjectName("centralwidget")
                self.title = QtWidgets.QLabel(self.centralwidget)
                self.title.setGeometry(QtCore.QRect(480, 30, 241, 51))
                font = QtGui.QFont()
                font.setFamily("Agency FB")
                font.setPointSize(20)
                font.setBold(True)
                font.setItalic(False)
                font.setWeight(75)
                font.setStyleStrategy(QtGui.QFont.PreferDefault)
                self.title.setFont(font)
                self.title.setAutoFillBackground(False)
                self.title.setFrameShape(QtWidgets.QFrame.NoFrame)
                self.title.setFrameShadow(QtWidgets.QFrame.Plain)
                self.title.setAlignment(QtCore.Qt.AlignCenter)
                self.title.setWordWrap(False)
                self.title.setObjectName("title")
                self.importimgshow = QtWidgets.QLabel(self.centralwidget)
```

```
                self.importimgshow.setGeometry(QtCore.QRect(60, 180, 500, 500))
                self.importimgshow.setFrameShape(QtWidgets.QFrame.WinPanel)
                self.importimgshow.setText("")
                self.importimgshow.setObjectName("importimgshow")
                self.detectbttn = QtWidgets.QPushButton(self.centralwidget)
                self.detectbttn.setGeometry(QtCore.QRect(750, 110, 201, 51))
                font = QtGui.QFont()
                font.setFamily("Franklin Gothic Demi Cond")
                font.setPointSize(12)
                font.setBold(True)
                font.setWeight(75)
                self.detectbttn.setFont(font)
                self.detectbttn.setAutoFillBackground(False)
                self.detectbttn.setStyleSheet("background-color: rgb(170, 170,
16
255);\n"
"color: rgb(0, 0, 0);\n"
"border-color: rgb(255, 255, 255);\n"
"selection-background-color: rgb(145, 218, 0);")
                self.detectbttn.setAutoDefault(False)
                self.detectbttn.setObjectName("detectbttn")
                self.detectimgshow = QtWidgets.QLabel(self.centralwidget)
                self.detectimgshow.setGeometry(QtCore.QRect(600, 180, 500, 500))
                self.detectimgshow.setFrameShape(QtWidgets.QFrame.WinPanel)
                self.detectimgshow.setText("")
                self.detectimgshow.setObjectName("detectimgshow")
                self.exitbttn = QtWidgets.QPushButton(self.centralwidget)
                self.exitbttn.setGeometry(QtCore.QRect(920, 700, 141, 31))
                font = QtGui.QFont()
                font.setFamily("Franklin Gothic Demi Cond")
                font.setPointSize(12)
                font.setBold(True)
                font.setWeight(75)
                self.exitbttn.setFont(font)
                self.exitbttn.setAutoFillBackground(False)
                self.exitbttn.setStyleSheet("background-color: rgb(170, 170, 255);\n"
19
"color: rgb(0, 0, 0);\n"
"border-color: rgb(255, 255, 255);\n"
"selection-background-color: rgb(145, 218, 0);")
                self.exitbttn.setAutoDefault(False)
                self.exitbttn.setObjectName("exitbttn")
                self.clearbttn = QtWidgets.QPushButton(self.centralwidget)
                self.clearbttn.setGeometry(QtCore.QRect(750, 700, 141, 31))
                font = QtGui.QFont()
                font.setFamily("Franklin Gothic Demi Cond")
                font.setPointSize(12)
                font.setBold(True)
                font.setWeight(75)
```

```
                self.clearbttn.setFont(font)
                self.clearbttn.setAutoFillBackground(False)
                self.clearbttn.setStyleSheet("background-color: rgb(170, 170, 255);\n"
"color: rgb(0, 0, 0);\n"
"border-color: rgb(255, 255, 255);\n"
"selection-background-color: rgb(145, 218, 0);")
                self.clearbttn.setAutoDefault(False)
                self.clearbttn.setObjectName("clearbttn")
                self.Importbttn = QtWidgets.QPushButton(self.centralwidget)
                self.Importbttn.setGeometry(QtCore.QRect(210, 110, 201, 51))
                font = QtGui.QFont()
                font.setFamily("Franklin Gothic Demi Cond")
                font.setPointSize(12)
                font.setBold(True)
                font.setWeight(75)
                self.Importbttn.setFont(font)
                self.Importbttn.setAutoFillBackground(False)
                self.Importbttn.setStyleSheet("background-color: rgb(170, 170,
255);\n"
"color: rgb(0, 0, 0);\n"
"border-color: rgb(255, 255, 255);\n"
"selection-background-color: rgb(145, 218, 0);")
                self.Importbttn.setAutoDefault(False)
                self.Importbttn.setObjectName("Importbttn")
                MainWindow.setCentralWidget(self.centralwidget)
                self.menubar = QtWidgets.QMenuBar(MainWindow)
                self.menubar.setGeometry(QtCore.QRect(0, 0, 800, 26))
                self.menubar.setObjectName("menubar")
                MainWindow.setMenuBar(self.menubar)
                self.statusbar = QtWidgets.QStatusBar(MainWindow)
                self.statusbar.setObjectName("statusbar")
                MainWindow.setStatusBar(self.statusbar)

                self.Importbttn.clicked.connect(self.getImage)
                self.detectbttn.clicked.connect(self.detectmilk)
                self.detectbttn.setEnabled(False)
                self.clearbttn.clicked.connect(self.clearText)
                self.exitbttn.clicked.connect(MainWindow.close) # type: ignore

                self.retranslateUi(MainWindow)
                QtCore.QMetaObject.connectSlotsByName(MainWindow)

        def retranslateUi(self, MainWindow):
                _translate = QtCore.QCoreApplication.translate
                MainWindow.setWindowTitle(_translate("MainWindow", "Milk Detector"))
                self.title.setText(_translate("MainWindow", "MILK DETECTOR"))
                self.detectbttn.setToolTip(_translate("MainWindow",
```

```
"<html><head/><body><p>Click to detect &quot;MILK&quot; or &quot;UHT&quot; in food
label</p></body></html>"))
                self.detectbttn.setText(_translate("MainWindow", "Detect Milk"))
                self.exitbttn.setToolTip(_translate("MainWindow",
"<html><head/><body><p>Click to exit the application</p></body></html>"))
                self.exitbttn.setText(_translate("MainWindow", "Exit"))
                self.clearbttn.setToolTip(_translate("MainWindow",
"<html><head/><body><p>Click to clear output image</p></body></html>"))
                self.clearbttn.setText(_translate("MainWindow", "Clear"))
                self.Importbttn.setToolTip(_translate("MainWindow",
"<html><head/><body><p>Click to import food label</p></body></html>"))
                self.Importbttn.setText(_translate("MainWindow", "Import Image"))

        def getImage(self):
                options = QtWidgets.QFileDialog.Options()
                options |= QtWidgets.QFileDialog.DontUseNativeDialog
                self.fileName, _ =
QtWidgets.QFileDialog.getOpenFileName(options=options)
                # self.fileName, _ = QtWidgets.QFileDialog.getOpenFileName(self,"Open
a image", "","All Files (*);;Image Files (*.jpg);;Image Files (*.png)",
options=options)
                if self.fileName:
                        #print(self.fileName)
                        pattern = ".(jpg|png|jpeg|bmp|jpe|tiff)$"
                        if re.search(pattern,self.fileName):
                                self.setImage(self.fileName)

        def setImage(self,fileName):
                #self.fileName = cv2.cvtColor(self.fileName, cv2.COLOR_BGR2GRAY)
                self.importimgshow.setPixmap(QPixmap(fileName))
                self.detectbttn.setEnabled(True)

        def detectmilk(self):
                # read non-rotated img
                img_nr = cv2.imread(self.fileName)
                img_nr = cv2.cvtColor(img_nr, cv2.COLOR_RGB2BGR)

                # ----------- Rotate image ---------------

                # change image to grayscale
                gray_ori = cv2.cvtColor(img_nr, cv2.COLOR_BGR2GRAY)
                # Apply canny
                canimg = cv2.Canny(gray_ori, 60, 200)
                # Apply Hough
                lines = cv2.HoughLines(canimg, 1, np.pi/180.0, 250, np.array([]))
                # rotate image
                if np.all(lines != None):
```

```
                        deg = []
                        for line in lines:
                                theta = line[0,1]
                                deg.append(180*theta/np.pi - 90)

                        deg_mode = mode(deg)
                        # rotation angle in degree
                        if deg_mode >= 20 and deg_mode <= 50:
                                img = ndimage.rotate(img_nr, mode(deg) - 45)
                        elif deg_mode <= -20 and deg_mode >= -50:
                                img = ndimage.rotate(img_nr, mode(deg) - 45 + 90)
                        elif deg_mode < -50 and deg_mode >= -90:
                                img = ndimage.rotate(img_nr, mode(deg) + 90)
                        elif deg_mode > 50 and deg_mode <= 90:
                                img = ndimage.rotate(img_nr, mode(deg) - 90)
                        else:
                                img = ndimage.rotate(img_nr, mode(deg))

                # ------------- Develop image -------------

                # convert rotated image to grayscale
                gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

                # dev image
                threshadap = cv2.adaptiveThreshold(gray, 255,
cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY, 201, 9)

                # opening
                kernel = cv2.getStructuringElement(cv2.MORPH_RECT,(3,3))
                opening = cv2.morphologyEx(threshadap, cv2.MORPH_OPEN, kernel)

                # closing
                kernel = cv2.getStructuringElement(cv2.MORPH_RECT,(3,3))
                closing = cv2.morphologyEx(opening, cv2.MORPH_CLOSE, kernel)
                dev = closing

                #-------------- Detect English --------------
                new_img = img.copy()
                p1 = p2 = p3 = p4 = 0

                pytesseract.pytesseract.tesseract_cmd = 'C:/Program
Files/Tesseract-OCR/tesseract.exe'

                # Extract the English word
                detectmilk = 'milk'
                data = pytesseract.image_to_data(img,
output_type=pytesseract.Output.DICT)
```

```
                for i, word in enumerate(data["text"]):
                        data["text"][i] = re.sub(',|:', '', word)
                wordrecog_en = [ i for i, text in enumerate(data["text"]) if
text.lower() == detectmilk]

                # try enhaced image it real image does not work
                if wordrecog_en == 0:
                        detectmilk = 'milk'
                        data = pytesseract.image_to_data(dev,
output_type=pytesseract.Output.DICT)
                        for i, word in enumerate(data["text"]):
                                data["text"][i] = re.sub(',|:', '', word)
                        wordrecog_en = [ i for i, text in enumerate(data["text"]) if
text.lower() == detectmilk]

                for word in wordrecog_en:
                        # get top, left position and width, height of extracted word
                        w = data["width"][word]
                        h = data["height"][word]
                        l = data["left"][word]
                        t = data["top"][word]
                        # define 4 coordinates of box
                        x1 = (l, t)
                        x2 = (w + l, t)
                        x3 = (w + l, h + t)
                        x4 = (l, h + t)
                        # draw 4 lines to create the box
                        new_img = cv2.line(new_img, x1, x2, color=(255, 0, 0),
thickness=5)
                        new_img = cv2.line(new_img, x2, x3, color=(255, 0, 0),
thickness=5)
                        new_img = cv2.line(new_img, x3, x4, color=(255, 0, 0),
thickness=5)
                        new_img = cv2.line(new_img, x4, x1, color=(255, 0, 0),
thickness=5)

                # ---------- Detect Thai --------------

                # reuse parameters
                x1 = x2 = x3 = x4 = 0
                h, w = gray.shape
                pytesseract.pytesseract.tesseract_cmd = 'C:/Program
Files/Tesseract-OCR/tesseract.exe'
                # Extract Thai string
                data_th = pytesseract.image_to_data(img,
output_type=pytesseract.Output.DICT, lang="tha")
```

```
                # detect both 'น' that follows by 'ม' and 'นม'
                wordrecog_m = [ i for i, text in enumerate(data_th['text']) if text ==
'น']
                wordrecog_k = [ i for i, text in enumerate(data_th['text']) if text ==
'ม']
                wordrecog_th = []
                wordrecog_th = [ i for i, text in enumerate(data_th['text']) if text
== 'นม']
                for i,m in enumerate(wordrecog_m):
                        for j,k in enumerate(wordrecog_k):
                                if k-m == 1:
                                        wordrecog_th.append(m)

                # try enhaced image if real image does not work
                if wordrecog_th == 0:
                        data_th = pytesseract.image_to_data(dev,
output_type=pytesseract.Output.DICT, lang="tha")
                        wordrecog_m = [ i for i, text in enumerate(data_th['text']) if
text == 'น']
                        wordrecog_k = [ i for i, text in enumerate(data_th['text']) if
text == 'ม']
                        wordrecog_th = []
                        wordrecog_th = [ i for i, text in enumerate(data_th['text'])
if text == 'นม']
                        for i,m in enumerate(wordrecog_m):
                                for j,k in enumerate(wordrecog_k):
                                        if k-m == 1:
                                                wordrecog_th.append(m)

                # Draw X if the snack contains milk
                if len(wordrecog_th) + len(wordrecog_en) != 0:
                        new_img = cv2.line(new_img, (0,0), (w,h), color=(255, 0, 0),
thickness=20)
                        new_img = cv2.line(new_img, (0,h), (w,0), color=(255, 0, 0),
thickness=20)
                # Draw ✓ if the snack does not contain milk
                else:
                        new_img = cv2.line(new_img, (0,np.int(h*3/4)),
(np.int(w/6),h), color=(0, 255, 0), thickness=20)
                        new_img = cv2.line(new_img, (np.int(w/6),h), (w,0), color=(0,
255, 0), thickness=20)

                # Draw box around detected word
                for word in wordrecog_th:
                        # get top, left position and width, height of extracted word
                        w = data_th["width"][word] + data_th["width"][word+1]
                        h = data_th["height"][word]
```

```
                        l = data_th["left"][word]
                        t = data_th["top"][word]
                        # define 4 coordinates of the box
                        x1 = (l, t)
                        x2 = (w + l, t)
                        x3 = (w + l, t + h)
                        x4 = (l, h + t)
                        # draw 4 lines to create the box
                        new_img = cv2.line(new_img, x1, x2, color=(255, 0, 0),
11
thickness=5)
                        new_img = cv2.line(new_img, x2, x3, color=(255, 0, 0),
thickness=5)
                        new_img = cv2.line(new_img, x3, x4, color=(255, 0, 0),
thickness=5)
                        new_img = cv2.line(new_img, x4, x1, color=(255, 0, 0),
thickness=5)
                plt.imsave("detectedmilk.jpg", new_img)

                h1, w1 = new_img.shape[:2]
                final_img = QImage(new_img, w1, h1, QImage.Format_Grayscale16)
                                                    30
                pixmap = QPixmap.fromImage(final_img)
                self.detectimgshow.setPixmap(pixmap)
                self.detectbttn.setEnabled(True)

                #Create window to show output in text whether it contains milk or not
                messagebox = QMessageBox()
                #if it has milk as an ingredient, show "CONTAIN MILK"
                if len(wordrecog_th) + len(wordrecog_en) != 0:
                        messagebox.setText("CONTAIN MILK")
                #if it not has milk as an ingredient, show "NOT CONTAIN MILK"
                else:
                        messagebox.setText("NOT CONTAIN MILK")
                messagebox.setWindowTitle("SHOW OUTPUT")
                messagebox.setStandardButtons(QMessageBox.Close)
                x = messagebox.exec_()

        def clearText(self):
                self.detectimgshow.clear()

if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec_())
```

Reference

(n.d.). *QImage Class*. Qt documentation. https://doc.qt.io/qt-6/qimage.html

docPhil99 (2022). *opencv_qt_label.md*. Github. https://gist.github.com/docPhil99/ca4da12c9d6f29b9cea137b617c7b8b1

Nederkoorn, C. (n.d.). *Top 10 Python GUI Frameworks Compared*. ActiveState. https://www.activestate.com/blog/top-10-python-gui-frameworks-compared/#:~:text=1%E2%80%93PyQt5,about%20any%20kind%20of%20application

Siahaan, V. (2019). *Image Color Space With OpenCV and PyQt*. Blogger. http://viviansiahaan.blogspot.com/2019/04/image-processing-with-opencv-and-pyqt.html

Sudsa-ard, K., Kijboonchoo, K., Chavasit, V., Chaunchaiyakul, R., Nio, A. Q., & Lee, J. K. (2014). Lactose-free milk prolonged endurance capacity in lactose intolerant Asian males. *Journal of the International Society of Sports Nutrition*, *11*(1). https://doi.org/10.1186/s12970-014-0049-4

FinalProject_MilkDetection

ORIGINALITY REPORT

12%	11%	4%	8%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Ain Shams University Student Paper	1%
2	stackoverflow.com Internet Source	1%
3	www.programcreek.com Internet Source	1%
4	discuss.pytorch.org Internet Source	1%
5	huggingface.co Internet Source	1%
6	forum.qt.io Internet Source	1%
7	Submitted to Ho Chi Minh University of Technology and Education Student Paper	1%
8	Submitted to Middlesex University Student Paper	1%
9	nanonets.com Internet Source	1%

10	dokumen.pub Internet Source	<1%
11	www.arborite.com Internet Source	<1%
12	dev-gang.ru Internet Source	<1%
13	www.opcito.com Internet Source	<1%
14	Submitted to Visvesvaraya Technological University, Belagavi Student Paper	<1%
15	www.murtazahassan.com Internet Source	<1%
16	Submitted to Florida Virtual School Student Paper	<1%
17	forum.facepunch.com Internet Source	<1%
18	www.coder.work Internet Source	<1%
19	Submitted to University of Westminster Student Paper	<1%
20	kandi.openweaver.com Internet Source	<1%

21	"Intelligent Computing", Springer Science and Business Media LLC, 2022 Publication	<1 %
22	I Gede Susrama Mas Diyasa, Eva Y Puspaningrum, Moch. Hatta, Ariyono Setiawan. "New Method For Classification Of Spermatozoa Morphology Abnormalities Based On Macroscopic Video Of Human Semen", 2019 International Seminar on Application for Technology of Information and Communication (iSemantic), 2019 Publication	<1 %
23	Submitted to Miami University of Ohio Student Paper	<1 %
24	ijarcce.com Internet Source	<1 %
25	www.instructables.com Internet Source	<1 %
26	Submitted to University College London Student Paper	<1 %
27	Submitted to Mahidol University Student Paper	<1 %
28	Submitted to Singapore Institute of Technology Student Paper	<1 %
	www.13download.com	

29	Internet Source	<1%
30	viviansiahaan.blogspot.com Internet Source	<1%
31	Joshua M. Willman. "Beginning PyQt", Springer Science and Business Media LLC, 2020 Publication	<1%
32	Submitted to Asia Pacific International College Student Paper	<1%

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		

FinalProject_MilkDetection

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16

PAGE 17

PAGE 18

PAGE 19

PAGE 20

PAGE 21

PAGE 22

PAGE 23

PAGE 24

PAGE 25

PAGE 26

PAGE 27

PAGE 28

PAGE 29

PAGE 30

PAGE 31

PAGE 32

PAGE 33

PAGE 34

PAGE 35

PAGE 36

PAGE 37

PAGE 38

PAGE 39

PAGE 40

PAGE 41

PAGE 42

PAGE 43

PAGE 44

PAGE 45

PAGE 46

PAGE 47

PAGE 48

PAGE 49

PAGE 50

PAGE 51