Assignment Report

Group members
Vũ Minh Hiếu - 16022405
Nguyễn Thanh Tùng - 16020063
Võ Lê Minh Tâm - 16020279
Nguyễn Thị Linh - 16022409

I. Introduction

This is the report for the midterm assignment of the Winter 2019 Image Processing class (1920I_INT3404 20) taught by Dr. Ta Viet Cuong. In the rest of this report, we will first summarize the requirements of the assignment. Next, we will briefly describe the data we use during the course of this project. After that, we will discuss about possible approaches, followed by the implementation details of our chosen approach. Lastly, we will give out some ideas to further improve our results.

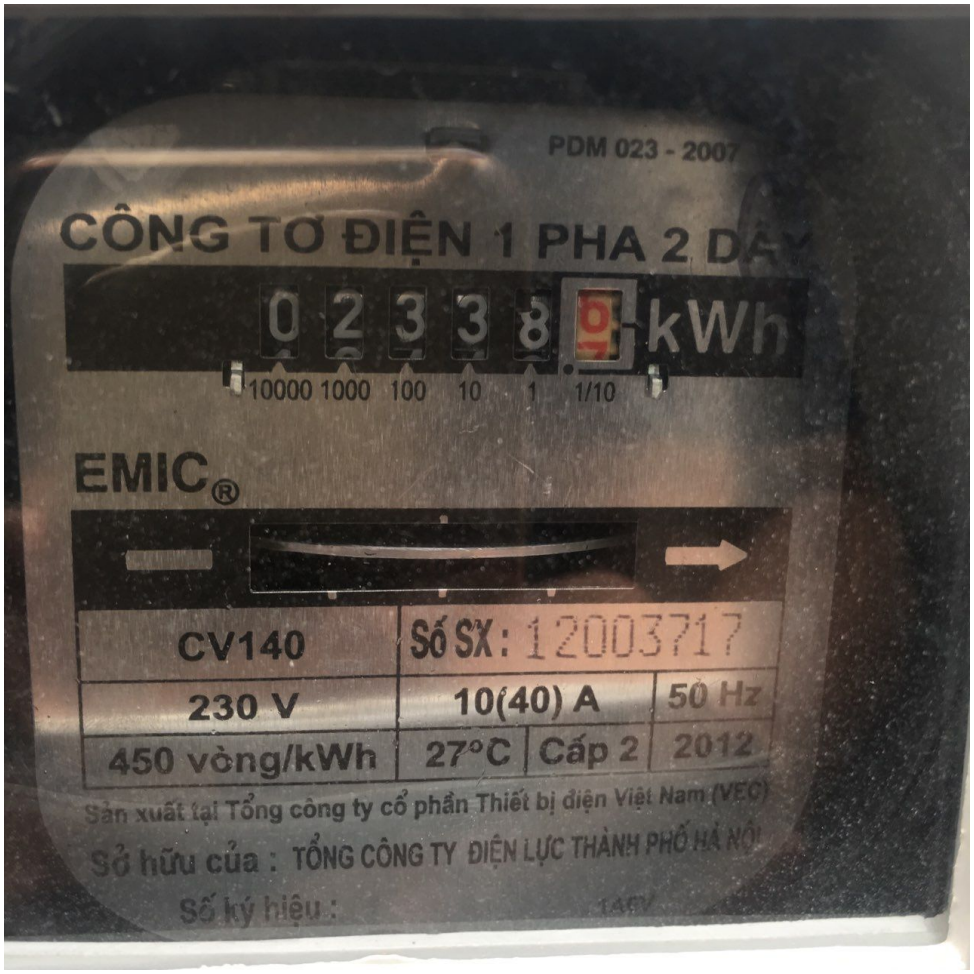
II. Problem statement

The assignment consists of two tasks:

- In this task 1, images of the dial of the electric meter are given. The requirement is to read the power consumption amount shown in the image. There are 6 number cells, but the last number are not required. Furthermore, if a cell is hanging between 2 digits, that cell is not required as well.

A sample image for task 1

- In task 2, images of the entire electric meter are given. Other requirements are the same as task 1.



A sample image for task 2

In addition, the size of the solution must be no greater than 10MB, and the processing time for one image is no longer than 5 seconds not including start up time.

III. Dataset

Along with the dataset given by the lecturer, we attempted to increase the amount of data with:

- Manually collect 50 more full-image data which then be hand-cropped to use for task 1.
- Download about 20 images from the internet. Most of the images from the internet are images of brand new electric meter and have the dial at “00000”,

which we considered not helpful to have a lot of, so we decided not to try to invest our time to gather more.
- The SCUT-WMN dataset, more detail can be found here.
- The SVHN dataset, more detail can be found here.

All data that do not come with annotations are labeled by hand. The label consist of bounding box coordinates and the value shown in those bounding boxes.

IV. Possible approaches

1. Pure image processing approach

   This is the most light-weight approach as it involves leveraging only image processing algorithms, but is also the hardest one. This approach requires a lot of parameter tuning and is prone to overfitting. Considering the diversity of the input data, it will be really hard to find a solution that works reasonably well.

   We quickly decided that this approach is too risky and believed that there is an approach superior to this in every way, so this approach was not experienced.

2. Pure machine learning approach

   The problem can be tackled by using an object detection model to localize the digits and then classify them using a classification model. Another way is to use object detection only for localizing the dial in a full-image, and then use a Convolution Recurrent Neural Network (CRNN) with Connectionist Temporal Classification (CTC) loss to read all 5 digits is one pass. With the recent advancement of Deep learning, this approach could achieve really good results. However, the lack of training data and the limitation of solution file size makes this approach inapplicable.

   We tried to train a CRNN-CTC model using the data given from the lecturer and the data we collected. Due to the small quantity but large variance of the data, the model quickly became overfitted and failed to generalize.

Attempting to tackle that, we then train the model with the colour-inverted version of the SCUT-WMN dataset and cross-validation on half of the full-image data. We then found out that images in the SCUT-WMN are very similar to each other but differ greatly from the targeted data. The fact that the training data and the validation/test data come from two far-apart distribution makes the model unable to learn anything.

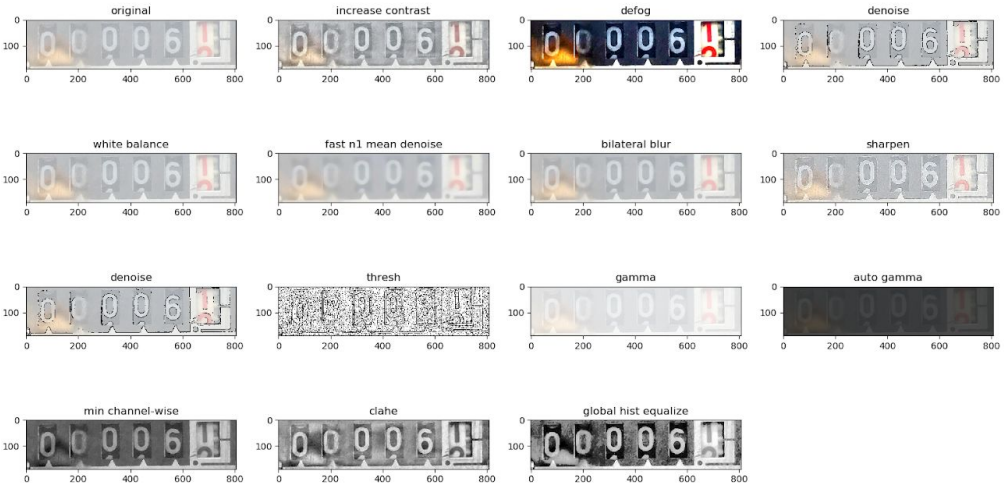
3. Combining image processing and machine learning

   Each of the two aforementioned approaches has its own advantages and disadvantages. In order to balance between different requirement criteria, we used a combination of image processing algorithms and machine learning models. This is the approach chosen for the final submission, the implementation details are described in the next section.

V. Implementation details

1. Preprocessing operations

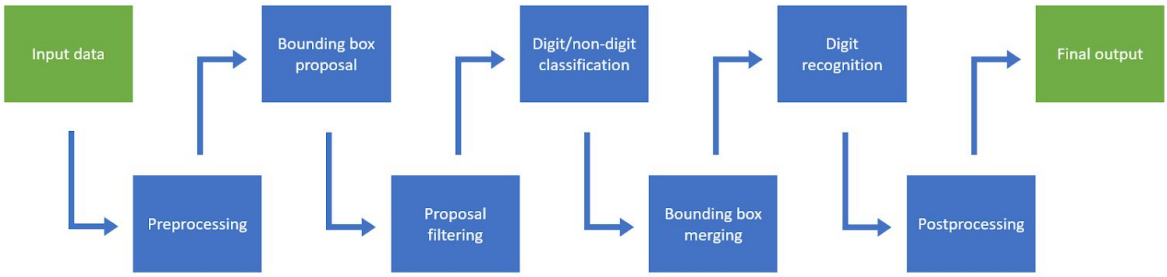
   Cleaning and preprocessing data is a crucial step in every data-driven project. Right from the start, we implemented several image processing operations and keep on adding throughout the course of the project. These operations are implemented in a single script so that they can be used on our demand.



Implemented preprocessing functions

2. Task 1

Our strategy on tackling this task is first generate digit bounding box proposals, then filter the proposal, and lastly, perform classification on the remaining proposals. Noted that each step has its own preprocessing subroutine. The overall pipeline is visualized below:

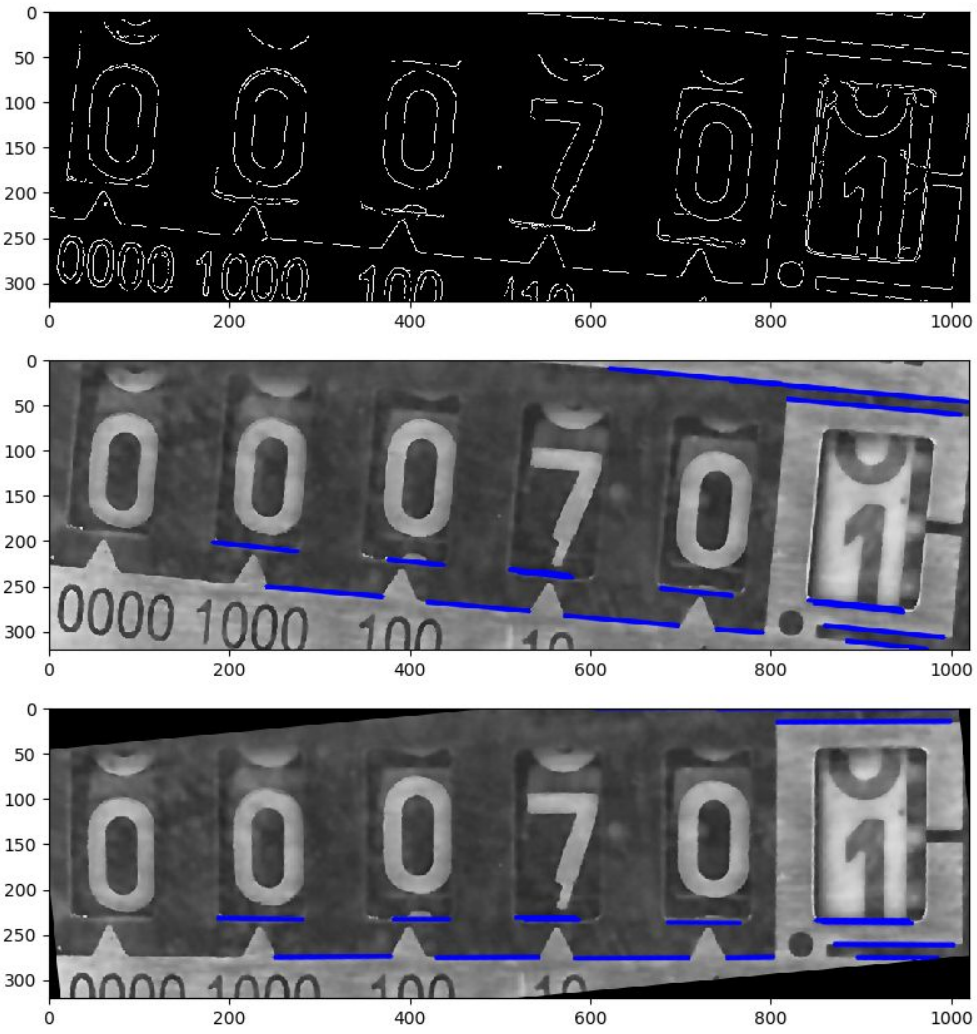


Pipeline overview

Our strategy is to utilize image processing anywhere possible and only consider using a machine learning if the task is easy enough and can benefit from a small pretrained.

2.1. Preprocessing

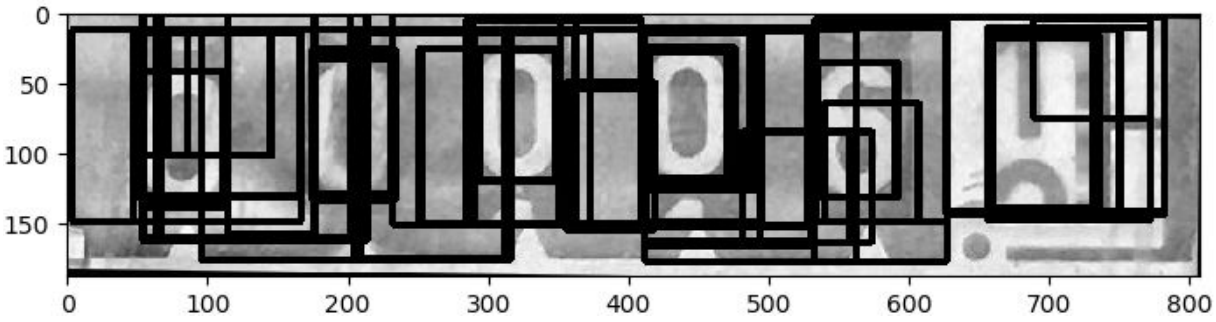
Since every processing step has its own preprocessing subroutine, this step perform only the general preprocessing operations. And in this case, we perform only rotation. The rotation routine starts with smoothing the image, for this we chose to use bilateral filtering due to its edge-preserving nature. Following that, Contrast Limited Adaptive Histogram Equalization (CLAHE) and then Canny edge detection is applied, giving us an image containing only edges. Next, we used Hough transform on the edge image to detect horizontal or nearly-horizontal lines, then calculated the average angle of those lines. Finally, the original image is rotated to horizontal using the calculated angle if the angle is less than 10 degrees.



From top to bottom: the edge image; horizontal and nearly horizontal lines found in the image (denoted in blue); the rotated image.

2.2. Bounding box proposal

In this step, the input image is preprocessed using CLAHE. Bounding box proposals are generated by processing the preprocessed image using Maximally Stable Extremal Regions (MSER) with Oriented FAST and Rotated BRIEF (ORB) features. The result of this step can be visualized below:



Bounding box proposals

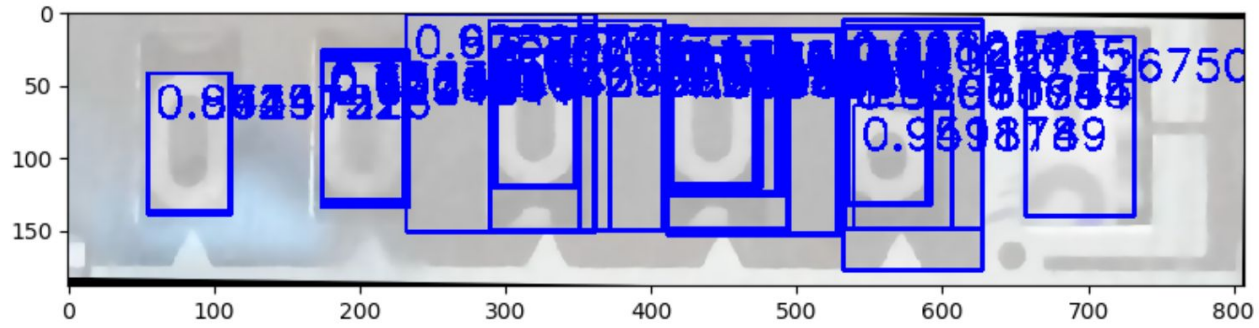
2.3. Proposal filtering

The bounding box proposals are first filtered by its size. We discard a bounding box if it satisfies one of the following:

- Its width is smaller than $\frac{1}{15}$ the width of the image
- Its height is smaller than $\frac{1}{5}$ the height of the image
- Its area is smaller than $\frac{1}{32}$ the area of the image
- Its area is bigger than $\frac{1}{5}$ the area of the image
- Its $\frac{width}{height}$ ratio is greater than 1.5
- Its $\frac{height}{width}$ ratio is greater than 3

2.4. Digit/non-digit classification

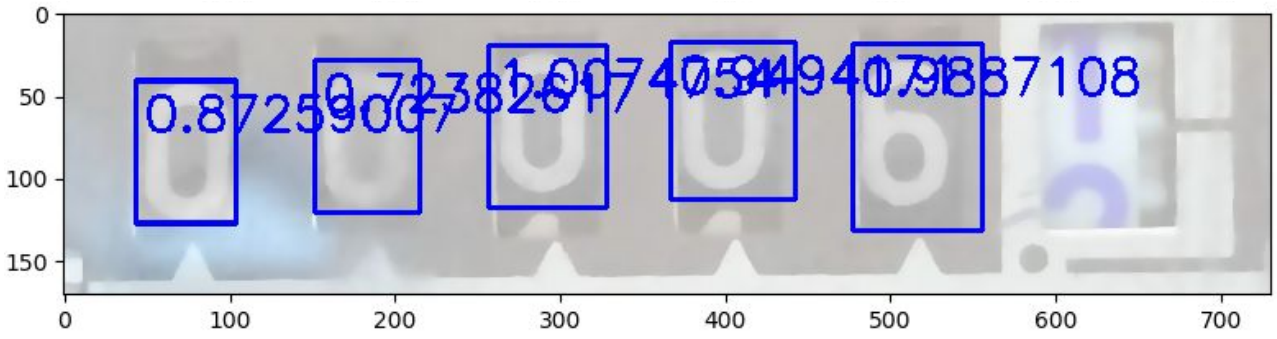
For the remaining proposals, we used a VGG-like model with only 2 VGG blocks and a 1024-neuron Fully Connected layer with 0.5 dropout before the output layer to predict the digit-ness probability score. The input image is resized to 32x32, converted to grayscale, then the mean pixel value of the training data is subtracted from it before feeding into the model. The model is trained with sample generated by MSER-ORB on the SVHN dataset, a sample is considered to be positive if it overlaps more than 60% with a ground-truth box, and is considered to be negative if it overlaps less than 5% with a ground-truth box. Using the result of this model, we discarded all proposals having digit-ness score equal or smaller than 0.4.



Remaining bounding boxes after being filtered

2.5. Bounding box merging

After the previous step, there are still many bounding boxes remain. However, they seem to be appear in cluster surrounding the digits. So we merge all boxes that have the Intersection over Union (IoU) value greater than 0.2 in a weighted fashion, using the digit-ness score produced in the previous step as weight.



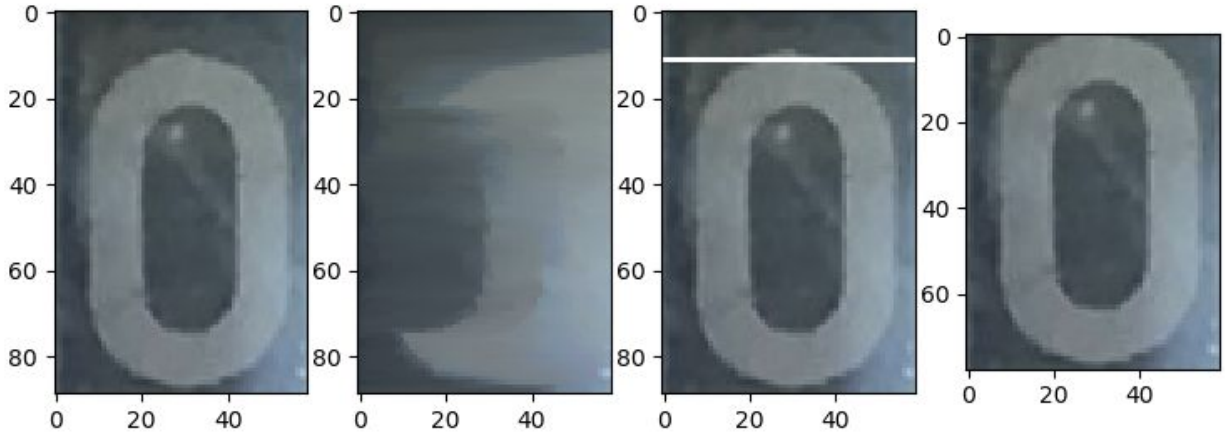
Merged bounding boxes

2.6. Digit detection

For the digit detector, we again used a VGG-like model with only 2 VGG blocks and a 512-neuron Fully Connected layer with 0.5 dropout before the output layer. Similar to the digit/non-digit classifier, the input image is resized to 32x32, converted to grayscale, then the mean pixel value of the training data is subtracted from it before feeding into the model. The model is trained with sample generated by MSER-ORB on the SVHN dataset, a sample is considered to be positive if it overlaps more than 60% with a ground-truth

box, and is considered to be negative if it overlaps less than 5% with a ground-truth box.

However, since the input is resized to 32x32, the change in the dimension ratio might distort the digit if input image is too “tall” or too “fat”. So before resizing, we attempted to trim away any redundant region in the image.

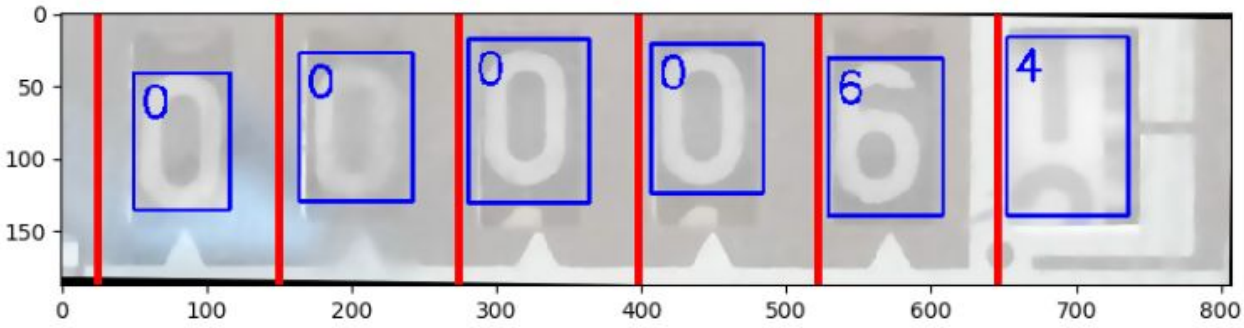


An example of trimming redundant region at the top of the image. From left to right: the input image in grayscale; the image after sorting pixel value in every row; the white line denoted where the image will be cut off calculated using the sorted image; the trimmed image.

We then used the recognizer to classify each proposal into one of 10 classes corresponding to 10 digits.

2.7. Postprocessing

Now that we have the recognition results of all proposals, there are still some problems: while we know for a fact that there are 5 digits needing to be recognized, what if our program return more or less than 5; what if there are multiple proposals overlapping with others or stacking vertically, in that case which one do we keep ? In order to solve this, we tried to detect the last digit, which appears in red, and used its location along with the width of all bounding box proposals to approximately horizontally divide the image into 6 sections corresponding to 6 digits of the final output. For every section, the final value of that section is the value of the largest proposal contained in that section.



Divide the image into 6 sections and produce the final output.

3. Task 2

3.1. Objective: Cropping out numbers box from the overview image for Task1.
Input image:

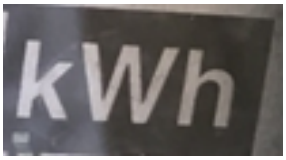
Expected output image:

3.2. Assess the problem:

To detect the position of number box, we should rely on its characteristic properties or that of adjacent components. There are many components capable of using, such as:

- The long black box that contain main digits:

- The ‘kWh’, which is quite unique:



- The last digit with red color:

However, form of the long black box is easily confused with the this one:

And in a few types of electric meter, the last digit (1/10 counting digit) is non-red:

or there might be many red components in the input image:

It seems, the ‘kWh’ is the best component to relate to detecting position of the number box.

After determined the unique component’s position, we have to relate it to the number box’s. The ‘kWh’ and the red last digit is right-side of number box, meanwhile the long black box contains it in the middle.

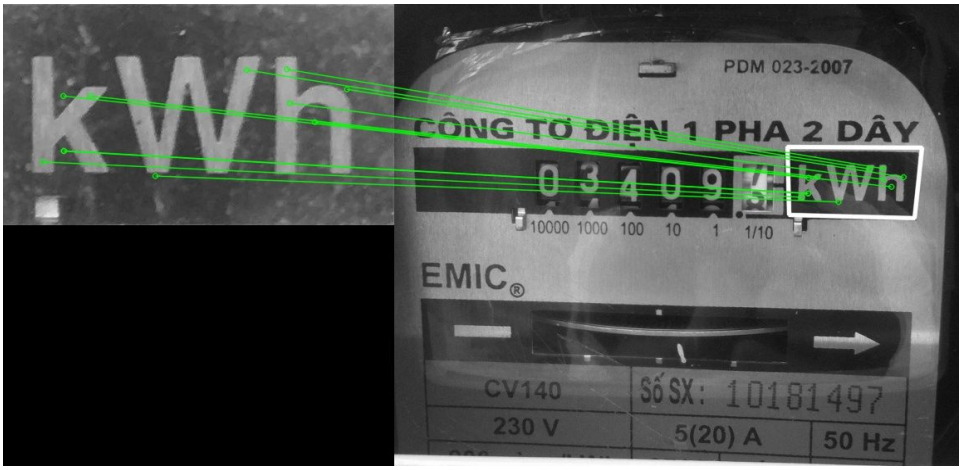
3.3. Used methods:

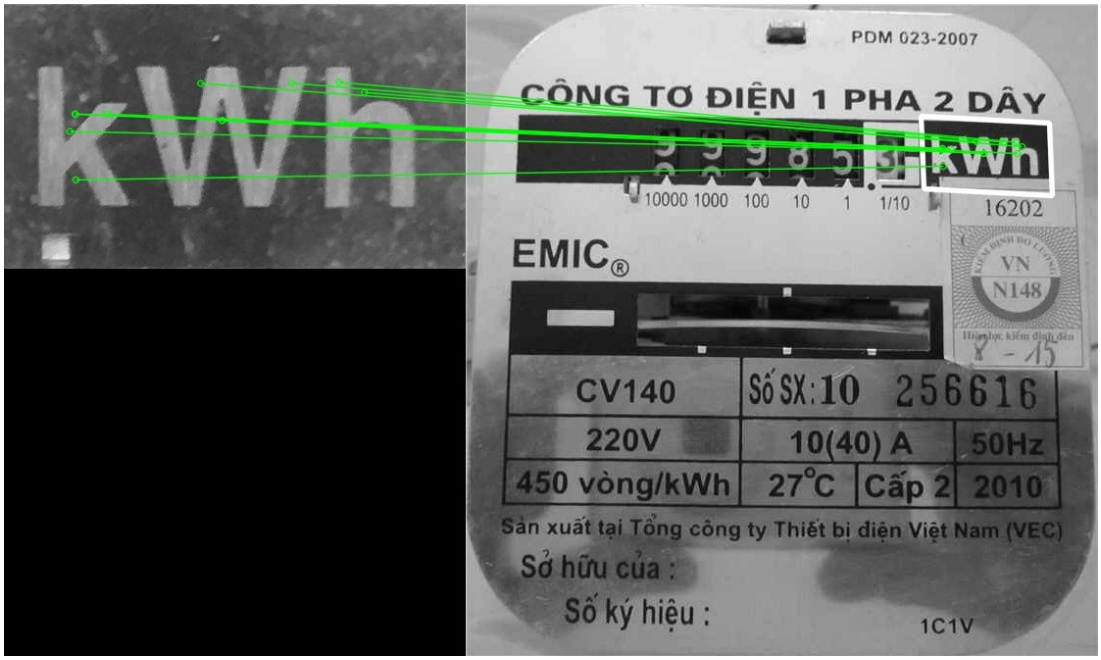
a. Feature matching for the “kWh”

For feature matching, we used a query image, which is a “kWh” image, find some feature points in it, we took another train image, which is a full image used for task 2, find the features in that image too and then collect the best matches among them. In short, we find locations of some parts of an object in another cluttered image. This information is sufficient to find the object exactly on the train image.

First, we cut out the most beautiful with good resolution ‘kWh’ image, then implemented Scale Invariant Feature Transform (SIFT) to extract keypoints and compute its descriptors.

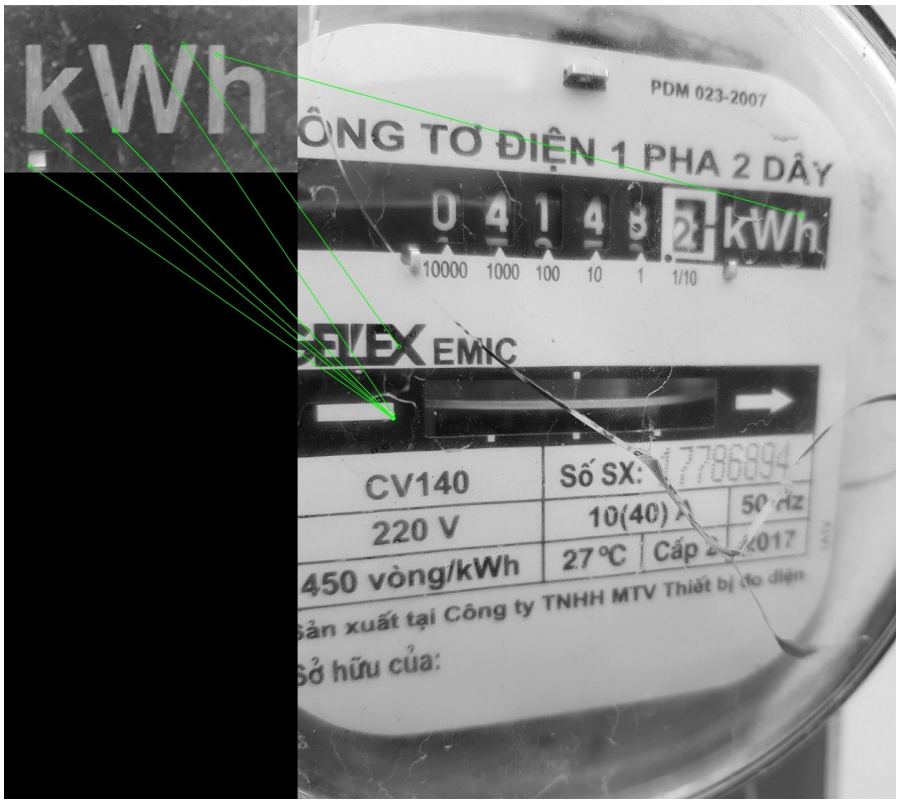
To match its key points to other overview images, Fast Library for Approximate Nearest Neighbors (FLANN) algorithm was used to build a keypoint matcher and it worked quite good.





even the original image was rotated.

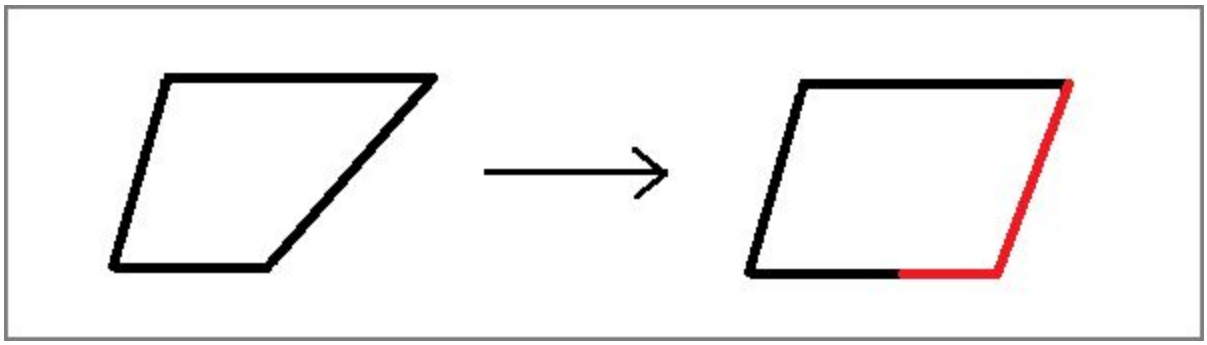
But sometimes, it made a mistake too.



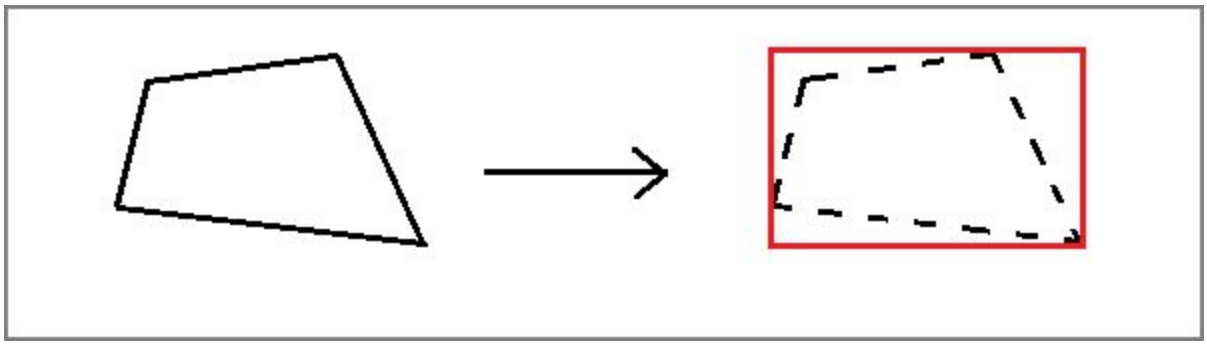
or detected quadrilateral box of ‘kWh’ was not a rectangle or a parallelogram.

With no-detected-kWh images, we would use other methods.
For more accurate relating to number box position, we used a few geometric methods to transform the quadrilateral box to parallelogram.

- If the box had 2 pairs of parallel edges, it’s already a parallelogram
- If the box had 1 pair of parallel edges, the vertex that showed the most obtuse angle should be translated along vector of parallel edges.



- If the box has no pair of parallel edges, it would be transform to a rectangle that contains it.



We decided the number box was left-side of the ‘kWh’ box, depended on the rotated angle of overview image (got when matched keypoints). Number box’s length was 2.2 times of ‘kWh’ box, and width was 1.2 times, just to be sure, and then crop it out for task 1.

However, since this approach uses the SIFT algorithm, which is a patented algorithm and is not available in openCV version 4.x and above, we cannot use this approach in our submission.

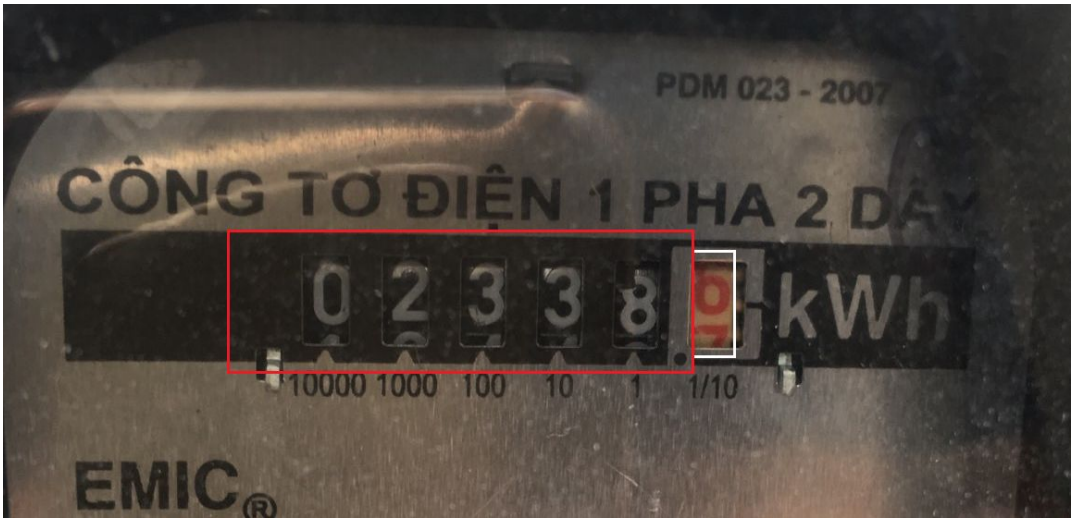
b. Detect the red last digit by color

The objective is to find out the most red components in the input image.
Formula of red-ness of any pixel is:

Red-ness = R - 0.5(G+B)

Set threshold is top 99.9% of red-ness range on whole image, much red pixels was set to 255, others was 0.

The rest was quite easy, find out contour and decide which was the correct one. And the number box was left-side of the last digit box. Number box's length was 12 times of its, and width was 1.2 times. Then crop it for task 1.



3.4. Combining with task 1

The image cropped by the method mentioned in the previous section sometimes contain a lot of redundant region on the left of the dial such as this one:

This could introduce problems for the post processing step of task 1. We solve this by running task 1 on the result of task 2 twice. In the first run we retrieve all the bounding boxes and use the width of those boxes to trim away unnecessary parts in the image. We use the result of the second run as the final result.

VI. Possible improvements

There are several improvements can be done to further improve our results, which are, but not limited to:

- Resize the input image of task 1 to a predefined height for better generalization
- Using openCV 3.x to be able to use SIFT for task 2.
- Collect more data to train the models with data more similar to the targeted data.