

**综合实践项目报告**

**Practical Project Report**

课题名称Topic：

Machine Vision-based Intelligent Recognition of Building Air Conditioning Interface



Figure 1: Illustration of one project’s result

教师姓名Supervisor：**Helin Gong**

学生姓名Student：**Babic Marko**学号Student ID：**J12426099007**

专业名称Major：**Computer Science - Information Engineer**

学院(系)：**巴黎卓越工程师学院SPEIT**

**填表说明INSTRUCTIONS**

1. 每位学生应在指导教师的指导下认真、实事求是地填写各项内容。文字表达要明确、严谨，语句通顺，条理清晰。外来语要同时用原文和中文表达，第一次出现的缩写词，须注出全称。

Under the guidance of the supervisor, each student should fill in all the information carefully and realistically. The literal expression shall be clear, rigorous, fluent and logical. Foreign words should be expressed in both original language and Chinese/English at the same time. Abbreviations appearing the first time shall be given the full name.

1. 要求与研究有关的主要参考文献阅读数量不少于10篇，其中外文资料应占一定比例。

Critical review of literatures shall be carried out before research, and at least 10 subject-related references shall be reviewed, including some foreign references.

1. 项目报告要求用英语、法语或中文撰写（由指导老师决定）。需满足至少4000中文字（英语/法语按比例折算为16000外文字符）。

The report could be written in English, French or Chinese (on the supervisor’s suggestion), with about at least 4000 Chinese characters (or at least 16000 English/French foreign digits).

1. 请用中文宋体小四号字体/外文使用Times New Roman字体12号字填写，并用A4纸打印，于左侧装订成册。

As for font and type size, use Times New Roman 12 for French, and SimSun 12 for Chinese. Print on A4 paper and bind on the left side.

1. 该表填写完毕后，须请指导教师审核，并签署意见。

After completing, please ask the supervisor to review and sign the opinions.

1. 此报告将作为巴黎卓越工程师学院综合实践项目考核材料之一。

This report is one of the main materials for the evaluation of Practical Project.

**Table of Contents**

[课题名称Topic 4](#_Toc182929632)

[Purpose of the project 4](#_Toc182929633)

[Significance of the project 4](#_Toc182929634)

[Overview on the actual research situation 4](#_Toc182929635)

[课题研究内容Research content： 4](#_Toc182929636)

[研究方法和研究思路（技术路线）Research methods and clues： 4](#_Toc182929637)

[研究结果Research results： 4](#_Toc182929638)

[研究进度安排Research schedule： 5](#_Toc182929639)

[参考文献References： 6](#_Toc182929640)

[指导教师意见Supervisor’s opinion 8](#_Toc182929641)

[学院（系）意见Institute's opinions 8](#_Toc182929642)

## 课题名称Topic

### Purpose of the project

The purpose of this project is to establish a programmed digital recognition model capable of reading the digits from an air-conditioner (annotated **AC = 空調設備**) interface. More precisely, from our own pictures taken by a smartphone.

In fact, we took two hundreds of pictures with a smartphone of AC’s interfaces in the Shanghai Jiao Tong University (annotated **SJTU =上海交通大学**) campus. Our final goal is to be able to read as many pictures as possible even if they have visual problem with highest accuracy and robustness. Indeed, some images can be taken with different angle, with different brightness, with reflections, camera settings, perspective distortions, glare obscuring digits, digit occlusion by dust or other objects, or resolution and scaling issues.

### Significance of the project

The main significance of our project is to become a relevant tool for building maintenance. For instance, a building containing different AC in many rooms.

Our project can be used by people that manages the temperatures of each room.

Indeed, a picture could be taken from an AC so as to put it direct in a database. Then this database could be displayed on a monitoring tool that will alert the user when some rooms have abnormal temperatures. Especially for small buildings, rather than investing in linking all AC to a database online or in an internal network. The pictures will help to check regularly the temperatures and regroup it in a cheaper and in a simpler way.

Furthermore, if an abnormal temperature is detected, an action will be taken so as to prevent energy loss, heat loss (that could make money loss) or to get the room more comfortable.

### Overview on the actual research situation

## 课题研究内容Research content：

### Taking a sample of pictures

Here is the first main task, we need to take certain amount of pictures in order to experiment our model.

In our project, these are the 2 main criteria:

* Taking a certain amount of pictures around 200.
* Take our **own** photos without using a bank of pictures made by someone else already prepared

To take our photos, we used our own smartphone.

More precisely, a **Motorola G84 5G**.

To get that much of pictures, we visited different schools on the SJTU campus where we found different type of AC interface from different brands.

When we found an AC, we took 4-5 photos of it. One photo in front of the AC, 2-3 with different angles, and one like the first one but zoomed out.

Our photos are stored in my personal computer. Moreover, I made a copy of them on a SD card and on a cloud drive in any case.

An important point to notice is that the image can be blurred or can get quality issues for the reason that the smartphone and the photographer are not perfect.

Figure RC1: Picture of me taking a photo of AC

The whole task went as planned, we managed to get **205 pictures.** The space used on the computer is 507MB.

### Establishing a first model

To establish a first model recognition, we had first to select a programming language. The supervisor told me that it could be done with the most common language like JAVA, C, C++ or Python. Consequently, I decided to use Python for different reasons:

* **Versatile, easy to read, learn, and write**
* **High-level language** that abstracts low-level details, making it more user-friendly.
* **Ideal for prototypes:** Python’s concise syntax allows to prototype applications quickly with less code.

**Interpreted language:** Python is interpreted, which allows for easier debugging and code development.

* **Extensive support libraries** making it suitable for scientific and data-related applications.
* **Open source and large active community base**
* **Interpreted language**: which allows easier debugging and code development.

[RCQ1]

However, it is important to notice that Python is not the best language in term of performance. For instance, this language is really far behind the other main languages in term of energy consumption, execution time and memory consumption.

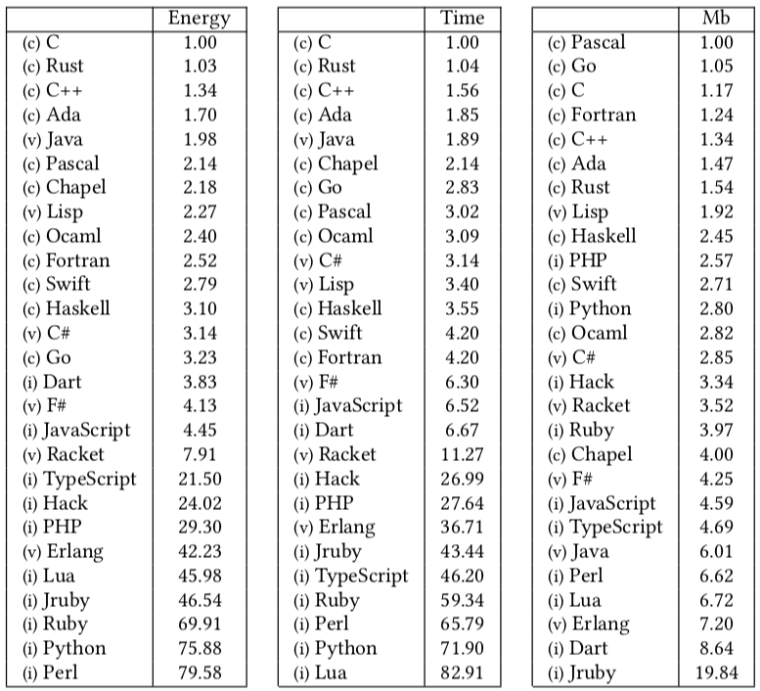


Figure RC2: performance criteria between different programming languages

To use python, we decided to take **Pycharm** (developed by JetBrains) as our python Integrated Development Environment(IDE) with a python version 3.12.

Moreover, we started to search which Python libraries would help to read characters on pictures:

* **pytesseract**: A Python wrapper for Tesseract-OCR Engine, which is highly effective for digit recognition.
* **EasyOCR**: A user-friendly library that can recognize digits and other characters from images with ease.
* **PaddleOCR**: A Python library developed by Baidu that is capable of recognizing digits and text from images with high accuracy.

After trying each one, we decided to use **EasyOCR**. This library was easier to test and to install than the other. Another reason comes from the fact that it does not necessity a heavy syntax to detect text on an image.

While testing this tool, we noticed that the process of reading is longer on bigger image. In addition, it reads every characters on the screen, that means that the number are not isolated.

For this reason, it is necessary to preprocess a picture before reading it. Indeed, cropping it will reduce the time of reading and reduce the risk of reading digits outside the AC interface. But also finding a filter that would avoid reading other elements of the screen than the temperature would help to create our first model.

Here are the different libraries related to image processing we started to use to establish this first model:

* **cv2**: An open-source computer vision library that provides a wide range of image and video processing functions.
* **Numpy**: A fundamental package for scientific computing with Python, providing support for arrays, matrices, and a variety of mathematical functions.
* **Matplotlib**: A plotting library used for creating static, animated, and interactive visualizations in Python.
* **Imutils**: a series of convenience functions to make basic image processing functions such as resizing, rotating, and converting between different image formats.

Here are the different steps of our first model, click [here](https://github.com/Bakame1/digitAC/blob/81526763e19f0f4adbe9b899fab6dbb65d0d4dcd/digitACv1.py#L64) to see the related code:

1. Load an image from our files
2. Pre-process the image with a gray filter, a blur filter then a cropping
3. Initialize EasyOCR reader to read text on our pre-processed image using GPU and English language
4. Manage the reading of the Celsius symbol. We suppose that AC should not get a temperature higher than a 99°C. Therefore, if we detect 3 digits read, we select only the first ones.
5. Draw bounding boxes on detected digit and the digits detected.
6. Return a string containing the digit recognized.

When we use the **easyocr.Reader** to read text from an image, the **readtext** method returns a structured output that contains detailed information about each detected text element within the image. Here's a more detailed breakdown of the structure returned:

1. **List of Dictionaries**: It is a list where each element represents a separate text region detected in the image.
2. **Dictionary Components**:

* **'text'**: This key holds the actual text string detected within the bounding box.
* **'bbox'**: This key contains the coordinates of the bounding box that encloses the detected text. The coordinates are usually in the format of **(x, y, width, height),** where x and y are the coordinates of the top-left corner of the bounding box, and width and height are the dimensions of the box.
* **'confidence'**: This key provides a confidence score indicating how certain the OCR engine is about the accuracy of the detected text. The score is a **value between 0 and 1**, with higher values indicating higher confidence.

{"text": "Detected Text String",

"bbox": [x, y, width, height],

"confidence": confidence\_value}

Nevertheless, it did not work on other images. We can clearly on the figure 401 where the image processing failed. Making then impossible to read digits. The variable containing the information on the them was consequently empty and we did not create exception for that case. Finally led to an error.

That is reason why we need to improve this model in order to read at least 5 images.

### Establishing a second model

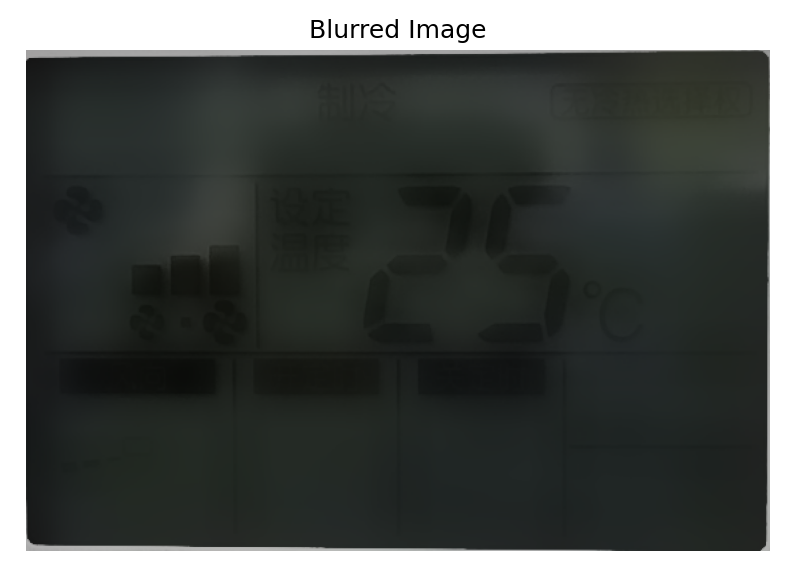
The main issues that came from the first model were related to the pre-processing aspect. Thus, to improve this, we separated the cropping part in another file name [**crop.py**](https://github.com/Bakame1/digitAC/blob/cb2878e50c8189d2e0a4814e2b732f8b8dd20f81/SecondModel/crop.py)and the reading part in [**readImage.py**](https://github.com/Bakame1/digitAC/blob/cb2878e50c8189d2e0a4814e2b732f8b8dd20f81/SecondModel/readImage.py)**.**

Cropping:

The main goal was to crop more pictures by separating this function we observed that really did not work in the first model on different images. To improve the cropping, we decided to use the fact that the screens of AC’s interfaces are usually gray/green. By creating a mask with these colors, the cv2 library permitted to find the contours of the screen based on it.

Thanks to this, we managed to crop more image correctly than before. For example, the image that failed to be processed in the first model can now be cropped correctly:

After upgrading the cropping, we can think that we just need to read the image. However, trying to do so can lead to a failure too. For instance, some other symbols/characters may be read even small details (see Figure 403)



To correct this problem, the strategy was to apply a blur so as to make the little details less readable and the temperature still distinguishable.

Here is what we done, we applied a bilateral filter so as to make the image more pixelated with a high strength. Then we try to read the image. If we cannot read the image, we apply half as much blur then we read it again with EasyOCR.

Figure RC3: Example of a blurred image

Thanks to just this 2 processes, without other image processing, we can now read at least 5 images. We managed to read pictures with the following number for instance: **1,2,5,58,74,142,177,204**.

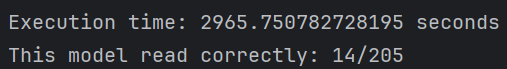
We can illustrate the process with picture n°5 in Figure RR404.

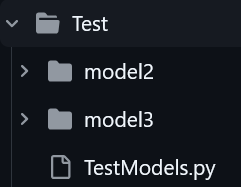
In that case, the cropping succeeded. Then we can see that the blur applied is too strong so the digit cannot be detected (step 4). We apply then less blur and the digit are finally found.

Another issue that happened was the incorrect reading of the digit 5. Sometimes it was read as a **“S”.** We managed this problem by replacing the character S by a **“5”** for the reason that the S mostly look to this number

### Establishing a testing program

Because when we modify a method we want to see if the image that worked one the previous model still works on them. Rather than testing one by one, it would be faster and better to have a program that test our model on all images. Like this, we would be able to compare our models.

Figure RC4: Example of the testing program result on model 2

How did we establish our testing program? First we created a directory named “Test” that contains different models adapted for the testing.

Indeed, the model usually display all the steps on 1 image and does not necessarily return a result. If we want to test one of our models on all the 205 images, we need to adapt these models in order to just return the value read without displaying any steps. Figure RC5: Testing directories

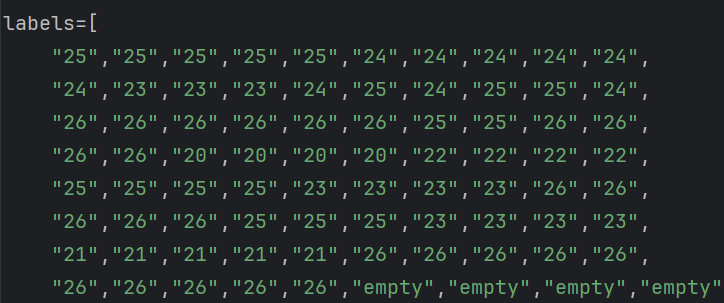
After doing this part, we created the file **TestModels.py**. We created a list (named labels) containing the value of all the pictures in string. For instance, the element 0 contains in this list is the string **“25”.** If there is nothing on the AC’s screen, the element related to this picture contains the string **“empty”**.

Figure RC6: A portion of the list of labels

Because there are only 205 pictures, we filled this list one by one. By doing this, we can now directly compare the result read on a picture and compare it with the correct one in this list. It can also be uncertainties about this list because we filled it manually.

Now we are able to do so iteratively and display relevant measures such as:

* Number of pictures read correctly on the total of the 205 images
* Execution time (so time it is very long) thanks to the library **time**
* Display which images we could not read with this model

### Establishing a third model

After testing model 2, we analyzed different image that could not be read. We noticed that a lot are not cropped correctly. We suppose that if we improve cropping, the execution time will reduce and the accuracy may increase. Therefore, in this model 3, we only focus on upgrading cropping.

Here are suppositions on the cropping fails:

* Angle of the picture
* Brightness that makes the screen harder to be detected by the HSV mask
* Elements in the background that have a color more near to the mask and that are rectangular
* Quality of the picture

To test different solutions, we grouped some photos per suppositions and we tried different new preprocess before the cropping. Finally, here are the 3 processes that improved the cropping on previous pictures:

* Converting the image to **grayscale** with the open-CV constant cv2.COLOR\_BGR2GRAY
* Applying a **gamma correction** to reduce the problems related to the brightness of the picture
* Using an **adaptive** **thresholding** on the image, which is particularly useful for converting a grayscale image to a binary image. The main objective is to handle images with varying illumination conditions, where a global threshold is suitable only for one specific light situation.

An important thing to say is that we try 3 different adaptive thresholding. In fact, by testing different values of the parameters in the cv2.adaptiveThreshold function. Some photos can be read with a “block\_sizes” parameter of 21, some with one of 51 and others with 81. After cropping using these 3 different thresholding, we take the picture cropped with the biggest width as the real cropped image.

The goal of the “block sizes” parameter is to define the size of the local neighborhood around each pixel in the image, which is used to calculate a threshold value adaptively.

A smaller “block\_sizes” results in a threshold that are more adaptive, which can be useful for capturing edges and details in the image. A larger “block\_sizes” smooths out more of the image, making it more suitable for general thresholding where the goal is to separate the foreground from the background.

Finally, we used our new testing program on this new model. We observed 2 main results(Figure RR7):

* + - The accuracy reduced
    - The execution time increased

### Establishing a fourth model

## 研究方法和研究思路（技术路线）Research methods and clues：

### Supervisor’s clues

Programming languages to use.

Guide me through different step to improve the model progressively.

Cropping the image so as to reduce the calculations.

Meeting to discuss about results and the next steps.

### Official Documentation reference

PyPI… To understand param, options,

### Stack Overflow

Stack Overflow is a question and answer website for programmers. It was created in 2008 by Jeff Atwood and Joel Spolsky. The site serves as a platform where developers can collaborate, share knowledge, and learn from each other. This website site was a good help when I had a problem or an error. Here I could see other people with the same issue and see what are the solutions brought by other programmers.

### Debugging

Like in every programming project, we had to debug the code by isolating the portion of the code that creates issues. By printing, displaying variables or processes, we could understand which section of the program had to be corrected and how.

### Videos

Especially for finding a library that reads text. A YouTube video helped me to use the EasyOCR library.

Advice to use python

github

Internet, overflow, tests debugging, separate sections of code and upgrade each, AI, persons with same prob, videos, upgrade testing method

piste : prof avec directives, videos, resultats des tests,

## 研究结果Research results：

### Model’s outputs

Model 1

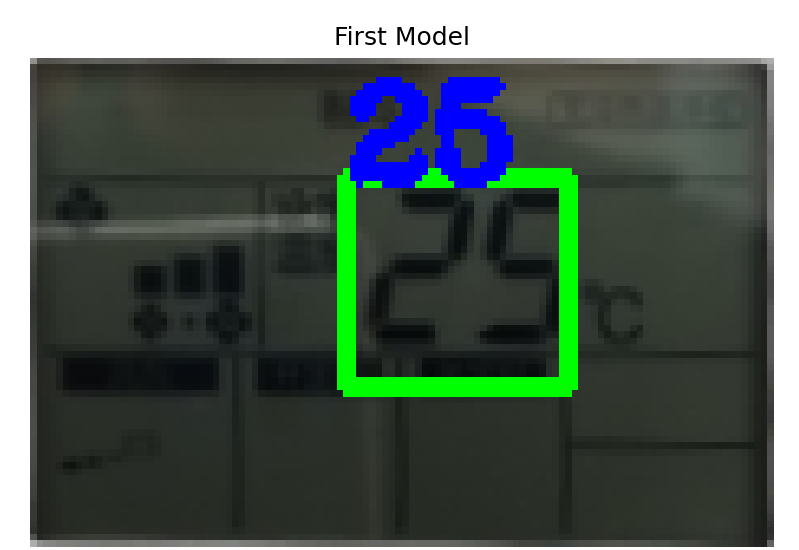
Figure RR2: First model’s failure on picture n°2

Figure RR1: First model successful result on picture n°1

Model 2



Figure RR3: Result of cropping the picture n°2 Figure RR4: Failure of reading a cropped image n°204

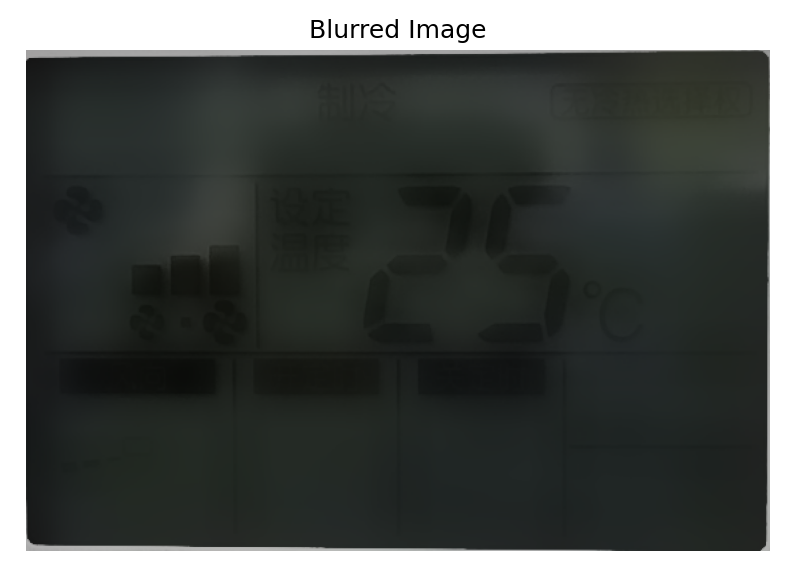
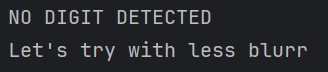
Even if the cropping succeeded, the reading may fail. On RR4, model 2 read **“\*”.**

After the model 2 organized and devised correctly, here are the different steps of it on figure RR5:

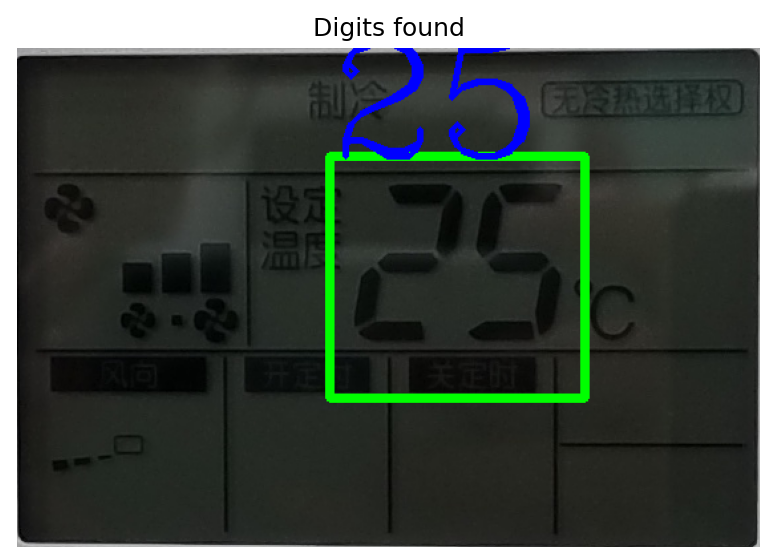


**2**

**1**



**4**



**6**

**5**

**3**

Figure RR5: The different steps of recognizing the digits with the second model of picture°5

Model 3

The 2 interesting examples are image 7 and 11. In fact, the image 7 help to see the problems related to objects in the background that could be cropped instead of the AC. Concerning the image 11, the problem was related to the brightness and the quality of the image that is poor.

**crop.py from model 3**

**crop.py from model 2**

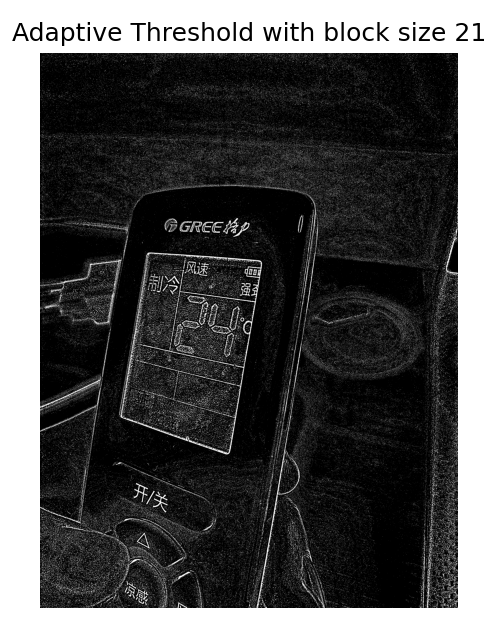
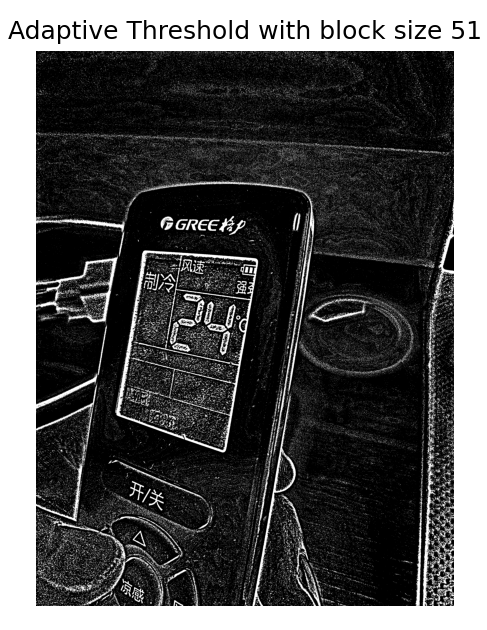
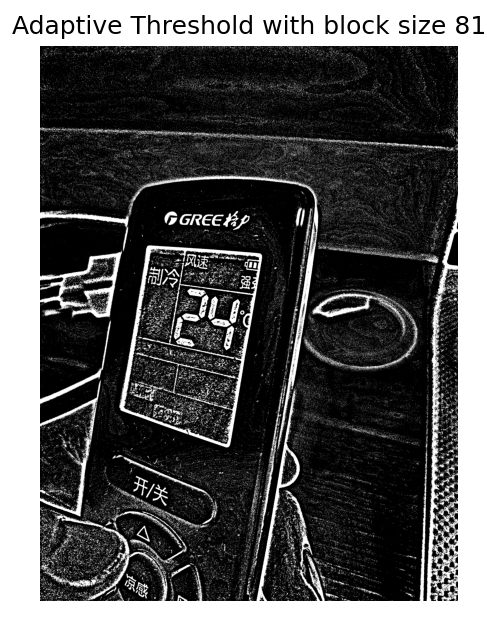
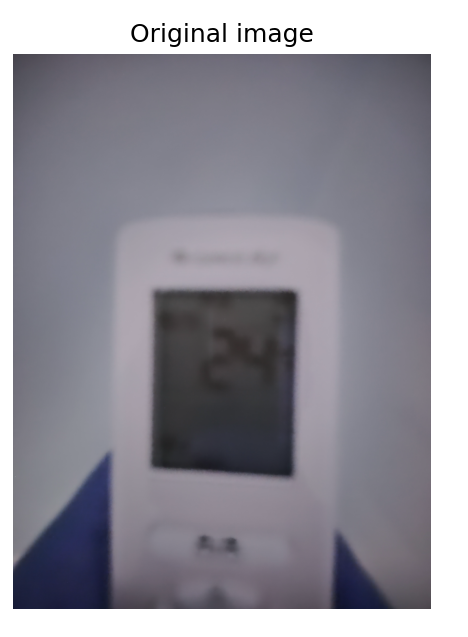
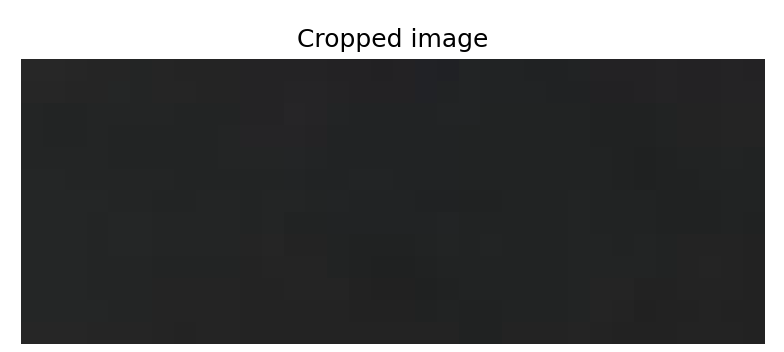
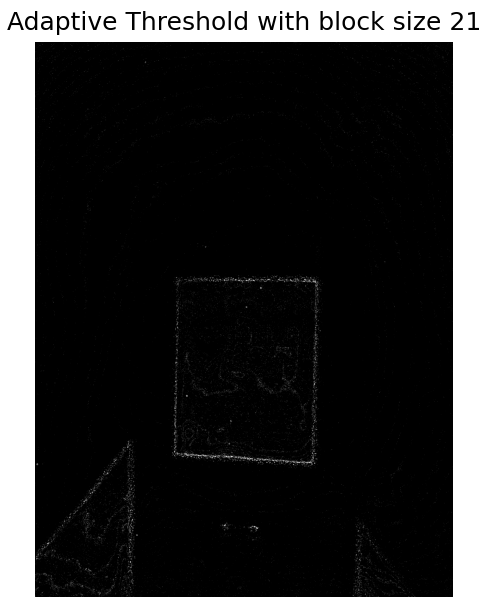
  

Figure RR6: cropping process on the image 7 from model 2 and 3

**crop.py from model 2**

**crop.py from model 3**

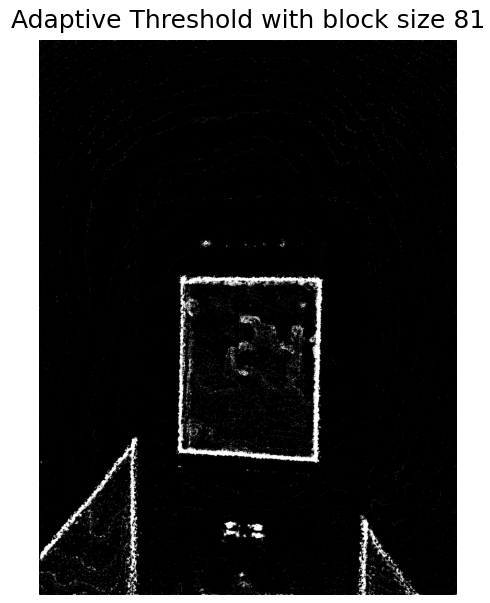
 

Figure RR7: cropping process on the image 11 from model 2 and 3

### Testing program results

Model 2

 Figure RR8: model 2 accuracy and execution time

Indeed, we managed to get approximately **9,7% of accuracy** and with an execution time around **38 minutes**.

Model 3

program on this new model. We observe 2 main results:

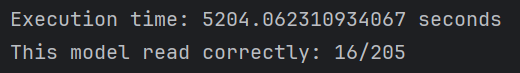
* + - The accuracy reduced
    - The execution time increased

Figure RR9: Testing output from model 3

## 研究进度安排Research schedule：

|  |  |  |  |
| --- | --- | --- | --- |
| **综合实践项目进度安排Researching plan：** | | | |
| No. | 各阶段内容 Progress | 时间安排Period | 备注Notes |
| 1 - 4 |  |  |  |
| 5 | Find a supervisor and start of the project | 10th – 12th October | Short amount of time |
| 6 | Taking 200 hundreds of photos in SJTU | 13th – 19th October | Long task |
| 7 | Testing different python libraries | 20th – 27th October | Many technical issues related to installations |
| 8 | Establishing a first model capable to recognize 1 picture | 28th October– 4th November | Find strategies on the image processing so as to isolate the digits |
| 9 | Testing and upgrading the first model | 4th– 14th November | Separate the different functions of the code and upgrade each one |
| 10 | New version capable of reading at least 7 pictures. Finishing the midterm report. | 15th– 21th November | A code more organized |
| 11 | Starting the final report | 22th– 27th November | Organize the ideas and the collected resources quoted or used. Fill it with the work done at this moment. |
| 12 | Establishing a testing program | 28th November– 2nd December | Get an idea of the accuracy of our model. Test the second model. |
| 13 | Upgrading cropping | 4th– 12th December | Find the common problems in cropping to improve the accuracy and execution time. |
| 14 | Upgrading the rest of the image processing | 13th– 22th December | Solve the problems found on pictures unread even after a good cropping. |
| 15 | Getting the best model as possible | 23th– 27th December | Increase the accuracy and the execution time. |
| 16 | Finishing the final report and clear the code | 28th December – Final report submit date | Make the code easier to read and the report more precise. |

## 参考文献References：

### Figures

Figure 1: Illustration of our project’s first result - [BACK]

Figure 123: Helin Gong - Picture of me taking a photo of AC – Mid October - [BACK]

Figure 139: Aurélien Max - Concepts, Objects, and Java: First Year of the Engineering Cycle in Computer Science at Polytech Paris-Saclay - 2023-24 - [BACK]

Figure 400: First model successful result

Figure 401: First model’s failure

### Quotes

[230] - [[LINK]](https://www.geeksforgeeks.org/python-language-advantages-applications/) - Python Language advantages and applications - Last Updated : 01 Nov, 2023 – [BACK]

### Libraries

**cv2** – [[LINK]](https://pypi.org/project/opencv-python/) - Released: Jun 18, 2024

**Numpy** - [[LINK]](https://numpy.org/doc/stable/) – Released 2024-08-18

**Matplotlib –** [**[**LINK**]**](https://matplotlib.org/stable/index.html) **-** Released 2024-05-16

**Imutils** – [[LINK]](https://github.com/PyImageSearch/imutils) – Last modification 2022

**Pytesseract** – [[LINK]](https://pypi.org/project/pytesseract/) - Released: Aug 16, 2024

**PaddleOCR** – [[LINK]](https://paddlepaddle.github.io/PaddleOCR/main/en/index.html) – Last release: 2024.10.18

**EasyOCR** – [[LINK]](https://pypi.org/project/easyocr/) - Released: Sep 24, 2024

**Time**

### Videos

YouTube Channel: Computer vision engineer – Text detection with Python and Opencv | OCR using Easy OCR | Computer vision tutorial - [[LINK]](https://youtu.be/n-8oCPjpEvM) - 2023

|  |
| --- |
| 指导教师意见Supervisor’s opinion 指导教师意见（课题难度是否适中、工作量是否饱满、进度安排是否合理、研究是否达到预期要求等）：  Supervisor's opinions (whether the project difficulty is suitable, whether the workload is full, whether the schedule is reasonable, whether the research is qualified, etc.)  指导教师签名Supervisor signature：  日期Date(YYYY-MM-DD)： |
| 学院（系）意见Institute's opinions 院长（系主任）签名Dean signature：  日期Date(YYYY-MM-DD)： |