University of Sheffield

Automatic Reading of Building Design Documents

Ming Liu

Supervisor: Dr Ramsay Taylor

COM6905 Research Methods and Professional Issues

This report is submitted in partial fulfilment of the requirement
for the degree of MSc in Computer Science With Speech and Language Processing
by Ming Liu

in the

Department of Computer Science

August 27, 2020

Declaration

All sentences or passages quoted in this report from other people's work have been specifically acknowledged by clear cross-referencing to author, work and page(s). Any illustrations that are not the work of the author of this report have been used with the explicit permission of the originator and are specifically acknowledged. I understand that failure to do this amounts to plagiarism and will be considered grounds for failure in this project and the degree examination as a whole.

Name: Ming Liu

Signature:

Date: August 27, 2020

Abstract

With a large number of data comes from a modern smart building every day, how to collect, store and analyse these data can be a problem. In this work, we only put our attention on a small part of the analysis process. Before doing the analysis, we need to figure out the relationship between Rooms and Sensors. Therefore, to be more specific, this work proposes an extraction pipeline, which can help to extract the relationship between Rooms and Sensors from the building drawings.

By partitioning the extraction process into three parts: Text and Coordinate Extraction, Relationship Extraction and Evaluation, fancy results can be achieved.

COVID-19 Impact Statement

The lockdown imposed because of COVID-19 caused additional challenges for the completion of this project. In the second semester of the project, the university switched to online delivery of all teaching, and university buildings were closed. All project meetings were shifted to email correspondence and video meetings.

Acknowledgement

First of all, I would like to thank my parents, two elder sisters, little nephew and an unborn baby nephew for motivating me to keep working hard. While studying at the University of Sheffield, they gave me generous support and encouraged me to keep moving forward.

At the same time, I would also like to thank my supervisor, Dr Ramsay Taylor, for his patient guidance, encouragement and support in this project, especially during this global pandemic periodic.

Meanwhile, I would also like to thank my friends and anyone I met in Sheffield. Thanks to all the teachers and staffs who gave me help and convenience during my studies.

Finally, I want to thank my ex-girlfriend for being with me in my toughest moments. If you ask me how I feel about you now, then what I want to tell you is that even if we haven't chatted for a long time, haven't spoken or even broken all contact, you are still a very, very important person to me. If you ask me what I want to say to you now, then I want to tell you that I like the marriage vows in "Corpse Bride": "With this hand I will lift your sorrows. Your cup will never be empty, for I will be your wine. With this candle, I will light your way into darkness. With this ring, I ask you to be mine."

In fact, I don't have many pursuits in my life. Every New Year, I am looking forward to it, and hope that all the regrets this year will be the preparation for the surprise next year.

Contents

List of Figures

List of Tables

Listings

Chapter 1

Introduction

Nowadays, indoors lives take us the most time (Jia et al., 2018). The quality of our lives relies largely on building construction. A modern building with hardware and software, as a shelter, keeps us from being affected by the outside surroundings and makes our life better. (Nimlyat, 2018). As a modern building, the basic elements like bricks and cement are not the only characters, the sensors are more important features instead. When a building was designed as a 'Smart Building', it means a number of services are recorded for tracking, analysing and improving. Omarov et al. (2017) propose some control methods on HVAC [1] systems, which can improve the quality in the room, as well as present an enhanced comfort presented to people. One study, which is based on the hospital, also shows that the positive quality indoors circumstances on patients could result in an affirmative meaning during regain (Nimlyat, 2018). Therefore, the smarter building, the better the life we enjoy.

With the rapid increasing of data we had from these sensors, more scenarios can be detected so that the building can adjust its power supply to save energy. That is why the technology we applied to buildings also plays a key role in lowering emissions (Tadokoro et al., 2014).

1.1 Problem Statement

The Diamond is a modern and high-tech building, which provides us with a suitable, convenient and enjoyable learning space by running a massive range of lighting, heating, networking, electrical and other services. These services will produce an abundance of data every day. By combining these dynamic data with static building drawings, the status of a specific room can be presented, as well as the related rooms. Therefore, how we extract the relationship between rooms and rooms as well as rooms and sensors can be a difficulty.

[1] Heating, ventilation, and air conditioning

1.2 Aims and Objectives

Our ultimate goal is to build a system, which automatically reads data and drawings, and reports to us what happened in a specific room at a specific time. However, it is a tough job. So this work aims to figure out the relationship between rooms and sensors through all the static building drawings. More specifically, it extracts the text and coordinate from drawings. After passing the drawings to the classifier, text and coordinate can be split into Room label or Sensor label respectively. At last, the relationship between rooms and sensors in terms of coordinate will be discovered by calculating the distances, which is similar to the K-Means clustering method (This is described in detail in Section 2.2.2) classifying data points into unrelated groups so that the data in the same group show similarity, whereas the data in the different group present more distinction (Na et al., 2010).

Besides, diverse ways, like PDFMiner in Python and Apache PDFBox® in Java, are taken in this work for text and coordinate extraction. (These are described in detail in Section 2.1.1.1 and 2.1.2) During the relation extraction process, it takes distance calculation or K-Means clustering methods to assess performance.

It is critical to mention the metrics of evaluation. The purity, for example, is used to assess the extent for a cluster has a single class (Sanderson, 2010). Also, Rand (1971a) raises a method called Rand index, which calculates the similarity between the result coming from our clustering algorithm and the standard classifications.

More significantly, this work is a fresh approach for relationship extraction in building drawings, and thereby, it is also essential to confirm the applicability of this approach in this area.

1.3 Overview

For chapter 2, it reviews some tools, methodologies and algorithms used in this work which provides some basic knowledge overall. In addition, chapter 3 contains some details in implementing the process as well as some requirements in which would be involved. What is more, Ethical, Professional and Legal Issues will be addressed in this chapter. After that, chapter 4 will promote the working plan. More importantly, the risk will be analysed ahead. In the last chapter, the conclusion of the whole work will be done. The bibliographies and appendices will be provided in the end.

Chapter 2

Literature Survey

This work is made up of three sub-tasks: Text and coordinate Extraction, Relationship Extraction and Evaluation. Some tools, methodologies and algorithms should be used to solving these tasks.

2.1 Text and coordinate Extraction Tools

In this section, some tools will be introduced to extract the text and coordinate from the PDF files of building drawings.

To begin with, all the data of this work comes from PDF files. Even though the PDF file is called the PDF Document, it is different from HTML or XML documents in terms of its structure. Most HTML or XML documents are structural documents, which show you where a sentence or a picture is, while the PDF file is more like a graph with coordinate. As the developer Shinyama (2015) says "PDF is evil".

2.1.1 PDFMiner in Python

The PDFMiner, as an open-source PDF-to-text converter, can extract information from PDFs and regenerate other types of formats.

2.1.1.1 The Classes in PDFMiner

There are at least five classes involved in the process of extracting a PDF file. **PDFDocument** class stores the data which is drawn from the **PDFParser** class. In terms of the **PDFResourceManager** class, which is used to process the fonts and images, the **PDFPageInterpreter** class can handle the page content. At last, the result produced from the above mentioned four classes will be transmuted by the **PDFDevice** class to whatsoever types. The connection between these five classes is illustrated in Figure 2.1.

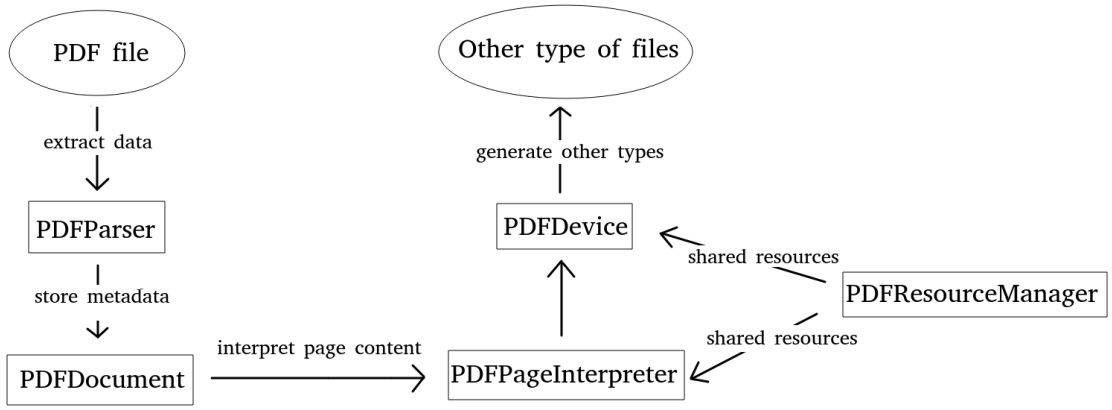


Figure 2.1: The Association Between Five Classes in PDFMiner

2.1.1.2 The Structure of PDF in PDFMiner

This work only touches on the text. Figure 2.2 is the structure of Text classes in PDFMiner. Figure 2.3 is the practical layout where these Text classes are presented.

LTPage:

It means the whole page and contains **LTTextBox**, **LTFigure** and **LTLine** or other objects. Nevertheless, this work does not care about the other types of object except for text.

LTTextBox:

LTTextBox is a square area where a bunch of **LTTextLine** are included. However, the square area is not a logical border, although it is formed by the analysis of the PDF file.

LTTextLine:

It has a group of **LTChar** objects where these **LTChar** objects are in the same line.

LTChar, **LTText**:

They are similar to each other, which holds a character.

2.1.1.3 Text Extraction Notes in PDFMiner

In addition to writing code with PDFMiner library, there are two utilities named **pdf2txt.py** and **dumppdf.py**. In the course of extracting text through PDFMiner, there are some parameters that need to be noticed. M, L, W are significant parameters. M means the margin of chars, L means the margin of lines and W means the margin of words. M = 2.0, L = 0.5, and W = 0.1 are the default values, individually. Besides, Figure 2.4 shows the layout of Text classes clearly. To adopt different PDF files, each value of margins should be appropriately chosen.

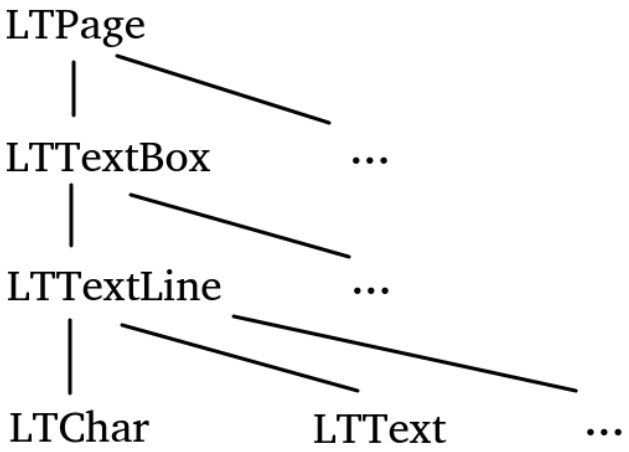


Figure 2.2: The Tree Structure of Text Classes in PDFMiner

2.1.2 Apache PDFBox® in Java

Apache PDFBox® is another open-source tool written in Java, which has the similar functionality with PDFMiner, it provides eight features *Extract Text*, *Split & Merge*, *Preflight*, *Print*, *Save as Image*, *Create PDFs* and *Signing*(PDFBox®, 2010). For more information, you can refer to Apache PDFBox® (https://pdfbox.apache.org/).

2.1.2.1 Text Extraction Notes in Apache PDFBox®

In addition to jar library, the Apache PDFBox® provides a command-line tool same as PDFMiner. This paper only focuses on **ExtractText** in which all the text inside the PDF file will be extracted. The usage of the command-line tool is shown as follows. Table 2.1 provides a part of the parameters for [**OPTION**]

Usage:
java -jar pdfbox-app-2.y.z.jar ExtractText [OPTIONS] <inputfile>[Text file]

Apart from that, when using java 8 or java 9, the version can not be higher than 1.8.0_191 or 1.9.0.4 respectively.

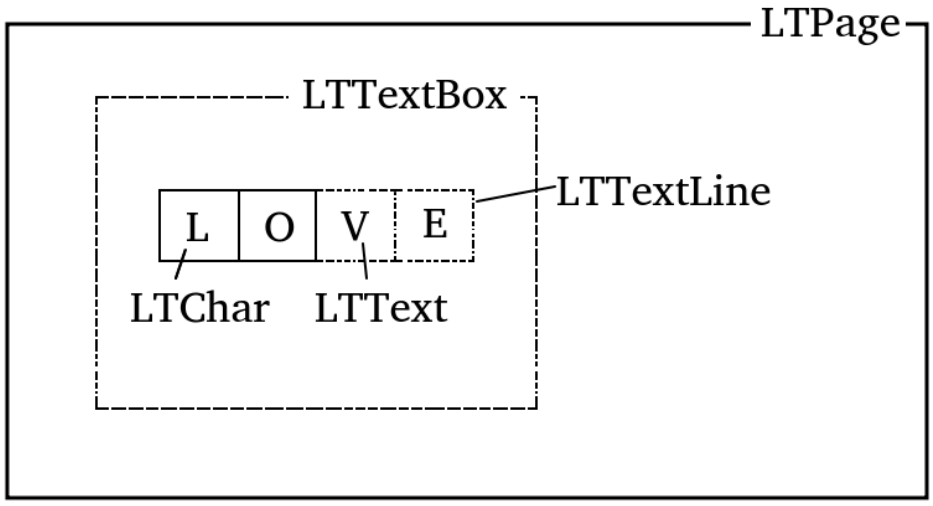


Figure 2.3: The Layout of Text Classes in PDFMiner

2.1.3 OpenCV OCR with Tesseract in Python

In this sub-section, there is a combined approach, which is different from PDFMiner or Apache PDFBox®, to identify words. First and foremost, it is not necessary to read the PDF files and extract information from them, while it uses OCR [1] to detect words. What is more, by taking EAST [2] deep learning model in OpenCV, all possible text areas can be detected. After that, all text areas containing text will be passed to Tesseract to recognise all text.

2.1.3.1 OpenCV OCR

OpenCV is an open-source, and a collection of function libraries focusing on solving real-time computer vision tasks (Pulli et al., 2012). You can find more details about OpenCV (https://opencv.org/) for reference. Based on the teamwork of Zhou et al. (2017), EAST algorithm is implemented as an OCR module in OpenCV. Zhou et al. (2017) state that EAST is an uncomplicated but formidable pipeline, which provides a text detection with swift speed and high precision in actual scenarios.

[1] Optical character recognition or optical character reader
[2] An Efficient and Accurate Scene Text Detector.

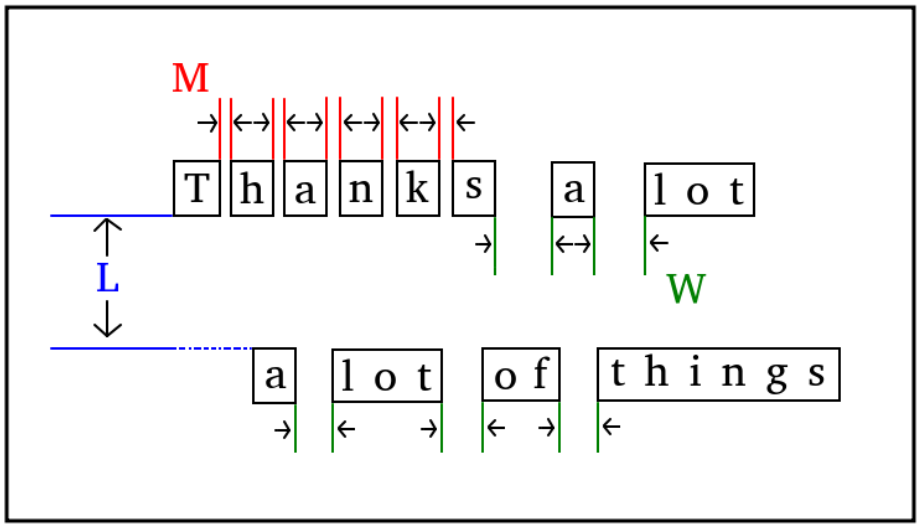


Figure 2.4: The Layout of Text Classes in PDFMiner

2.1.3.2 Tesseract

Tesseract is a free and open-source OCR engine which can be used in different platforms(Kay, 2007). Ever since 2006, it had become the most accurate OCR engine in the world sponsored by Google(Vincent and Lead, 2007).

Indeed, the Tesseract engine had shown its talent in 1995. In the work of Rice et al. (1995), they report that the accuracy of Tesseract reaches the top three among all OCR engines. With the development of Tesseract, it can recognize over 100 languages now. What is more, it could be retrained to adapt to other languages as long as sufficient data is provided.

2.1.3.3 Workflow

Figure 2.5 shows the workflow of text recognition by using OpenCV OCR and Tesseract. For the first step, PDF files should be converted to images so that they can be processed by the EAST algorithm in OpenCV. Then, the EAST will identify the regions of text and store it. Next, these regions called ROIs [3] will be put into Tesseract. Finally, The Tesseract will show the result of texts.

[3] A region of interest, are samples within a data set identified for a particular purpose.

Parameters	Default	Description
-password	EMPTY	The password of PDF file; it is empty by default.
-sort	False	If the value is True, it will sort result before output.
-html	False	If the value is True, it will generate an HTML file instead of pure text.

Table 2.1: Parameters for Command-line Tool in Apache PDFBox®

2.2 Relationship Extraction Methodologies

In this section, a few methodologies in extracting relationship between rooms and sensors will be present.

2.2.1 Euclidean Metric

Before introducing the Euclidean metric, the Euclidean space must be talked ahead. Because Euclidean metrics or Euclidean distance is a measurement between two data points among Euclidean space. For the more, there are many types of positive number dimension in Euclidean spaces. Our world could be a three-dimensional space where we can apply the Euclidean metric for measurement and compare it with others.

2.2.1.1 Mathematical Representation

The Euclidean distance is defined by the length between point **a** and point **b**. this paper only consider the 2-dimensional Euclidean space, which is the Euclidean plane. Figure 2.6 shows how distance is represented. Equation 2.1 gives the formula of computation.

$$d(\mathbf{a}, \mathbf{b}) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2} \tag{2.1}$$

2.2.1.2 Euclidean Distance Matrices (EDM)

In recent papers, the Euclidean Distance Matrices, or EDMs, come into vogue, and the most basic academic research is carried out by Young and Householder (1938). For example, Table 2.2 is a Euclidean Distance Matrices. The symmetric structure appears, which could be converted to heatmap 2.7 so that you can see it more clearly.

With these distance between rooms and sensors, we can simply classify certain sensors to one category, which belongs to a specific Room.

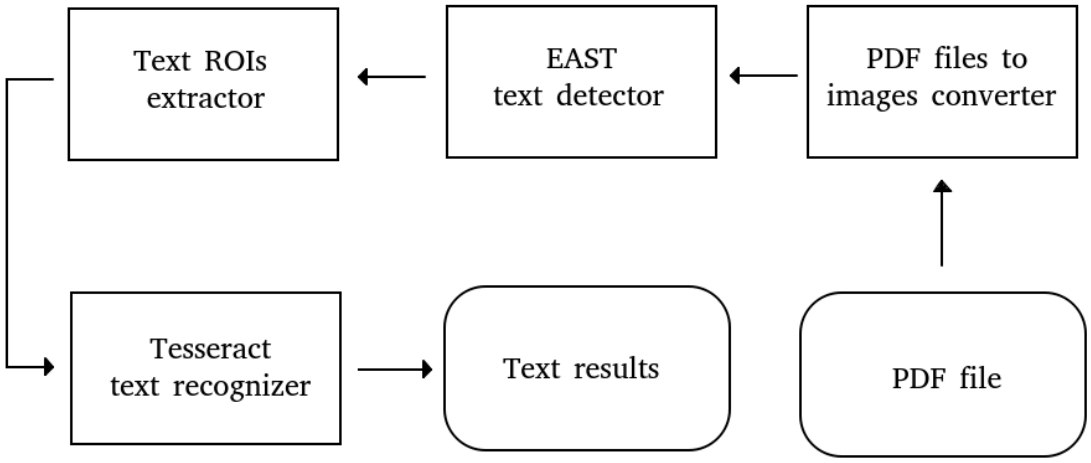


Figure 2.5: OpenCV OCR with Tesseract

2.2.2 K-Means Clustering

In 1967, The "k-means" was initially created by MacQueen et al. Nowadays, K-Means clustering is an iterative and unsupervised learning algorithm, which enjoys a high reputation around the world (MacQueen et al., 1967). Its aims at dividing datasets into Kpre-defined subsets containing no intersection data. What is more, it is usually in big data, machine learning and data mining (Lee et al., 2011).

In real society, this algorithm still has its limitation, and thereby some researchers extend this algorithm with background knowledge (Wagstaff et al., 2001) or domain knowledge (Huang, 1998) so that it could be more suitable for an actual situation. In addition, other groups apply this algorithm in some special areas, particularly in detecting schizophrenia (Lee et al., 2011).

Just to emphasize the point above, what we use here is the standard algorithm named the least squared Euclidean distance.

2.2.2.1 The Process of K-Means Clustering

Based on the distance between centres and data points, K-Means trys to split all datasets into Kpre-defined clusters so that the sum of distance within-cluster can be the smallest. Importantly, it is up to how we define the distance.

The process of K-Means Clustering is illustrated below:

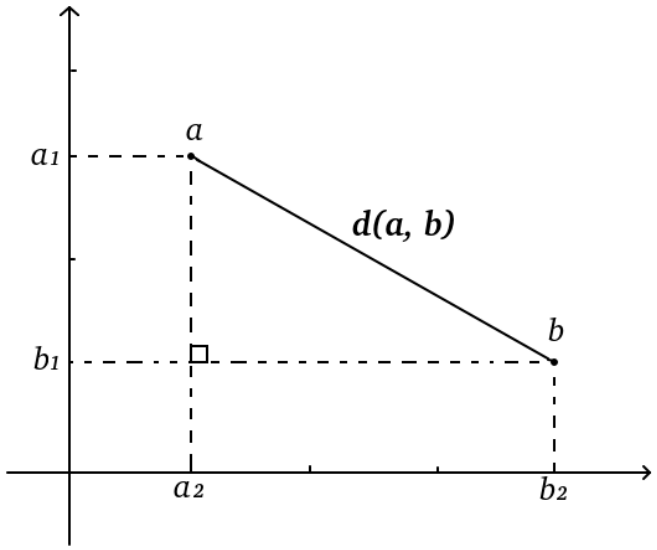


Figure 2.6: The Calculation Process Legend of Euclidean Distance

(s1) Firstly, the value of K should be set and then randomly choose data points in the number of K as initial, where K is lower than the size of the whole datasets centres.

(s2) Then, by calculating the distance between each centre points with all data points, we rearrange these data points into different groups where each group hold the smallest sum of distances.

(s3) Next, recalculating the new group centre points, which may not be an existing point.

(s4) Finally, by repeating the process from step (s2) until there are no changes in all centre points, the process comes to an end.

2.2.2.2 Mathematical Representation

S contains k sets, where $\mathbf{S} = S_1, S_2, ..., S_k$. If all the data points **x** are given by $(x_1, x_2, ..., x_n)$, and each x is made up of n-dimensional vectors, our goal is to figure out the equation 2.2 below.

	aa	bb	cc	dd	ee	ff
aa	0	201	188	99	130	211
bb	201	0	65	130	89	133
cc	188	65	0	121	111	104
dd	99	130	121	0	30	88
dd	130	89	111	30	0	115
dd	211	133	104	88	115	0

Table 2.2: An Example of Euclidean Distance Matrices

$$\begin{aligned} \arg\min_{\mathbf{s}} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2 &= \arg\min_{\mathbf{s}} \sum_{i=1}^{k} |S_i| \operatorname{Var} S_i \\ &= \arg\min_{\mathbf{s}} \sum_{i=1}^{k} \frac{1}{2\,|S_i|} \sum_{\mathbf{x},\mathbf{y} \in S_i} \|\mathbf{x} - \mathbf{y}\|^2 \end{aligned} \tag{2.2}$$

where μ_i is the mean of points in S_i.

2.3 Evaluation Algorithms

For the last section, it shows two types of evaluation algorithms in this work.

2.3.1 Purity

Purity is a common and primary metric, which is often used for evaluating the performance of the clustering result. In general, the more considerable value of purity means the higher quality of clustering (Sripada and Rao, 2011).

The formula of purity can be defined as Equation 2.3.

$$\frac{1}{N} \sum_{m \in M} \max_{d \in D} |m \cap d| \tag{2.3}$$

Where M is a series of clusters, D means a range of classes and N represents the number of data points in each group. However, when a lopsided result comes into purity algorithm,

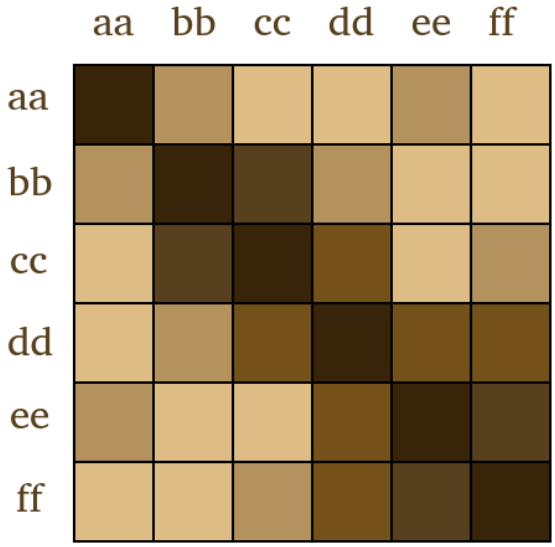


Figure 2.7: An Example of Euclidean Distance Matrices Heatmap

it becomes a disaster. For instance, supposing there are two clusters with 100 data points, it always gives a high purity value even if there is one cluster contains 98 data points, and the other only has 2.

2.3.2 Rand Index

Rand (1971b) proposes an algorithm, which is used for evaluating the quality of clustering by comparing the similarity between the result coming from our clustering algorithm and the standard classifications. Given the quantity of true positives TP, true negatives TN, false positives FP and false negatives FN, the equation is described as below:

$$RI = \frac{TP + TN}{TP + FP + FN + TN} \tag{2.4}$$

Nevertheless, There is a shortage in Rand Index, where the same equivalence is carried out among false negatives and false positives. Thereby, Rand (1971b) puts forward another method called Adjusted Rand Index to solve this problem. Furthermore, F-measure ends this issue too.

2.3.3 F-measure

F-measure, F score or F_1 score, is often used in text classification area for evaluating the performance of different classifiers (Fujino et al., 2008). In this work, we will apply this

algorithm to assess the performance of the relationship extraction pipeline. What is more, only the traditional F-measure will be taken into consideration. Here below equation is the formula of F-measure.

$$F_1 = \left(\frac{2}{\mathrm{R}^{-1} + \mathrm{P}^{-1}} \right) = 2 \cdot \frac{\mathrm{P} \cdot \mathrm{R}}{\mathrm{P} + \mathrm{R}}. \tag{2.5}$$

Where the P is precision and the R means recall. What is more, the formulas of precision and recall are described as:

$$\begin{aligned} P &= \frac{TP}{TP + FP} \\ R &= \frac{TP}{TP + FN} \end{aligned} \tag{2.6}$$

2.4 Summary

In this chapter, all the tools, methodologies and algorithms involved in this work are acquainted. There are three steps, which has been mentioned at the start of this chapter. In the first step, PDFMiner, Apache PDFBox® and OpenCV OCR with Tesseract are explained in detail, especially how it helps to extract text and coordinate. Then, in the second step, two methodologies: the Euclidean Metric and K-Means Clustering are introduced during the Relationship Extraction process. At last, three types of evaluation algorithms: Purity, Rand Index and F-measure are presented, helping assess the quality of our extraction pipeline.

Chapter 3

Requirements and Analysis

In this chapter, three sub-tasks will be put into details, showing how we handle the process of relationship extraction under the project requirements.

3.1 Project Requirements

As it is mentioned before, the project requires us to build a system, which can automatically describe the status when asking any rooms in the Diamond building. More powerfully, it could tell you how these rooms are related. When the temperature raises, which happens only in one room but not all rooms, it can tell you where is the possible damaged root of the cooling system because it holds the relationship between rooms and rooms, rooms and sensors.

However, such a colossal system can not be easily implemented. This work only focuses on the first step toward finding the relationship between rooms and sensors but not the whole system. Only if the system possesses these relationships, it can respond to any other types of problems. Thereby, in this work, we create a relationship extraction pipeline to tackle with this problem. Figure 3.1 shows how the data flow among these three steps.

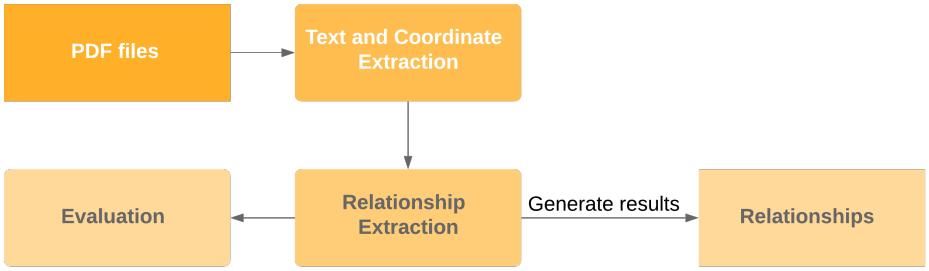


Figure 3.1: The Relationship Extraction Pipeline

3.2 Project Data

It needs to be emphasized that all documents we use, which comes from the contractor's drawings, are one particular type where the text is on floor plans.

3.3 Function Requirements

The functional requirements are shown in Table 3.1 below.

Function	**Sub-function**	**ID**	**Necessity**	**Description**
Text and coordinate Extraction	PDFBox Extractor	1.1	Mandatory	Extracting PDF file through Apache PDFBox®
	PDFMiner Extractor	1.2	Optional	Extracting PDF file through PDFMiner
	OpenCV Extractor	1.3	Optional	Extracting PDF file through OpenCV OCR with Tesseract
Relationship Extraction	Euclidean Distance	2.1	Mandatory	Extracting relationship through Euclidean Distance
	K-Means Clustering	2.2	Optional	Extracting relationship through K-Means Clustering
Evaluation	Purity metric	3.1	Mandatory	Using Purity metric to evaluate the performance
	Rand Index metric	3.2	Optional	Using the Rand Index metric to evaluate the performance
	F-measure metric	3.3	Optional	Using the F-measure metric to evaluate the performance

Table 3.1: The Function Requirements

3.4 Function Improvement

At the second step of the extraction pipeline, if we directly apply the Euclidean distance or K-Means clustering methodology, therefore, some information about the boundaries will be directly ignored. Since a room only has one corresponding text label to identify this room, if the room is too large, it will happen that a sensor, which is too far from the room text label, is judged to be associated with another room. To solve this problem, we shell use OpenCV to find the boundaries about the rooms so that the result of clustering can be better.

3.5 Ethical, Professional and Legal Issues

This work does not require ethical review because the building drawings and sensor information, which used during this work, are proprietary and owned by the University. Although getting it is easy, there are ethical concerns and legal issues with me not sharing too much of it with anyone. What is more, some aspects of building data can be personal and confidential, where it showed named peoples' offices for example. During the working period, data leaks or illegal operations never happens. All the data comes from the result of the experiments themselves without any artificial tampering. Moreover, it against nothing on BCS code of conduct or the legislation.

Chapter 4

Planning

4.1 Risk Analysis

Table 4.1 shows that to what extent do these risks affect the entire job, where the **Risk Level** from 1 to 3 represents low to high.

ID	Description	Risk Level	Action
1	Lacking experience on OpenCV tool; may cause the coding error.	3	Self-learning and communicate with supervisor regularly.
2	Boundaries extraction method may not work well; may cause ill result after the relationship extraction process.	2	Considering change other methods to extract the boundaries of rooms.
3	Lack of essay writing skills; may cause inappropriate writing style.	1	Attending the online writing courses provided by Department of Computer Science.
3	Real-time performance shortfalls because different program languages may be involved.	1	Consider using the same program language in dealing with the work.

Table 4.1: A List of Risks

4.2 Project Breakdown Structure

Figure 4.1 illustrates the project breakdown structure. Even if there are three sub-tasks, two of them can be classified as extraction operation. By exercising PDFminer, PDFbox or OpenCV OCR, the text and coordinate can be extracted from PDF files. After that, Euclidean Distance or K-Means Clustering is used in finding the relationship between text. At last, Purity, Rand Index and F-measure are carried out in the evaluation process.

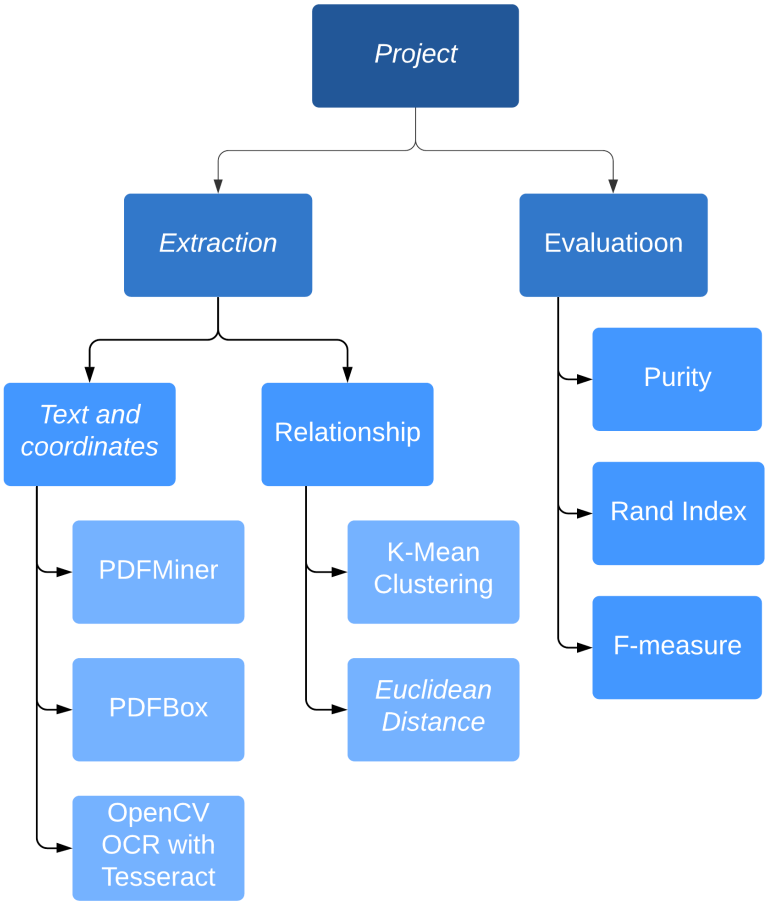


Figure 4.1: The Project Breakdown Structure

4.3 Project Plan

Table 4.2 shows that the main objectives with their dates of work.

Objectives	**Start**	**Finish**
Write Background Report	17 Apr, 2020	27 May, 2020
Text and Coordinate Extraction	29 Mar, 2020	25 Jun, 2020
Relationship Extraction	26 Jun, 2020	15 Jul, 2020
Evaluation	16 Jul, 2020	10 Aug, 2020
Write Dissertation Report	29 May, 2020	31 Aug, 2020

Table 4.2: The Main Objectives and Dates

There are more details about the project plan, which can be found in **appendix A** including the whole Gantt chart.

Chapter 5

Implementation, Assembly, Test and Evaluation

In this chapter, all methods mentioned above will be implemented. What's more, the most suitable method in each sub-task will be taken out to form the extraction pipeline. After that, the assembled extraction pipeline will take a test as well as three types of evaluation methods.

5.1 Implementation

In this section, three sub-tasks will be implemented in different ways, respectively.

5.1.1 Text and Coordinates Extraction

The first sub-task of the whole project is to extract the text and coordinates from building drawings. Before processing the drawings, there are two types of drawing should be introduced. One is a black and white drawing, where the text is on floor plans. The other is a colourful drawing, where the text mixes with other lines. Therefore, the drawing is a picture.

Figure 5.1 and Figure 5.2 are two types of building drawings, which show the same room named *E.06 BREAKOUT ROOM 3.7*. Owing to the types of building drawings are not unique, some methods are only suitable for one of these types. Hence, I apply PDFMiner tool as well as PDFBox tool to the first monochrome type of building drawing and use OpenCV tool to the colourful one.

In order to be consistent, only the file named *34676-M57-0302_Iss7.pdf* will be shown in this work during the process of extraction. Character $ will be used to represent line breaks when representing the content of the text.

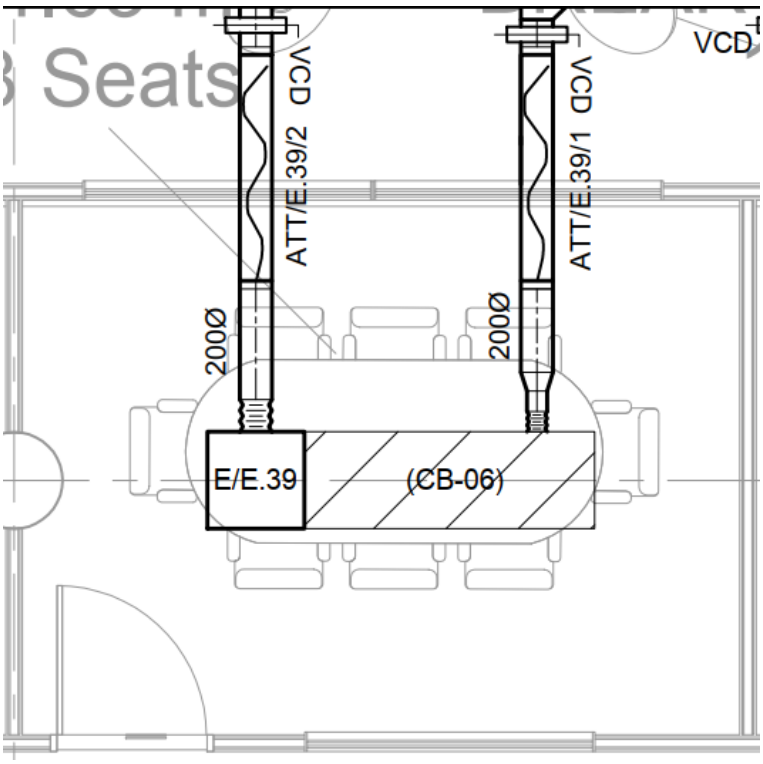


Figure 5.1: A Sample of Monochrome Building Drawing

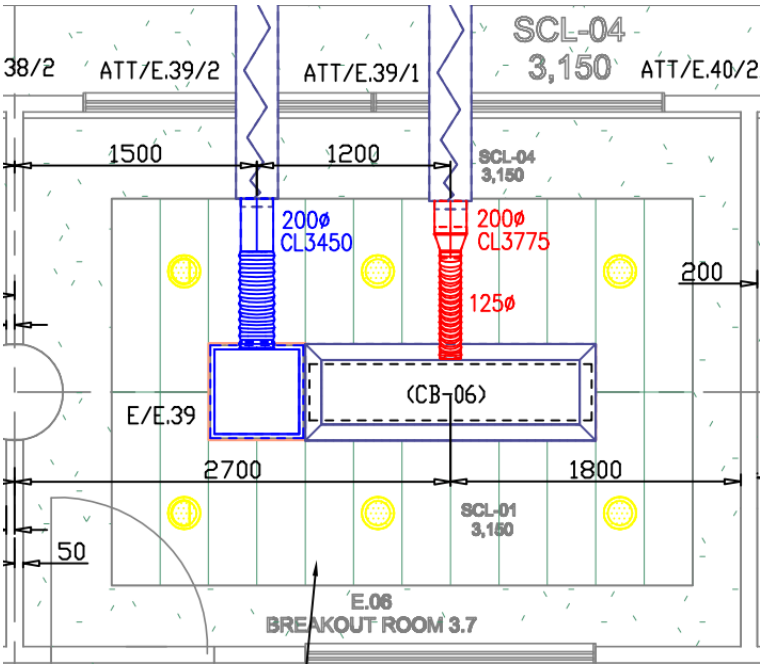


Figure 5.2: A Sample of Colourful Building Drawing

5.1.1.1 PDFMiner Extractor Implementation

To solve the first sub-task, I firstly used a tool named PDFMiner written in python language. It is a third-part open-source tool extracting information from PDF document.

The code, which I used to extract text and coordinate, is based upon the tutorial example in the PDFMiner docs(https://pdfminersix.readthedocs.io/en/latest/tutorial/composable.html) and the example from Code Examples (https://code-examples.net/en/q/15d65e1).

Extraction Result of the PDFMiner Extractor

```
------------------------------------------------------------
          ID,       X-axis,   Y-axis,    Text

          78,       641,      1440,      SD.2901

          79,       645,      1176,      (CB-06)

          80,       670,      1341,      BREAKOUTROOM3.8

          81,       670,      1224,      (cid:145)
                                         (cid:19)
                                         (cid:19)
                                         (cid:21)

          82,       675,      2105,      1150x500

          83,       684,      1683,      SD.2901

          84,       699,      1256,      1
                                         9
                                         E
                                         T
                                         T
                                         A

          85,       699,      1281,     .

          86,       701,      1311,      V
                                         C
                                         D
------------------------------------------------------------
```

A Sample of Extraction Result by PDFMiner

The above part is a sample result of extraction by PDFMiner tool. Where *ID* means the number of discovered text, *X-axis* means x-axis coordinate, *Y-axis* means y-axis coordinate and *Text* means the text of the corresponding x-axis and y-axis coordinates. It should be emphasized that coordinate (0, 0) locates on the bottom-left of building drawings. Therefore, the value of y-axis goes up when a point moves from bottom to top on the building drawings. Similarly, the value of x-axis increases when a point shifts from left to right.

The above result, which is extracted by PDFMiner, shows that most of the text can be extracted correctly. Notably, the vertical text also can be extracted according to the visual order from top to bottom.

However, the extraction effect of PDFMiner is not good. There are four main shortages from the above result.

Firstly, it shows that the text 'SD.2901' appears in coordinate (641, 1440) and ID is 78. Nevertheless, Figure 5.3 indicates that the actual text in the red rectangle area is 'S/D.29/01'. As a result, forward slashes always are ignored by this tool.

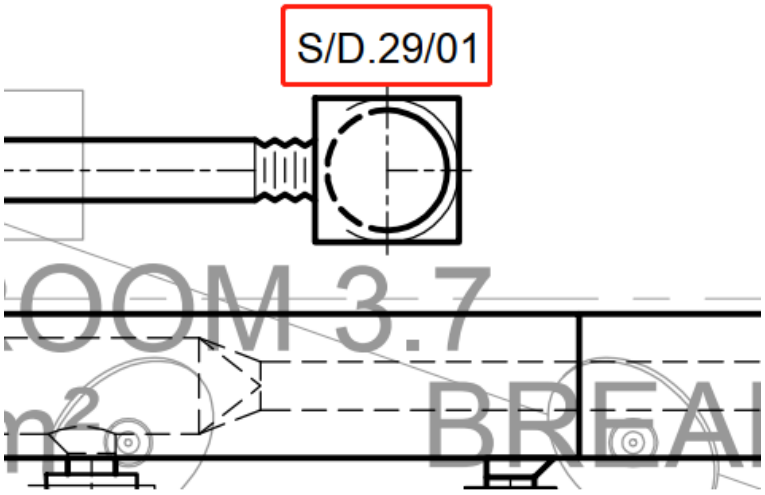


Figure 5.3: The Actual Text: 'S/D.29/01'

Secondly, it shows that the text 'BREAKOUTROOM3.8' turns up in coordinate (670, 1341) with ID is 80, but Figure 5.4 shows the actual text in the red rectangle area is 'BREAKOUT ROOM 3.8'. It means the spaces are removed by this tool, and the words thereby cannot be identified. Besides, the blue and red rectangle areas should be an entirety, so all text in these areas are better to be extracted at the same time.

Thirdly, it shows that the text '(cid:145)$(cid:19)$(cid:19)$(cid:21)' comes out in coordinate (670, 1224) with ID is 81, yet Figure 5.5 demonstrates the actual text in the red rectangle area is '200∅'. After some investigation, it is clear that PDFMiner writes strings like '(cid:...)' when it is not able to recognise the letter font or encoding. Fortunately,

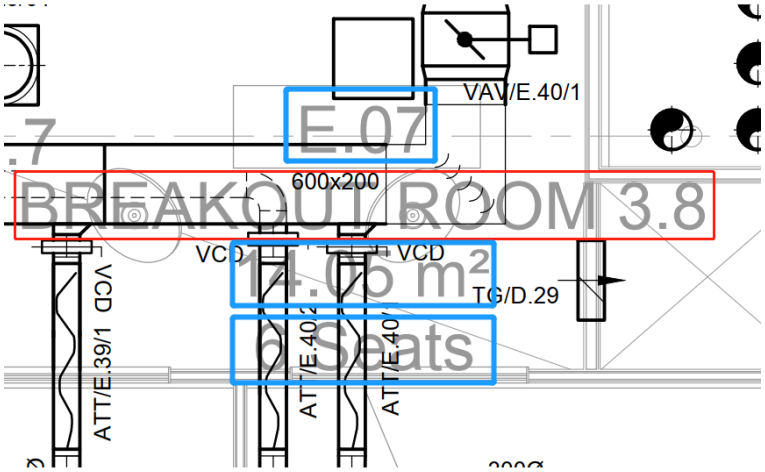


Figure 5.4: The Actual Text: 'BREAKOUT ROOM 3.8'

only some special characters, which are not targeted text, fail to be recognised.

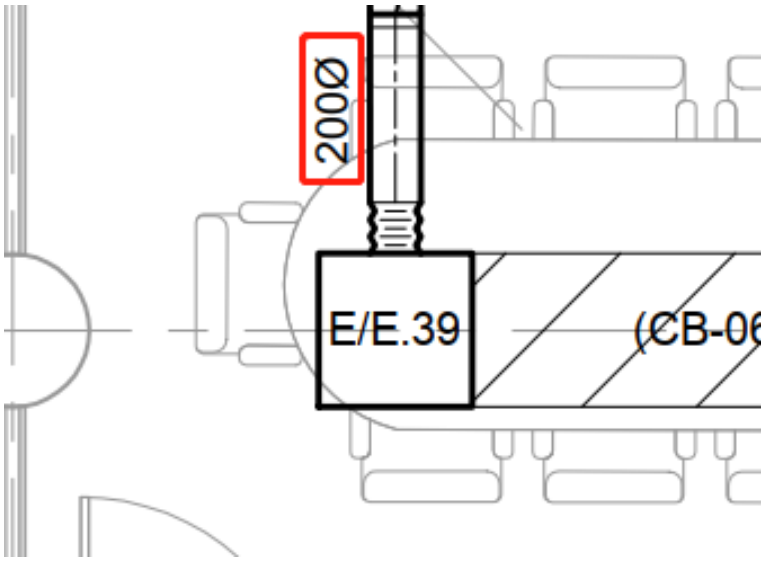


Figure 5.5: The Actual Text: '200⌀'

Fourthly, it shows that the text '1$9$E$T$$T$A' presents in coordinate (699, 1256) with ID is 84, however Figure 5.6 indicates the actual text in the red rectangle area is 'ATT/E.39/1'. The orientation, which is ignored by this tool, is the problem here. There are three directions in Figure 5.6: Direction from left to right; Direction from top to bottom; Direction from bottom to top. According to the result sets, the tool neglects the direction from bottom to top so that the extracted text looks upside down.

Besides, there is a small problem that the extracted text may do not appear in building drawings. As I look through the whole result sets, only some meaningless dots are identified,

so it has almost no effect on our task.

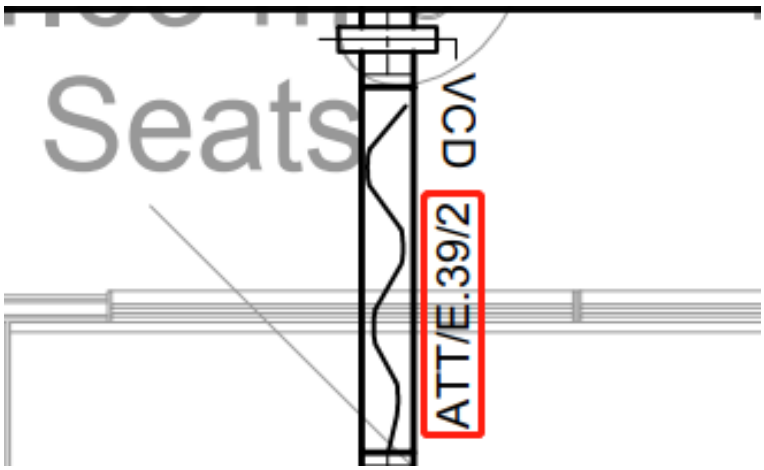


Figure 5.6: The Actual Text: 'ATT/E.39/1'

Conclusion of the PDFMiner Extractor

Anyhow, PDFMiner is a useful tool in terms of extracting information from PDF file, although it is not appropriately suitable for our task. This sub-task requires the text and coordinate of rooms and sensors should be correctly and orderly extracted. It means that the missing text and the wrong order can not be tolerated, therefore the first, second and fourth shortages are not acceptable so that the PDFMiner Extractor needs to be given up from handling this sub-task.

5.1.1.2 PDFBox Extractor Implementation

The next tool, which is named PDFBox wrote in Java language, is used to extracting text with its coordinate from monochrome PDF document.

The code, which I used to extract text and coordinate, is based upon the example codes of the Apache PDFBox(https://github.com/apache/pdfbox/blob/trunk/examples/src/main/java/org/apache/pdfbox/examples/util/PrintTextLocations.java) in GitHub.

Extraction Result of the PDFBox Extractor

```
1 ------------------------------------------------------------
2       ID,       X-axis,       Y-axis,       Text
3
4       58,       489.58,       940.76,       E/D.29
5
6       59,       504.75,       884.72,       VCD
7
8       60,       532.86,       1009.6,       BREAKOUT ROOM 3.7
```

```
9
10      61,     533.06,     977,        E.06
11
12      62,     534.25,     1069.8,     8 Seats
13
14      68,     610.72,     1104.2,     ATT/E.39/2
15
16      63,     560.43,     763.76,     VCD
17
18      64,     581.32,     232.32,     800
19
20      65,     589.99,     1205,       E/E.39
21
22      66,     605.08,     1055.7,     VCD
23
24      67,     609.40,     193.68,     100
25
26      69,     644.22,     789.20,     VAV/D.29/01
27
28      70,     660.46,     941.84,     S/D.29/01
29
30      71,     661.34,     1205,       (CB-06)
31 ------------------------------------------------------------
```

A Sample of Extraction Result by Apache PDFBox

The above part is a sample result of extraction by PDFBox tool. It should be noted that that coordinate (0, 0) locates on the top-left of building drawings, which is different from the previous PDFMiner tool. Thus, The horizontal coordinate will increase from left to right. Correspondingly, the vertical coordinate is going to increase from top to bottom.

There are also some shortfalls, but PDFBox looks much better comparing to the PDFMiner. First of all, it correctly extracts text with space. For example, when ID is 60 and the coordinate is (532.86, 1009.6), the text is 'BREAKOUT ROOM 3.7'. Next, the forward slashes are perfectly spotted as well, such as the text is 'ATT/E.39/2', when the coordinate is (610.72, 1104.2) and Id is 68. Owing to the supporting of an arbitrary angle of text rotation provided by PDFBox, it is undoubted that any direction of text can be found rightly. In addition, PDFBox can extract more accurate coordinates than PDFMiner.

Notwithstanding PDFBox can make up for some shortcomings coming from PDFMiner, others are still problems. PDFBox also holds the same problem as PDFMiner, where not all types of fonts and encodings can be recognised. It is various from PDFMiner that the unrecognised text will not be displayed but simply discarded. So as to be able to detect this problem, we need to observe the output warning messages while running the PDFBox detector codes.

```
Aug 11, 2020 3:31:39 PM org.apache.pdfbox.pdmodel.font.PDType0Font toUnicode
WARNING: No Unicode mapping for CID+36 (36) in font ArialMT
Aug 11, 2020 3:31:39 PM org.apache.pdfbox.pdmodel.font.PDType0Font toUnicode
WARNING: No Unicode mapping for CID+81 (81) in font ArialMT
Aug 11, 2020 3:31:39 PM org.apache.pdfbox.pdmodel.font.PDType0Font toUnicode
WARNING: No Unicode mapping for CID+55 (55) in font ArialMT
Aug 11, 2020 3:31:39 PM org.apache.pdfbox.pdmodel.font.PDType0Font toUnicode
WARNING: No Unicode mapping for CID+241 (241) in font ArialMT
Aug 11, 2020 3:31:39 PM org.apache.pdfbox.pdmodel.font.PDType0Font toUnicode
WARNING: No Unicode mapping for CID+18 (18) in font ArialMT
```

Figure 5.7: The Warning Messages of Unknown Font or Encoding

Figure 5.7 shows that some text has no Unicode mapping in font ArialMT so that these text can not be extracted. After investigation, I found that the problem did not affect our task because all texts related rooms and sensors already exist. Only a small part, such as room size and the unit of area (m^2), are ignored.

Besides, if you focus on three IDs(60, 61, 62), it obvious that these three parts of the text should be considered as a whole to describe a room. Also, the order of these three IDs needs to be adjusted to meet the actual situation, thereby the correct order should be 61, 60 and 62. Figure 5.8 shows the actual text.

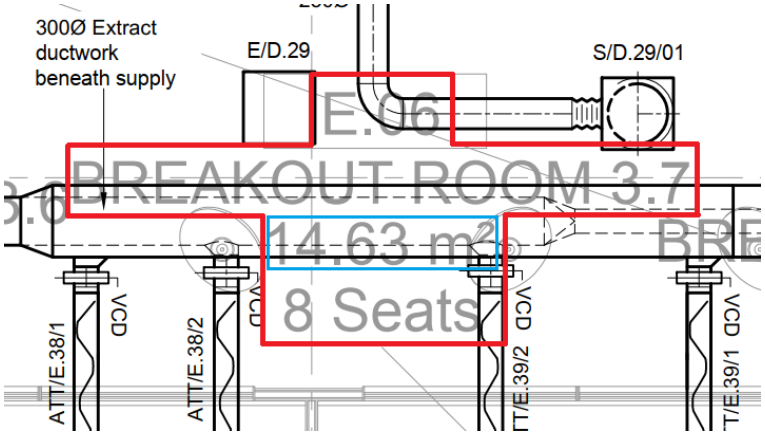


Figure 5.8: The Actual Text: 'E.06 BREAKOUT ROOM 3.7 14.63 m^2 8 Seats'

Conclusion of the PDFBox Extractor

According to the extraction results, PDFBox is suitable for this sub-task when applying to monochrome PDF files. All useful and related texts are correctly extracted without error. However, only one problem has already been mentioned that some texts about room names are in an incorrect order, thus some steps need to be taken during the extraction process to ensure order.

5.1.1.3 OpenCV Extractor Implementation

The following tool, which is used to recognise text from colourful PDF files, is different from the previous two. The colourful PDF files, which is used in our project, contain text where

not on floor plans, as I mentioned before. So as to handing this type of text recognision, I use OCR with Pytesseract and OpenCV.

Installation

It is quite easy to install OpenCV, Tesseract and pytesseract[1] on Ubuntu with below commands. In addition, Anaconda is required to be installed before you use these commands.

```
$ sudo apt install python3-opencv
$ sudo apt install tesseract-ocr libtesseract-dev
$ conda install -c conda-forge poppler pytesseract
```

Extraction Code

Code section 5.1 contains the whole extraction process, where the code is based on the tutorial example on the Nanonets website(https://nanonets.com/blog/ocr-with-tesseract/#installingtesseract). The PDF file is converted to JPEG file first, and then Tesseract will try to find all texts in the JPEG file. Finally, all the texts will be highlighted with green rectangles.

```
1  import cv2
2  import pytesseract
3
4  from PIL import Image
5  from pdf2image import convert_from_path
6
7  # reset maximum image pixels (34676-M57-0302_Iss7.pdf: 387460068)
8  Image.MAX_IMAGE_PIXELS = 1000000000
9
10 # file name
11 MONOCHROME_FILE = '34676-M57-0302_Iss7'
12 COLOURFUL_FILE = 'AMG-34676-M57-0302_Iss2'
13 FILE_NAME = COLOURFUL_FILE
14
15 # convert pdf to jpeg
16 pages = convert_from_path(FILE_NAME + '.pdf', 500)
17
18 for page in pages:
19     page.save(FILE_NAME + '.jpg', 'JPEG')
20
21 # read picture file
22 img = cv2.imread(FILE_NAME + '.jpg')
```

[1]The python wrapper for tesseract.

```
23 h, w, c = img.shape
24
25 # mark the text in the picture
26 boxes = pytesseract.image_to_boxes(img)
27 for b in boxes.splitlines():
28     b = b.split(' ')
29     img = cv2.rectangle(img, (int(b[1]), h - int(b[2])), \
30             (int(b[3]), h - int(b[4])), (0, 255, 0), 2)
31
32 # write picture to file
33 cv2.imwrite(FILE_NAME + '_with_boxes.jpg', img)
34 # show the picture
35 # cv2.imshow(FILE_NAME + '_with_boxes', img)
```

Listing 5.1: Extraction Code Using OpenCV with Tesseract

Extraction Result of the PDFMiner Extractor

Figure 5.9 illustrates a part of extraction results, which is extracted through OpenCV and Tesseract. All green boxes in Figure 5.9 represent a character or a group of characters. Sadly, the results show that OpenCV and Tesseract are not suitable for the current task, where the PDF file contains so many details that makes OpenCV and Tesseract overcapture text. What is worse, this combination tool can not provide coordinate of text. In my entire project structure, I need to cluster the text by coordinates. Without text with its corresponding coordinate, the work can not be finished.

Besides, there are some issues when recognising text. To begin with, texts enclosed by the red rectangle in Figure 5.9 show that not all characters in PDF file are correctly extracted, hence some related information will lose. Not only the text related to the room could not be fully extracted, but also the text referred to the sensor can not be extracted entirely. Therefore, it is impossible to identify room and sensor through these texts.

Next, there are so many false detections. If you put your attention on the text enclosed by the sky-blue rectangle in Figure 5.9, the unknown content in the number 0 is detected.

Then, the recognised results enclosed by the yellow rectangle in Figure 5.9 are not text from a visual point of view.

Last but not least, Figure 5.10 shows that the text enclosed by the sky-blue rectangle is impossibly recognisable. Because this type of PDF file contains text, which is not on floor plans so that some texts are covered by the line of a wall or facility. Moreover, There are a lot of green lines in the right half of Figure 5.10, which looks messy.

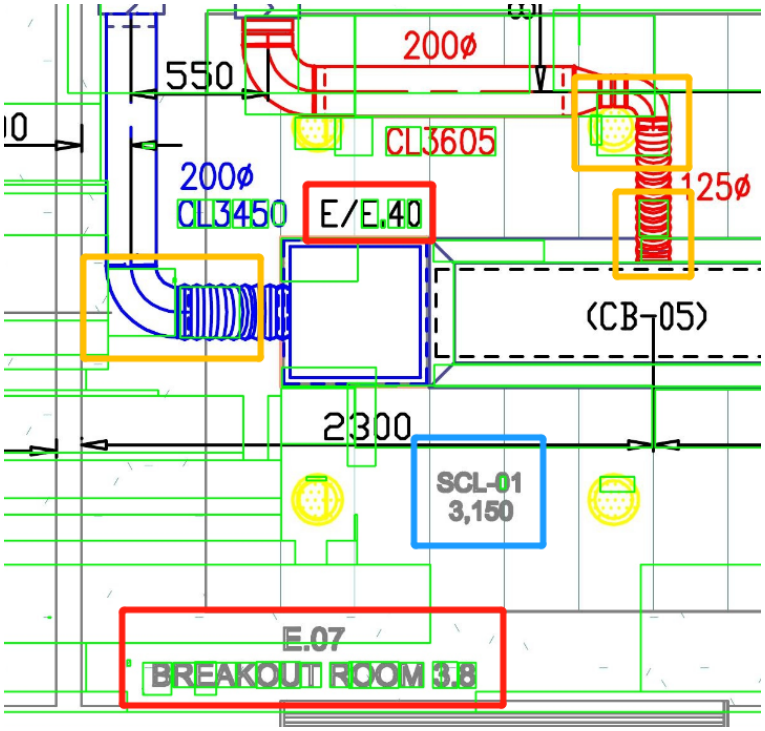


Figure 5.9: A Part of Extraction Result by OpenCV and Tesseract

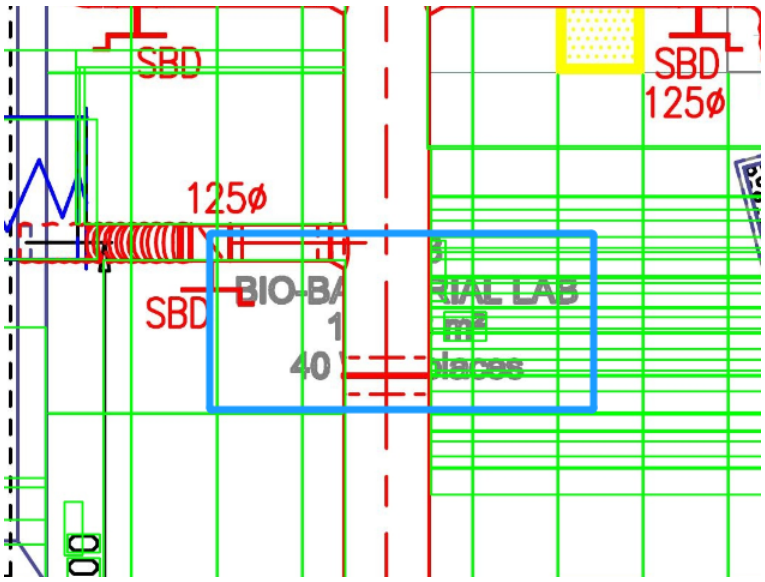


Figure 5.10: The Unrecognisable Covered Text

Conclusion of the OpenCV and Tesseract Extractor

It's quite obvious that this join tool is not suitable for this sub-task as the result indicates this tool will mix up all texts. Particularly, it extracts texts without its corresponding coordinate. Thus, the next task can not be carried out. Comparing to PDFMiner and PDFBox, OpenCV and Tesseract have more shortages than them.

5.1.2 Relationship Extraction

The second sub-task of the whole project is relationship extraction, which requires a clustering process to assign all sensors to its corresponding room after the text and coordinate of room and sensor are correctly extracted.

In this section, all work is based upon the results of the previous sub-task, which means there are no PDF files anymore. The more reliable the results got from previous sub-task, the more accurate the results obtained from the clustering process.

5.1.2.1 Euclidean Distance Implementation

The first idea that comes to my mind is to calculate the distance between rooms and sensors, where the simplest distance algorithm should be the Euclidean distance. Associate the sensor with the room closest to it until all sensors are assigned to a room.

Extraction Code

Code section 5.2 shows the core function code to extraction relationship between rooms and sensors, where it contains a parameter that is a type of distance calculation function. By default, the function is the Euclidean distance function, where code section 5.3 indicates its implementation. Additionally, the function can be replaced by other types of distance function, such as Manhattan distance, Minkowski distance and Chebyshev distance. Amongst the process of relationship extraction, all the sensors information iterate first, and then distance function will be applied to calculating the distance between the sensor and all rooms. After that, the number of the room for the shortest distance will be set as the group ID of the sensor.

```
/**
 * Assign all sensors to its closest room.
 *
 * @param func A Distance Calculation Function
 */
public void extractRelation(BiFunction<
        CoordinateText.Coordinate,
        CoordinateText.Coordinate, Float> func) {
    // check label
    if (!canExtract) {
        return;
```

```
12      }
13
14      // iterate all sensors
15      coordinateSensorsTexts.forEach(coordinateSensorText -> {
16          List<Float> distances = Lists.newArrayList();
17          coordinateRoomsTexts.forEach(coordinateRoomText -> {
18              // calculate and store distance
19              distances.add(func.apply(
20                      coordinateSensorText.getMiddle(),
21                      coordinateRoomText.getMiddle()));
22          });
23
24          // find minimum distance
25          Float minDis = Collections.min(distances);
26          int groupId = distances.indexOf(minDis);
27
28          // set groupId
29          coordinateSensorText.setGroupId(groupId);
30      });
31  }
```

Listing 5.2: Relationship Extraction Code by Calculating Distance

```
1  // A Function of Calculating Euclidean Distance
2  public static BiFunction<CoordinateText.Coordinate,
3          CoordinateText.Coordinate, Float>
4              EUCLIDEAN_FUNC = (a, b) -> (float) Math.sqrt(
5                  Math.pow(b.getX() - a.getX(), 2)
6                  + Math.pow(b.getY() - a.getY(), 2)
7              );
```

Listing 5.3: A Function of Calculating Euclidean Distance

Relationship Extraction Result by Euclidean Distance

The below part is a sample result of relationship extraction through calculating the euclidean distance between sensors and rooms to determinate the group. Firstly, there are two types of *Type* which are *ROOM* and *SENSOR* respectively. Then, the id of groups are random, but the id is the same when some sensors are close to the same room.

It is clear from the results that some texts are the same but different in position. However, it is not the fault of the previous text and coordinate extractor, because the PDF file cannot provide more details about the same text. Therefore, it can be ignored here.

Besides, the results shows that two *ATT/E.39/2* are identified as group *3* . Nonetheless, Figure 5.11 indicates only four texts of sensors, which are enclosed by the red rectangles, are belonging to *E.06 BREAKOUT ROOM 3.7*. Texts in blue rectangles are false detection. It

is quite easy to understand that these selected texts are closer to *E.06 BREAKOUT ROOM 3.7* but not other rooms. So how to reduce this false positive can be another problem waiting to be solved. I will introduce how to deal with this problem in a later section.

```
-------------------------------------------------------------
   X-axis,   Y-axis,    Text,                Type,      Group

   532.86,   1023.44,   E.06                 ROOM,      3
                        BREAKOUT ROOM 3.7
                        8 Seats

   390.64,   1092.1,    ATT/E.38/1,          SENSOR,    3

   456.28,   1090.8,    ATT/E.38/2,          SENSOR,    3

   610.72,   1104.2,    ATT/E.39/2,          SENSOR,    3

   489.58,   672.68,    E/D.29,              SENSOR,    3

   703.90,   698.48,    S/D.29/01,           SENSOR,    3

   644.22,   789.20,    VAV/D.29/01,         SENSOR,    3

   489.58,   940.76,    E/D.29,              SENSOR,    3

   660.46,   941.84,    S/D.29/01,           SENSOR,    3

   589.99,   1205,      E/E.39,              SENSOR,    3

   810.42,   1050.91,   E.07                 ROOM,      4
                        BREAKOUT ROOM 3.8
                        6 Seats

   710.43,   1105.57,   ATT/E.39/1,          SENSOR,    4

   798.52,   1095.85,   ATT/E.40/2,          SENSOR,    4
-------------------------------------------------------------
```

A Sample of Relationship Extraction Result by Euclidean Distance

Conclusion of the Euclidean Distance

It is quite convenient to use Euclidean distance to calculate the distance between the sensor and the room for grouping. Even though this method shows its talent in clustering, it is not perfect because the results contain some false positive. There is still work to be done to improve accuracy.

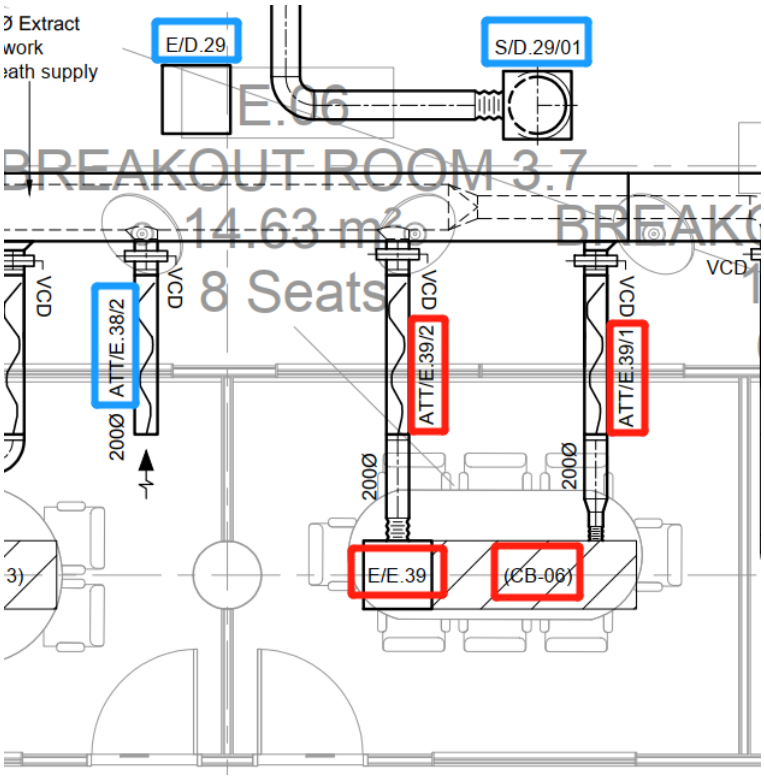


Figure 5.11: The Relationship Extraction Result of *E.06*

5.1.2.2 K-Means Clustering Implementation

In the previous section, the operation of grouping by calculating Euclidean distance is introduced. Consider that this sub-task is a clustering process, thus K-Means Clustering method can be useful.

Implementation by Calling Scikit-learn Library

Code section 5.4 shows the core code in clustering process. It first creates a K-Means object by Scikit-learn library and then trains the model with coordinates. After that, it predicts the result label for each coordinate.

```
1    # construct a K-Means object
2    kmeans = KMeans(num_classes, random_state=random_seed,
3                    max_iter=50, algorithm='full')
```

```
    # train and predict clustering result
    clusters = kmeans.fit_predict(features)
```

Listing 5.4: Extraction Code Using OpenCV with Tesseract

The previous code section 5.4 indicates that the relationship extraction process is easy to implemented though Scikit-learn library. However, if you have a look at the result from below, a problem occurs, where two rooms are in the same group.

During the initial process of KMeans object, the number of K can be manually set to the number of room, which is identified in the previous task. But I do not set the initial centre points, therefore random initial centre points may cause this.

To solve this problem, centre points are added to the K-Means object. All room coordinates, which are identified in the previous extraction process, are set as the initial centres of K-Means clustering process.

```
-----------------------------------------------------------------
        ID, X-axis, Y-axis, Text,           Type,   Group

        44, 1321.0, 629.84, E.27            ROOM    0
                            ICT ROOM

        45, 1453.1, 764.18, E.28            ROOM    0
                            FEMALE TOILETS

        46, 1532.9, 748.21, ATT/E.44/1      SENSOR  0

        48, 1461.7, 628.28, E/E.44          SENSOR  0

        51, 1461.7, 729.68, E/E.44          SENSOR  0
-----------------------------------------------------------------
```

A Sample of Relationship Extraction Result by Scikit-learn K-Means Clustering(1)

Although the impact of the problem has diminished by setting initial centres, the below results point that it still exists. When group id equals to *6*, two sensors in a group but not belongs to any room. When group id equals to *7*, three rooms are still classified in the same group. Figure 5.12 reveals that there are three rooms close to each other without any other sensors. Thus, the three-room are easily classified in the same group by this type of unsupervised learning algorithm.

```
-----------------------------------------------------------------
        ID, X-axis, Y-axis, Text,           Type,   Group

        41, 908.90, 1205,   (CB-05),        SENSOR, 6

```

```
        42, 1086.2, 1369.7, TG/E.34,          SENSOR, 6

        43, 1095.9, 960.56, E.33
                            STAIR 2,          ROOM,   7

        44, 1106.4, 1090.7, E.31
                            LIFT,             ROOM,   7

        45, 1301.6, 1090.4, E.32
                            CORE 2,           ROOM,   7
-------------------------------------------------------------------
```

A Sample of Relationship Extraction Result by Scikit-learn K-Means Clustering(2)

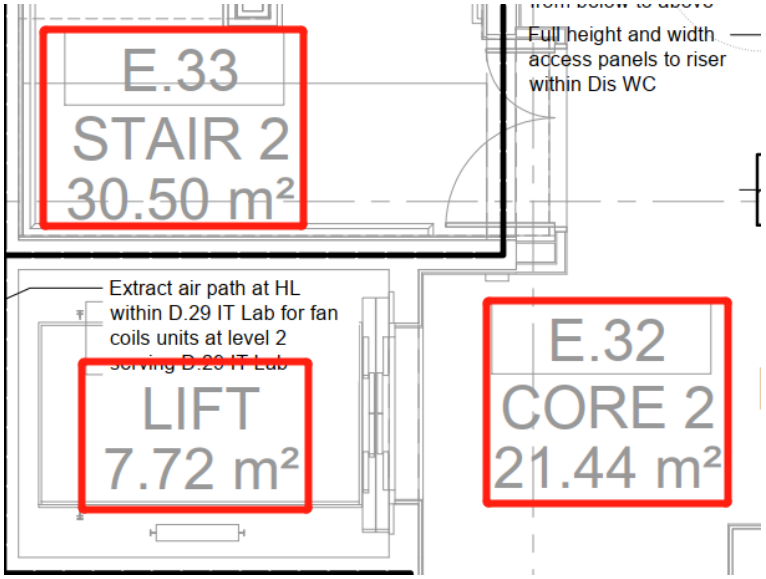


Figure 5.12: *E.31 - E.32 - E.33*

Conclusion of the K-Means Clustering

K-Means Clustering is similar to the former method, where it can use Euclidean distance to calculate the distance between each data points. Even though the K value and the initial centre points are set before clustering, it still makes more significant mistake than the Euclidean distance method. Because it does not divide up every room, but the sub-task requires each room holds a different group id.

5.1.2.3 Auxiliary Relationship Extraction

The above two methods solve the problem of relationship extraction to some extent, but these methods only consider the coordinate distribution but not the actual circumstance. The room has a specific size and scope, where only sensors in or close to the range of the

room should be taken into account. Therefore, the room range can be used as additional information for relationship extraction.

In this section, the room range extraction technology will be introduced here as an auxiliary for relationship extraction. Actually, the whole room range extraction also is an extraction pipeline, but it will not be taken into details.

The Room Range Extraction Pipeline

In this pipeline, most operations are done by calling OpenCV library, especially, the last step use the *cv2.findContours()* method to find contours of rooms. Before that, it will be a series of pre-processing. The following will be the five main steps of the room range extraction pipeline. More details about the last step will be introduced later.

(s1) **Remove text from PDF file.**

(s2) **Convert PDF file to PNG image file.**

(s3) **Remove room doors.**

(s4) **Dilate image to reduce noise.**

(s5) **Erode image to enhance details.**

(s6) **Find contours of rooms.**

The first four steps are called pre-processing. Firstly, I will still use PDFBox with its provide example code to remove all texts on floor plans to make PDF file more clear, which helps to eliminate most noise. The next step is to convert PDF file to PNG image file because the following step requires OpenCV library, where it can accept the image but not a PDF file. Then, the doors of rooms in the image need to be removed, however, it still is a problem to be solved for me so that this step will be ignored during pre-processing. After that, the dilation and erosion process will be taken to reduce noise and improve image quality with the help of OpenCV.

Finally, use OpenCV to get the outlines of all the rooms. More precisely, the following steps are needed to extract the outline of the house.

- **Remove noise left from door removal.** It contains a parameter, which controls the minimal area of blobs to be kept. When the area is less than the threshold, it will be regarded as noise, and then, can be ignored directly.

- **Detect corners.** It contains a parameter, which controls the numbers of detected corners. The bigger the value of this parameter is, the more the room will be removed.

- **Draw lines to close the rooms.** It contains a parameter that indicates the maximum line length to add to close off open doors. It needs to be emphasised that Drawing a line through other existing lines should be avoided.

- **Mark the outside of the house.** The areas outside of the house should be ignored.

- **Find the connected components.** The connected components should be treated as rooms.

Conclusion of the Room Range Extraction Pipeline

When I feed a simple picture of this pipeline, it shows a great achievement. Figure 5.13 is the result after a series of pre-processing, where holds no doors and big noise blocks. After passing it to the room range extraction pipeline, Figure 5.14 shows the result of room segmentation, where all rooms are filled with different colours.

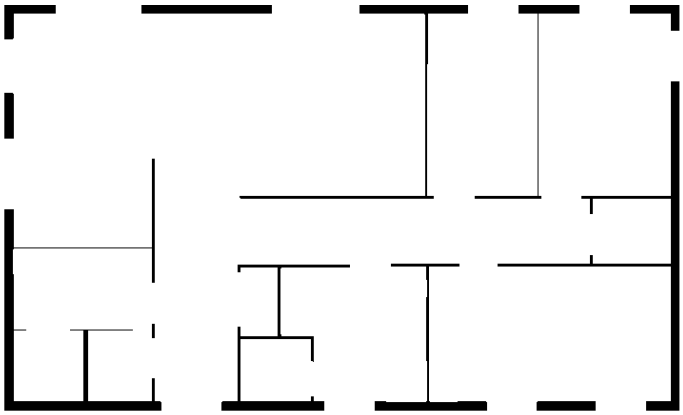


Figure 5.13: A Example of Building Drawing Picture

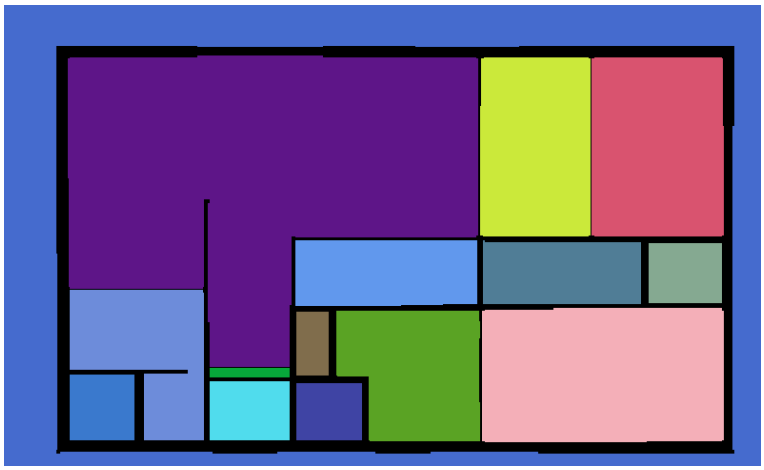


Figure 5.14: The Result of Building Drawing By Taking Room Range Extraction Pipeline

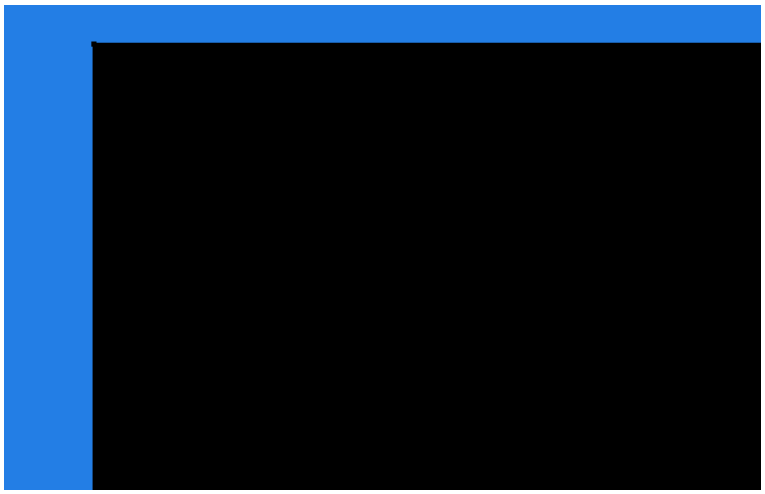


Figure 5.15: A Terrible Result of Room Extraction

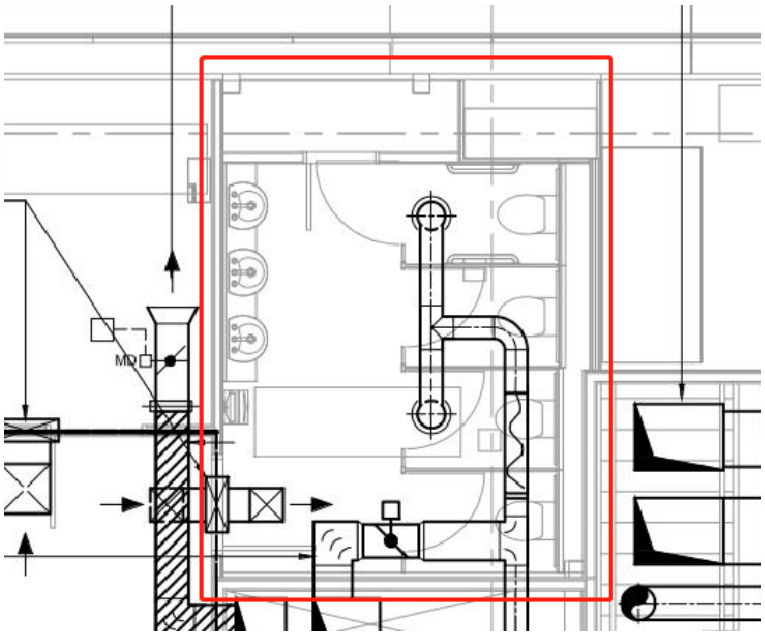


Figure 5.16: A Range of Room: E.28 FEMALE TOILETS

Comparing Figure 5.13 with Figure 5.14, most of the rooms are correctly split and marked with colour in addition to the corridors. Although, it seems that this pipeline works well, when I put the actual building drawings into this pipeline, it looks terrible. As you can see from Figure 5.15, there is a black area on the right bottom side of the picture. Except for that, the blue area indicates the boundary. Why the result looks like nothing, if we go back to the actual building drawing, the problem may be reasonable.

Figure 5.16 indicates room *E.28 FEMALE TOILETS*, where the red square shows its boundary. It is apparent that the walls are not clear or the lines are not thick enough. Besides, there are pipes inside of the room with a darker colour so that it much more obvious than what the wall shows. More importantly, there are too many details inside the room, thus make the room range extraction pipeline hard to do the split job.

Consequently, it is quite difficult to use the room extraction pipeline to divide rooms with complex building drawings.

5.2 Test

In this section, it is about how to identify whether the result, which is extracted from the relation extraction pipeline, is correct or wrong. Only how I define the accuracy will be introduced.

5.2.1 Benchmark and Result

First of all, Table 5.1 gives a part of benchmark result of PDF file *34676-M57-0302-lss7*. It has three columns, containing *text*, *type* and *group*. *type* contains only two types, which are *ROOM* and *SENSOR* respectively. What is more, *group* is a random value, where the same value indicates the same group.

Then, Table 5.2 gives a part of relationship extraction result of PDF file *34676-M57-0302-lss7*. It contains five columns, including x, y, *text*, *type* and *group*. x and y are the coordinate of its corresponding *text*. Apart from that, the rest columns are the same with the columns of Table 5.1.

In order to compare the benchmark and the result, which is extracted by the relationship extraction pipeline, the *ROOM* type needs to be focused. It firstly selects all rows, which the group ID is the same as the one where the type is *ROOM*. Then, if the row in result also in the benchmark, the row will be regarded as correct. Furthermore, if the row in result but not in the benchmark, it will be ignored.

5.2.2 The Accuracy Formula

To make the calculation simple, the accuracy formula is defined by the Equation 5.1:

$$Accuracy = \frac{CC}{ADP} \tag{5.1}$$

where *CC* means the number of rows, which is correctly classified. *ADP* means the number of data rows.

text	type	group
BREAKOUT ROOM 3.5	ROOM	0
SILENT STUDY SPACE 1_30 Workplaces	ROOM	1
SUF-03	SENSOR	1
E/E.26	SENSOR	1
FCU/E/02	SENSOR	1
4 No. S/E.26/3	SENSOR	1

Table 5.1: A Benchmark Result of File *34676-M57-0302-lss7*

5.3 Evaluation

In the previous section, the accuracy is simply defined to assess to what extent the pipeline can correctly extract. However, the accuracy works bad when the data sample is unbalanced. Therefore, in this section, three evaluation methods, which are *Purity*, *Rand Index* and *F-measure* respectively, will be introduced to evaluate the relationship extraction pipeline performance. These methods have already been introduced before so that the following subsections will mainly engage in the implementation.

5.3.1 Purity Method Implementation

In this section, the implementation of purity is introduced. Even though the definition or the formula is different from accuracy metric, they are actually the same after taking into implementation. For each classified cluster, select the max purity value and then treat it as the purity of its cluster. Finally, put all clusters' purity value together to calculate the purity of the whole relationship extraction pipeline.

5.3.1.1 Purity Code

Code section 5.5 indicates the function of purity implemented by Java. Where *resultCoordinateTexts* is an instatnce of Class *CoordinateText*, which contains a text, the coordinates of the text, the type of the text, and the group ID of the text.

x	y	text	type	group
-12.652222	991.52	BREAKOUT ROOM 3.5	ROOM	0
149.38501	672.68005	E/D.29	SENSOR	0
-9.466431	1751.54	SILENT STUDY SPACE 1_30 Workplaces	ROOM	1
215.83435	1630.88	E/E.26	SENSOR	1
107.13391	1736.48	SUF-03	SENSOR	1
229.2633	1781.6001	FCU/E/02	SENSOR	1
435.86438	1798.16	4 No. S/E.26/3	SENSOR	1
28.776611	2355.8	6No. S/E.12/2	SENSOR	1

Table 5.2: A Room Extraction Result of File *34676-M57-0302-lss7*

```
/**
 * Calculating the Purity value
 * of relationship extraction pipeline
 *
 * @return The value of Purity
 */
public double getPurity() {

    AtomicReference<Double> allPurity
                        = new AtomicReference<>(0.0);
    resultCoordinateTexts
        .stream()
        .collect(
            Collectors.groupingBy(CoordinateText::getGroupId)
        ).forEach((k, vs) -> {
            int matchNum = 0;
            int benchmarkGroupId = -1;
            for (CoordinateText v : vs) {
                String name = v.toString();
                // obtain the groupId in benchmark set
                if (v.getType() == Type.ROOM) {
                    benchmarkGroupId
```

```
                        = benchmarkCoordinateTexts.stream()
                            .filter(
                                coordinateText ->
                                  coordinateText.toString()
                                        .equals(name)
                            ).findFirst()
                            .get()
                            .getGroupId();
                    continue;
                }

                int finalBenchmarkGroupId = benchmarkGroupId;
                boolean anyMatch
                    = benchmarkCoordinateTexts
                        .stream()
                        // same groupId && same text
                        .anyMatch(
                            coordinateText ->
                              coordinateText.getGroupId()
                                    == finalBenchmarkGroupId
                              && coordinateText.toString()
                                              .equals(name)
                        );
                if (anyMatch) {
                    ++matchNum;
                }
            }
            double tmpPurity = matchNum * 1.0
                                / resultCoordinateTexts.size();
            allPurity.set(allPurity.get() + tmpPurity);

        });
    return allPurity.get();
}
```

Listing 5.5: A Function of Calculating Purity

5.3.2 Rand Index Method Implementation

The next evaluation method is the Rand Index, where it compares the benchmark result with the extracted results to measure the similarity of them. By counting the number of the appeared same status to obtain the value of Rand Index. Equivalently, the implementation

is still our focus.

5.3.2.1 Rand Index Code

Code section 5.6 shows the function of Rand Index implemented by Java. This function is easy to implement because the formula itself is simple. However, it is not easy to control the text results, which it extracted from PDF file, so that some useless texts may be also extracted. Thus, some texts, which come from the extracted result but do not appear in the benchmark result set, should be ignored. Ideally, both the extracted result set and benchmark set should hold the same number of rows. Actually, in a real world scenario, make sure that the number of benchmark result set is greater than or equal to the number of extracted result set.

```
1
2
3  /**
4   * Calculating the Rand Index value of
5   * relationship extraction pipeline.
6   *
7   * @return The value of RandIndex
8   */
9  public double getRandIndex() {
10
11     assert benchmarkCoordinateTexts.size()
12                         >= resultCoordinateTexts.size();
13
14     int sames = 0;
15     int coordinateSize = benchmarkCoordinateTexts.size();
16     // Iteration
17     for (int i = 0; i < coordinateSize - 1; ++i) {
18         for (int j = i + 1; j < coordinateSize; ++j) {
19             boolean xStatus = getStatus(
20                         resultCoordinateTexts.get(i),
21                             resultCoordinateTexts.get(j));
22             boolean yStatus = getStatus(
23                         benchmarkCoordinateTexts.get(i),
24                             benchmarkCoordinateTexts.get(j));
25
26             sames += (xStatus == yStatus) ? 1 : 0;
27         }
28     }
29     // Rand Index Formula
```

```
30      return sames * 2.0
31                  / (coordinateSize * (coordinateSize - 1));
32  }
33
34
35
36  /**
37   * Judge if id of group are the same or not.
38   *
39   * @param x instance of CoordinateText object
40   * @param y instance of CoordinateText object
41   * @return if id of groups are the same return {@code True},
42   * otherwise, return {@code False}
43   */
44  public boolean getStatus(CoordinateText x, CoordinateText y) {
45      return x.getGroupId() == y.getGroupId();
46  }
```

Listing 5.6: A Function of Calculating Rand Index

5.3.3 F-measure Method Implementation

The last assessment method is the F-measure method. It is a little bit sophisticated than previous two ways. Firstly, each cluster in extracted result set should calculate its F-measure value, and then put them together with its weight to calculate the ultimate F-measure value. During the calculation of each clusters' F-measure value, each cluster in extracted result set should do the same calculation process with all clusters in benchmark result set, where the max F-measure value will be regard as the F-measure of the specific cluster in extracted result.

5.3.3.1 F-measure Code

Code section 5.7 gives the function of F-measure method implemented by Java. This function require the calculation of precision and recall value each cluster in extracted result set so that it spends much more time in calculation than previous two methods.

```
1  /**
2   * Calculating the F-measure value
3   * of relationship extraction pipeline
4   *
5   * @return The value of F-measure
6   */
```

```
7  public double getFMeasure() {
8
9      double fMeasure = 0.0;
10
11     int m = (int) benchmarkCoordinateTexts.stream()
12             .map(CoordinateText::getGroupId)
13             .distinct()
14             .count();
15     int n = resultCoordinateTexts.size();
16
17     for (int j = 0; j < m; ++j) {
18         int finalJ = j;
19         // find each group in benchmark set
20         List<CoordinateText> bCoordinateTexts
21                 = benchmarkCoordinateTexts.stream()
22                     .filter(
23                         coordinateText -> coordinateText
24                                 .getGroupId() == finalJ
25                     ).collect(Collectors.toList());
26         long pJ = bCoordinateTexts.size();
27
28         // find ROOM in each benchmark group
29         CoordinateText roomCoordinateText
30                 = bCoordinateTexts.stream()
31                     .filter(
32                         coordinateText -> coordinateText
33                                 .getType() == Type.ROOM
34                     ).findFirst()
35                     .get();
36
37         // find group id in result set
38         int rGroupId = resultCoordinateTexts.stream()
39                 .filter(
40                     coordinateText -> coordinateText.toString()
41                         .equals(roomCoordinateText.toString())
42                 ).findFirst()
43                 .get()
44                 .getGroupId();
45
46         // find the same group in result set
47         List<CoordinateText> rCoordinateTexts
```

```
                = resultCoordinateTexts.stream()
                    .filter(
                        coordinateText -> coordinateText
                                .getGroupId() == rGroupId
                    ).collect(Collectors.toList());
        int cI = rCoordinateTexts.size();

        // compare and calculate F-measure_j
        long pjci = rCoordinateTexts.stream()
                .filter(
                    rCoordinateText -> bCoordinateTexts.stream()
                        .anyMatch(
                            bCoordinateText -> bCoordinateText
                            .toString()
                            .equals(
                              rCoordinateText.toString()
                            )
                        )
                ).count();

        // precision && recall
        double precision = pjci * 1.0 / cI;
        double recall = pjci * 1.0 / pJ;

        // F-measure
        double tmpFMeasure = 2 * precision * recall
                                / (precision + recall);

        // weight
        double wJ = pJ * 1.0 / n;

        // accumulative F-measure
        fMeasure += tmpFMeasure * wJ;
    }

    return fMeasure;
}
```

Listing 5.7: A Function of Calculating F-measure

5.3.4 Evaluation Methods Results

```
+-------------------------------------------------------------+
|                       Extraction Result                     |
+----------------------+--------------------------------------+
| Extracted PDF File:  |    34676-M57-0302_Iss7.pdf           |
+----------------------+------------+------------+------------+
|       Accuracy       |   Purity   | Rand Index | F-measure  |
+----------------------+------------+------------+------------+
|        63.77%        |   0.6377   |   0.9165   |   0.8430   |
+----------------------+------------+------------+------------+
```

The Results of Evaluation methods

The above shows the results from the evaluation methods. The *Accuracy* and *Purity* the same, where it has been told before. The *Rand Index* and *F-measure* give much higher rating of this relationship extraction pipeline than *Accuracy* and *Purity*.

5.4 Relationship Extraction Pipeline Assembly

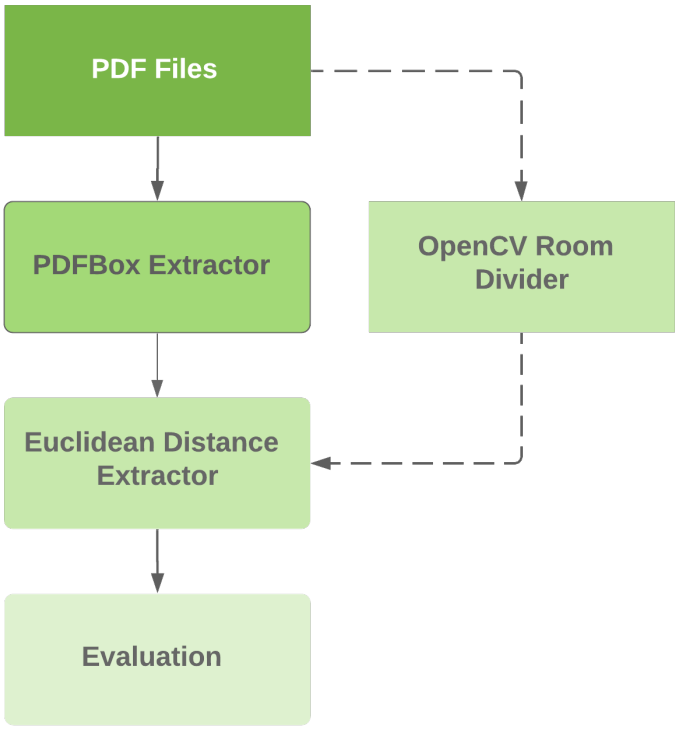


Figure 5.17: Relationship Extraction Pipeline Structure

There are three text extraction methods, two relationship extraction methods, one auxiliary method, as well as three evaluation methods, have been introduced in the earlier section. In this section, to assemble a complete extraction pipeline, which also includes evaluation, it requires to choose the best text extraction and the most suitable relationship extraction method.

5.4.1 Relationship Extraction Pipeline Structure

Figure 5.17 indicates the final relationship extraction pipeline. It starts with a PDF file as its input. And then, it turns out two ways, where one is text and coordinate extractor, and the other is optional. On the extractor side, the PDFBox, which almost meets my needs, is my ultimate choice. On the other side, it is an auxiliary method that can extract the range of each room, therefore, it could be a position limitation when clustering the texts. Next, the PDFBox Extractor is followed by the Euclidean Distance Extractor. K-Means Clustering Extractor is given up because it can not divide each room into a single group, where it makes the evaluation impossible. Meanwhile, if the room range information is provided by the OpenCV Room Divider, it will be mixed into the process of relationship extraction to make the extraction result more precise. Eventually, the final extracted results will be put into three different evaluation methods to identify the extent of its performance.

5.4.2 Linking Between Extractors

Before the text and coordinate results are sent to the next extractor, two steps need to be taken.

- Merge room text
- Remove useless text

The first step is to solve a problem, which has already mention in previous sub-section 5.1.1.2. Figure 5.8 indicates a situation that some text should be merged together so as to describe a specific room. Not only the PDFBox but also the PDFMiner has this problem. The below is a sample that needs to be combined together.

```
-------------------------------------------------------------
       ID,       X-axis,       Y-axis,       Text

       60,       532.86,       1009.6,       BREAKOUT ROOM 3.7

       61,       533.06,       977,          E.06

       62,       534.25,       1069.8,       8 Seats
-------------------------------------------------------------
```

To solve this problem, two parameters, which are *tolerance_x* and *tolerance_y* individually, are introduced. When the coordinates of the text are not too far apart, it is considered a description of the same room. More specifically, when the difference between the abscissa is less than or equal to the *tolerance_x* and the difference between the ordinates is less than or equal to the *tolerance_y*, these texts should be merged together. By default, *tolerance_x* is *10* and *tolerance_y* is *100*.

Besides, the merged text should obey the visual order, hence these texts will be sorted by y-axis value. Lastly, the connected text replace CRLF [2] with a slash in order to facilitate storage and display. What is more, the coordinate of text also should change, where it simply use the mean-value coordinate of them.

At the end, the result of above sample will be shown as below.

```
-------------------------------------------------------------
ID,     X-axis,     Y-axis,     Text

61,     533.39,     1018.8,     E.06_BREAKOUT ROOM 3.7_8 Seats
-------------------------------------------------------------
```

The second step is to remove the useless text, which is not a part of description for rooms or sensors, or not appear in the benchmark result set.

[2]The term CRLF refers to Carriage Return (ASCII 13, \r) Line Feed (ASCII 10, \n). They're used to note the termination of a line, however, dealt with differently in today's popular Operating Systems.

Chapter 6

Results and Discussion

In this chapter, the findings, goals achievements and some further works will be introduced .

6.1 Findings

6.2 Goals Achieved

Table 6.1 shows the goals achievements, where all functions have been implemented by Java or Python no matter the necessity is *Mandatory* or *Optional*. Also, an improved function, which is used to split rooms and obtain its range, named *OpenCV Room Divider* also has been implemented. However, it works terrible for the building drawings.

6.3 Further work

Mainly, increase the performance of the relationship extraction pipeline should be a vital work in the future. One is to integrate the OpenCV Room Divider into the final pipeline. Other is to try various clustering method to improve accuracy.

6.3.1 Improvement in OpenCV Room Divider

Even though the room range divider has already been implemented, it still works not ideally for the building drawings. After observation, the main reason is the drawings themselves, where the lines of rooms are not apparent sufficiently so that the divider is not sensitive enough to obtain all range of rooms. What is more, there are many irrelevant lines representing facilities and it will interfere with the rooms segmentation.

6.3.2 Improvement in Relationship Extraction

Function	**Sub-function**	**ID**	**Necessity**	**Completeness**
Text and coordinate Extraction	PDFBox Extractor	1.1	Mandatory	Fully Achieved
	PDFMiner Extractor	1.2	Optional	Fully Achieved
	OpenCV Extractor	1.3	Optional	Fully Achieved
Relationship Extraction	Euclidean Distance	2.1	Mandatory	Fully Achieved
	K-Means Clustering	2.2	Optional	Fully Achieved
	OpenCV Room Divider	2.3	Improvement	Partially Achieved
Evaluation	Purity metric	3.1	Mandatory	Fully Achieved
	Rand Index metric	3.2	Optional	Fully Achieved
	F-measure metric	3.3	Optional	Fully Achieved

Table 6.1: The Function Achievements

Chapter 7

Conclusions

To conclude, a series of experiments show that it is possible to extract the relationship between rooms and sensors from the building drawings with the extraction pipeline up to now. Besides, when the pipeline is assembled finally, it might be a risk in applying different program language, which is caused by different tools. In the next step of this work, we will try to focus on increasing the performance of this pipeline, especially using OpenCV tool to do the boundaries detection, so that the error rate could be reduced.

Bibliography

Fujino, A., Isozaki, H., and Suzuki, J. (2008). Multi-label text categorization with model combination based on f1-score maximization. In *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-II*.

Huang, Z. (1998). Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data Mining and Knowledge Discovery*, 2(3):283–304.

Jia, R., Jin, B., Jin, M., Zhou, Y., Konstantakopoulos, I. C., Zou, H., Kim, J., Li, D., Gu, W., Arghandeh, R., Nuzzo, P., Schiavon, S., Sangiovanni-Vincentelli, A. L., and Spanos, C. J. (2018). Design automation for smart building systems. *Proceedings of the IEEE*, 106(9):1680–1699.

Kay, A. (2007). Tesseract: an open-source optical character recognition engine. *Linux Journal*, 2007(159):2.

Lee, H., Malaspina, D., Ahn, H., Perrin, M., Opler, M. G., Kleinhaus, K., Harlap, S., Goetz, R., and Antonius, D. (2011). Paternal age related schizophrenia (pars): Latent subgroups detected by k-means clustering analysis. *Schizophrenia Research*, 128(1-3):143–149.

MacQueen, J. et al. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA.

Na, S., Xumin, L., and Yong, G. (2010). Research on k-means clustering algorithm: An improved k-means clustering algorithm. In *2010 Third International Symposium on Intelligent Information Technology and Security Informatics*, pages 63–67. IEEE.

Nimlyat, P. S. (2018). Indoor environmental quality performance and occupants' satisfaction [ieqpos] as assessment criteria for green healthcare building rating. *Building and Environment*, 144:598–610.

Omarov, B., Altayeva, A., and Cho, Y. I. (2017). Smart building climate control considering indoor and outdoor parameters. In Saeed, K., Homenda, W., and Chaki, R., editors, *Computer Information Systems and Industrial Management*, pages 412–422, Cham. Springer International Publishing.

PDFBox®, A. (2010). Apache pdfbox® - a java pdf library. *The Apache Software Foundation*.

Pulli, K., Baksheev, A., Kornyakov, K., and Eruhimov, V. (2012). Realtime computer vision with opencv. *Queue*, 10(4):40–56.

Rand, W. M. (1971a). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850.

Rand, W. M. (1971b). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850.

Rice, S. V., Jenkins, F. R., and Nartker, T. A. (1995). The fourth annual test of ocr accuracy. Technical report, Technical Report 95.

Sanderson, M. (2010). Christopher d. manning, prabhakar raghavan, hinrich schütze, introduction to information retrieval, cambridge university press 2008. isbn-13 978-0-521-86571-5, xxi + 482 pages. *Nat. Lang. Eng.*, 16(1):100–103.

Shinyama, Y. (2015). Pdfminer: Python pdf parser and analyzer (2010). *Cited on*, 13.

Sripada, S. C. and Rao, M. S. (2011). Comparison of purity and entropy of k-means clustering and fuzzy c means clustering. *Indian journal of computer science and engineering*, 2(3):343–346.

Tadokoro, S., Jia, Q.-S., Zhao, Q., Darabi, H., Huang, G., Becerik-Gerber, B., Sandberg, H., and Johansson, K. H. (2014). Smart building technology [tc spotlight]. *IEEE Robotics & Automation Magazine*, 21(2):18–20.

Vincent, L. and Lead, U. T. (2007). Announcing tesseract ocr. *Google Code Blog, August 2006*, 31.

Wagstaff, K., Cardie, C., Rogers, S., Schrödl, S., et al. (2001). Constrained k-means clustering with background knowledge. In *Icml*, volume 1, pages 577–584.

Young, G. and Householder, A. S. (1938). Discussion of a set of points in terms of their mutual distances. *Psychometrika*, 3:19 – 22.

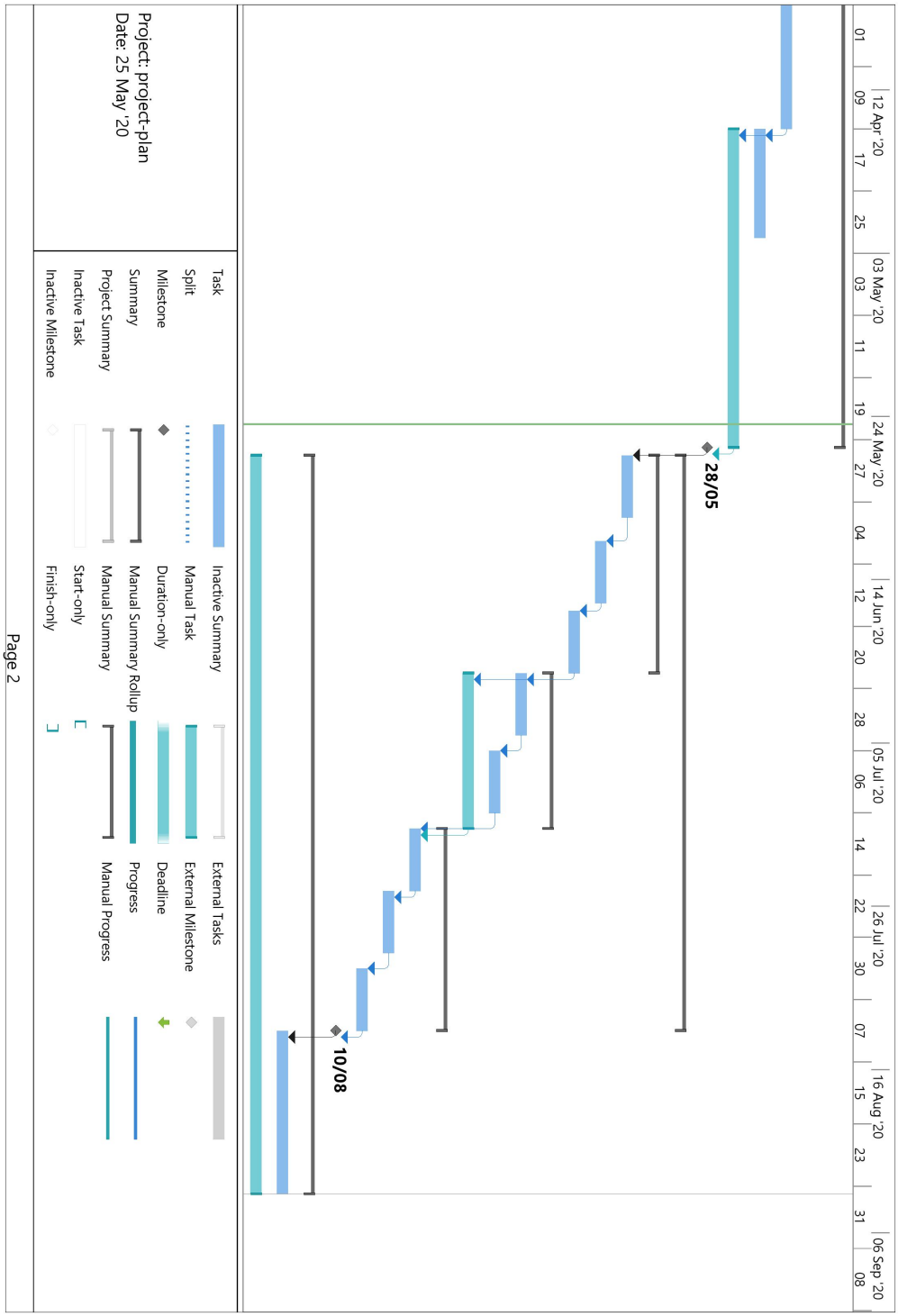
Zhou, X., Yao, C., Wen, H., Wang, Y., Zhou, S., He, W., and Liang, J. (2017). East: an efficient and accurate scene text detector. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 5551–5560.

Appendices

Appendix A

An Appendix of Project Gantt Chart

ID	Task Mode	Task Name	Duration	Start	Finish
1		**Initiating**	**64 days**	**28 Feb '20**	**28 May '20**
2		Identify objectives	19 days	28 Feb '20	25 Mar '20
3		Identify resources	15 days	27 Mar '20	16 Apr '20
4		Create a project plan	10 days	17 Apr '20	30 Apr '20
5		Write background report	29 days	17 Apr '20	27 May '20
6		Preparation completion	0 days	28 May '20	28 May '20
7		**Execution**	**52 days**	**29 May '20**	**10 Aug '20**
8		**Text and coordinate extraction**	**20 days**	**29 May '20**	**25 Jun '20**
9		PDFMiner	6 days	29 May '20	05 Jun '20
10		PDFBox	6 days	09 Jun '20	16 Jun '20
11		OpenCV OCR with tesseract	6 days	18 Jun '20	25 Jun '20
12		**Relationship extraction**	**14 days**	26 Jun '20	15 Jul '20
13		Euclidean distance	6 days	26 Jun '20	03 Jul '20
14		K-Means clustering	6 days	06 Jul '20	13 Jul '20
15		Boundaries extraction	14 days	26 Jun '20	15 Jul '20
16		**Evaluation**	**18 days**	16 Jul '20	10 Aug '20
17		Purity	6 days	16 Jul '20	23 Jul '20
18		Rand Index	6 days	24 Jul '20	31 Jul '20
19		F-measure	6 days	03 Aug '20	10 Aug '20
20		Work completion	0 days	10 Aug '20	10 Aug '20
21		**Deployment**	**67 days**	29 May '20	**31 Aug '20**
22		Assemble extraction pipeline	15 days	11 Aug '20	31 Aug '20
23		Write dissertation report	67 days	29 May '20	31 Aug '20


12 Apr '20
03 May '20
24 May '20
14 Jun '20
05 Jul '20
26 Jul '20
16 Aug '20
06 Sep '20
28/05
10/08
Project: project-plan
Date: 25 May '20
Task
Split
Milestone
Summary
Project Summary
Inactive Task
Inactive Milestone
Inactive Summary
Manual Task
Duration-only
Manual Summary Rollup
Manual Summary
Start-only
Finish-only
External Tasks
External Milestone
Deadline
Progress
Manual Progress
Page 2