UMEÅ UNIVERSITY
Department of Computing Science
Master Thesis, D 20p
Emil Gunnarsson, c00egn@cs.umu.se

23th April 2006

Automatic blueprint registration software

By Emil Gunnarsson

Internal Supervisor:
Michael Minock

External Supervisor:
Bo Johansson, Referat Fastighetsinformation AB

Abstract

The company Referat Fastighetsinformation AB registers data found on blueprints. The data needs to be manually entered into a database and as this task is very time demanding, a solution to decrease the time for each registered blueprint is desired. The purpose of this project is to evaluate and create a software product that performs automatic category classification. Automatic category classification is the task to read information from the blueprint and enter it to the database. The focus has been on recognizing text and to classify it to its correct category. There are many different methods to perform Optical Character Recognition (OCR) and a selection of the most common ones has been studied. Tests show that the developed product performs OCR well when the text is of good quality, but is sensitive to noise. Automatic category classification proved to be a very difficult task and a semi-automatic implementation where the user manually selects a category for the recognized text is desirable.

Table of contents

1 Introduction

This paper is part of a Master Thesis 20p project in Computing Science at Umeå University. The task is to create an Optical Character Recognition prototype software that automatically reads and classifies text in blueprints.

1.1 Company

The Master Thesis is being made for Referat AB. The company is located in Jönköping, Sweden, and includes a temporary manned office in Stockholm. The activity started in 1994 and the company has eight employees. The company is owned by Bo Johansson Projektteknik AB, which in turn is owned by Bo Johansson (90%) and the employees (10%).

Referat Fastighetsinformation AB deals with digital document management within the construction industry and carries on information projects.

Areas of work

The company motto "Från kaos till progressiv utveckling (From chaos to progressive development)" refers to the current situation in a major part of the construction industry information management. The view on the technical development is based on the ambition to achieve a good result for the customers. This result causes involvement in the development of:

- industry standards
- usage of new technology (without forcing on the customers untested technology)
- usage of IT-tools and enhance where tools are missing or does not give the right function
- organize information by data-layer principles.

Product and service-range

1. Document management systems with the product FDOK4 developed in cooperation with Litium Affärsutveckling AB.
2. Scanning (many different formats)

3. CAD (quality check-up, digitalized objectoriented and 3D, visualization) Area-measuring and inspection of the information correctness with the reality.
4. Registration of metadata for documents with a company developed, web-based application.
5. Operates and participates in development projects for the customer and industry development
6. Creates and designs databases for documents and organizes/performs assignments.
7. Requirement for digital management information

Products

Referat Fastighetsinformation AB is a service-company that besides document management operates development projects where Landstingsfastigheters PTR "Program för Teknisk Standard" has resulted in a database information system with a web-based Graphical User Interface and program-modules for input, information search, reports etc.

1.2 Goals

The company wants a program that is able to speed up the process of register blueprints. Each blueprint is scanned and then metadata found on the blueprint is stored in a database. Metadata that is looked for is scale, date, name, category etc.

Figure 1.1 shows a table with metadata of an arbitrary blueprint. Unfortunately, different architects use different ways of displaying this data. Cells can be placed in any order, and the cells may also be named a bit different between blueprints. It's also quite common that some cells aren't named. There are other problems as well, such as different fonts. Fonts vary among blueprints and multiple fonts are often used in the same blueprint.

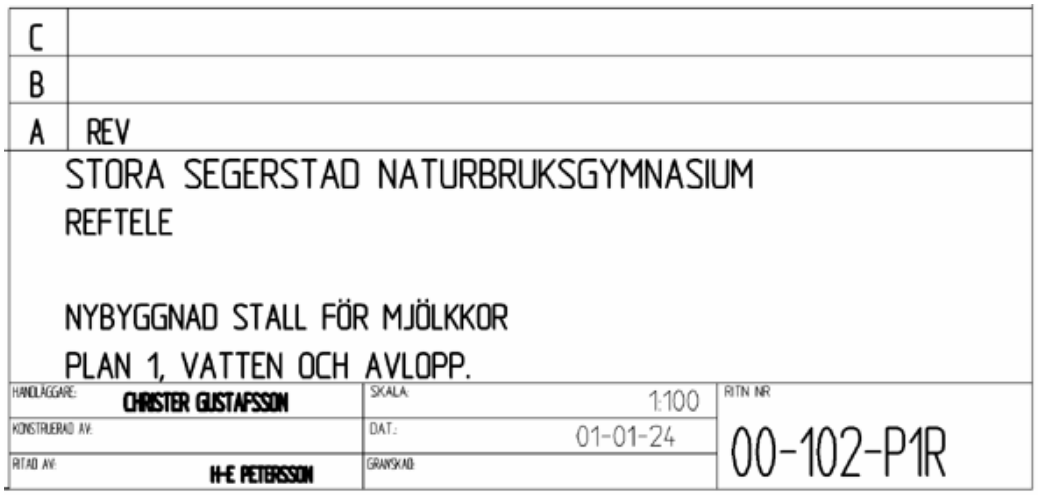


Figure 1.1. Blueprint registration data

The product should not only be an Optical Character Recognition software but also able to classify the words into different categories. This is a complex task since all words found on the blueprint are not relevant and all cells are not named.

1.3 Development tools

There are several aspects when it comes to decide which software development environment that should be used. The developed product should be able to read text from an image and classify it to different categories. As this is a complex task the program needs to be able to perform calculations very rapidly. The more images the program can process every minute the better. Top priorities are therefore speed and ease of application enhancement.

With this in consideration the program language *C++* is chosen for its speed in computation and the objectoriented structure. The program will be subdivided into small parts (objects) that each handles a specific task. By using an objectoriented language, changes can easily be made to the source code without reconstructing major parts of code.

The final product will only be used on *Microsoft Windows* platforms, and therefore will *Visual C++* and *MFC* be used as the programming environment.

1.4 Prototype

The timeframe makes it impossible to make a finished product ready for use. The focus will be to create a foundation with many parts that each individually can be developed further in the future. All the major parts of the program will be included in the prototype, such as Optical Character Recognition and classification algorithms.

1.5 Restrictions

Since the program is a prototype software for a commercial company, which may be used in the future, the source code will not be included in this paper, nor will it be available for download.

2 OCR Techniques

A description of what Optical Character Recognition (OCR) is follows along with brief overviews of different techniques for OCR.

2.1 Optical Character Recognition

The goal with OCR is to recognize machine printed text. The algorithms have become faster and more accurate and techniques to read handwritten text are now a more challenging problem. Most algorithms assume that the text is printed on a uniform background with a high contrast between the text and the background. If this is not the case many algorithms fail. OCR is one of the most successful areas of pattern recognition [5]. There are many different techniques to perform OCR. There are however some fundamental issues all of the techniques have to deal with in some way or another. These are shown in figure 2.1. Different algorithms may perform the steps in different orders.

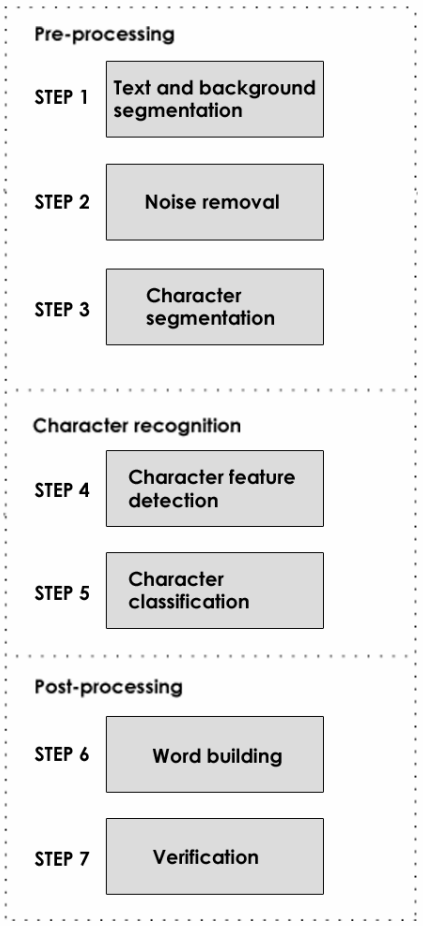


Figure 2.1 The different steps of an OCR algorithm

2.1.1 Segmentation of text and background

Many algorithms need the text to be printed on a uniform background with a high contrast between the text and the background. The accuracy of the OCR algorithms on a non-uniform background is much lower than on a uniform background [6].

It is much more complex to segment text and background in a non-uniform background image. There are however many algorithms that address this problem, for example Tsai [7] who presents an algorithm that extracts luminance and saturation features that are used for segmentation.

It is important to decide which pixels are considered as background and which are considered as text. For a grey-scale image this can easily be done with a

threshold function. One of the best has proved to be Niblack's threshold algorithm [8], which computes a local mean and a local standard deviation in the neighborhood and uses this in the threshold-function.

A simpler threshold function is to assign T to a value that is used as a threshold. All pixels below T will be considered as background and all pixels above will be set as text. Many algorithms then convert the image to a binary image (each pixel is either white or black). Only one bit is needed to the store color information for each pixel [6].

There are several methods that process the image even further. For example if the font size is known segments of connected pixels of unreasonable sizes may be removed.

2.1.2 Noise removal

Some of the pixels in the previous step will most likely be put in the wrong group. There are several algorithms that focus on identifying and removing noise (small blobs of data). Two popular methods for identifying and removing noise are morphological operators and neighborhood operations [6].

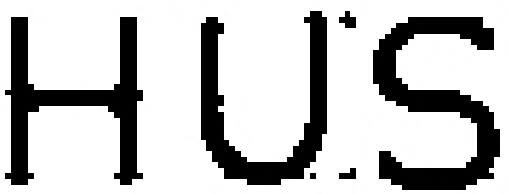


Figure 2.2. Image with noise

An example of noise is shown in figure 2.2. The dots between the characters U and S are not relevant and should be classified as noise and thus be removed.

2.1.3 Segmentation of words or characters

In the beginning of OCR many algorithms focused on isolating individual characters and then used character recognition algorithms to interpret them. Many new algorithms instead find character regions and perform character recognition directly in the region before performing character segmentation [6].

Words and characters can be segmented by using histograms. How this works is explained more in detail in the chapter "Techniques".

2.1.4 Character feature detection

Feature detection and classification are the two stages most people consider OCR to be. They are also the most important ones. Feature detection algorithms detect features in a character. Early algorithms of OCR usually looked at the whole character as one big feature (template matching), while newer algorithms often tries to find sub-features within the character [6]. Features looked for by these algorithms are open areas, closed shapes, diagonal lines, horizontal and vertical lines, line intersections etc. Features invariant to font, size and other small variations are preferred.

2.1.5 Character classification

Classification is the process of deciding which character best matches the features found in the character [6]. Feature detection is a very important step, and without it OCR would not work. But it also needs a solid classification algorithm.

Some characters have the same features as both uppercase and lowercase. In such a situation the whole word the character is part of may be used to decide whether it is uppercase or lowercase.

2.1.6 Word building

This is the last stage of OCR. The characters recognized by the OCR algorithm form words. A way to verify that the characters actually are correct is to match them against words in a dictionary.

If the word does not exist, it is likely that one or more characters have been classified as a wrong character. The word may be discarded, left as it is or corrected by using a correction algorithm.

It is not necessary to verify that the words are correct, but the method makes the OCR algorithm more robust as it is less sensitive to single character errors.

2.2 Techniques

There are two basic types of methods used for OCR. The most basic is Template matching algorithms and the more complex type is Feature detection algorithms. A more detailed description of Template matching and a selection of Feature detection algorithms follow. These techniques presented in this chapter pertain to step 4 and step 5 in figure 2.1.

2.2.1 Template matching

Template matching does not look for sub-features and the whole character itself is seen as one feature. The character Z is matched against each template T_i and a similarity or dissimilarity value is computed. Template T_k with the highest similarity (or lowest dissimilarity) value is identified. If template T_k is greater than a predefined threshold value, character Z is classified as T_k. A common dissimilarity function is the *mean square distance* [10].

$$D = \sum_{i=1}^{M} (Z(x_i, y_i) - T(x_i, y_i))^2$$

M = total amount of pixels in the image

Mean square distance function assumes that Z and T_i are of the same size. A solution to handle different sizes is to scale each character to the size of the template. This method may cause some loss of data. Template matching is very sensitive to other variations as well, especially askew images.

A method to increase robustness was proposed by Tubbs [12] by using weights for each pixel. A trainingset of templates is used to determine the weights. A trainingset of templates is used to compute these weights. A simple weight-function is to calculate the number of times a certain pixel was set as a text-pixel for all the templates and then divide it by the total number of templates [1].

Another way to increase robustness may be by using many sets of the same character and match against each one of these. Each set differs slightly from the other, which makes the algorithm less sensitive to variations. This does however decrease the speed of the recognition.

2.2.2 Neural networks

The method described here is Backpropagation Neural Networks. A Backpropagation network consists of several layers on neurons: the input layer, one or more hidden layers and an output layer. Each layer consists of an arbitrary number of neurons.

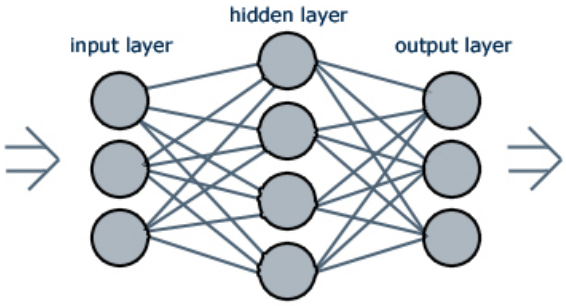


Figure 2.3. Example of a neural network

Each node in the different layers is interconnected to nodes in the next layer via weighted links. The input layer retrieves some values from the image. Then output values are computed and passed on to the next layer of nodes, which in turn computes their own output value and passes it on. This continues through all the layers until the output layer is reached and an output value is computed [9].

Neural networks uses training sets to decide how the neurons should compute values. Since the output for every input is known, and error for every pass through the network can be calculated. It then backpropagates through the network to adjust link values so the network will produce better result next time. This process is called "training" [9]. It is possible to over learn the network and it will then be too strict and not accept any variations whatsoever.

2.2.3 Unitary image transformations

A unitary transformation is applied to the character image. A trainingset is used to determine the amount of variance for each pixel. The pixels are ordered by highest variance and the ones with most variance are used as features. This way only the pixels with high variance among all templates are used for classification. There are many different unitary transformations that can be applied. One of the most used is the Karhumen-Loeve (KL) [1].

Like template matching, unitary image transformation are not rotation-invariant and size of the character must be equal to size of the template.

For all unitary image transformations an inverse function exists to recreate the original character [1].

2.2.4 Projection histograms

From the beginning it was used as an OCR technique, but nowadays it is more common to use it for line, word and character segmentation. Projection is made along the x-axis and the y-axis separately (horizontal and vertical histograms) as shown in figure 2.4. The amount of pixels for each row (and each column) is calculated and stored (called *bins*).

A simple function for comparison of two histograms is:

$$d = \sum_{i=1}^{n} | y_1(x_i) - y_2(x_i) |$$

Where y_1 and y_2 are histograms, x_i is the bin value for column i, n is the total amount of bins and d is the sum of all errors.

This function, however, will be very sensitive to rotation and other variations. A better way to do this would be to compare *cumulative histograms* [1]. Instead of comparing each bin, the first k bins are summed. The function then compares these sums. This algorithm is less sensitive to rotation and other small variations.

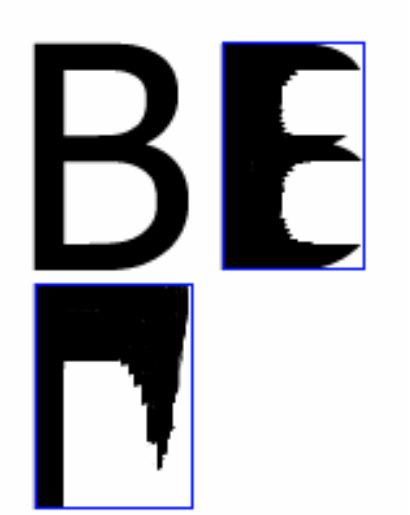


Figure 2.4 Vertical and horizontal histograms of the character B

As mentioned earlier, histograms may also be used for word and character segmentation. Two consecutive characters will have a few empty bins between them. Empty bins (or almost empty if the image contains noise) is used for segmentation. The same algorithm is used to segment words and lines of text.

2.2.5 Zoning

Each character is divided into different zones. A grid of size n x m can describe these zones.

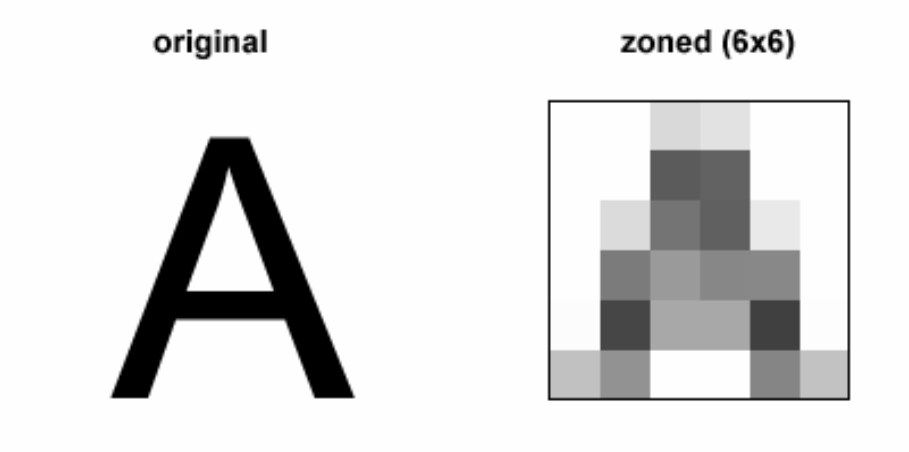


Figure 2.5 Example of zoning of the character A

For each cell an average grey level value is computed over all pixels contained [1]. Other attributes may be computed for the zones such as density and line-approximation. These zones are used as features in the recognition algorithm.

2.2.6 Geometric moment invariants

Moment invariants are properties of a region of an image and can be invariant to translation, scale and rotation. These describe the image in a small set of attributes and these attributes can be used in recognition algorithms.

Hu presented *absolute orthogonal moment invariants* which are invariant to translation, scale and rotation [2], and it has been widely used [1].

2.2.7 Zernike moments

Zernike moments are a class of orthogonal moments of an arbitrary order [3]. Higher order moments are more detailed but also more sensitive to noise variations in the image. Zernike moments are defined over the interior of a unit circle [3].

Zernike moments have many desirable properties, such as rotation invariance, robustness to noise and fast computations [11]. However, the Zernike moments are not invariant to scale and translation, meaning the character must be normalized before the moments are calculated. This is the major drawback of the Zernike moments as this involves re-sampling of the image and loss of data [11].

2.2.8 Spline curve approximation

First the outer contour of the character needs to be found. Breakpoints are identified on the contour (points with high curvature). Spline curve approximations between the breakpoints are calculated and the breakpoints and parameters of the spline approximation are used for matching [1].

2.3 Implementation

A lot of different techniques have been used to be able to perform OCR and classify the text into categories. Descriptions of the most important algorithms that are used are presented here.

2.3.1 Background and text segmentation

This is the first step in figure 2.1. It is very important that it is clear which pixels are background pixels and which are not. A threshold function is used to separate text (data) and background pixels. Every pixel P_i is processed and if the grey-scale value of P_i is less than a threshold value K then it is considered to be text. If not, it is considered to be a background pixel.

```
For Every pixel Pi
    If GreyValue(Pi) < K Then
        SetAsText(Pi)
    Else
        SetAsBackground(Pi)
    End If
Next pixel
```

2.3.2 Character segmentation

To handle step 2 in figure 2.1, the program scans the image for blobs of data. The size of the blob is then checked. If it is unreasonably small or unreasonably large it is discarded, else it is considered as a potential character. This is a simple method and does not perform very well on all types of backgrounds.

Next task to handle is step 3, character segmentation, in figure 2.1. Blobs of data are found by using a flood-fill function. When a pixel that is considered as a text pixel is found, the flood-fill function is called on that pixel. The flood-fill function calculates the amount of neighbourhood pixels and the smallest bounding rectangle. To find neighbourhood pixels each surrounding pixel is checked. The grey-scale distance of the pixels is calculated. If the distance is below a predefined threshold value it is considered a neighbour.

```
For Each neighbourhood pixel Pi
    If GreyDistance(GreyValue(Pi),GreyValue(Pactive)) < K Then
       AddToBlob(Pi)
```

```
    End If
Next pixel
```

The found data is now passed to the Character class for recognition and classification.

2.3.3 Character recognition

This algorithm is the essence of the program. It is responsible for recognition of the character and making the correct classification. A mixture of algorithms is used but the main algorithm the recognition is built on is the zoning algorithm. The zoning algorithm was chosen for its simplicity and yet high accuracy. This subchapter pertain to step 4 and step 5 in figure 2.1.

The character is subdivided into smaller segments. For each segment the average density is calculated. The density is the number of pixels within that segment divided by the total amount of pixels for the whole character. The character can be seen as an *m* x *n* grid where m and n can be any number greater than 0. Figure 2.5 is an example of a 6 x 6 grid of the character A. To display the different densities for the segments a grey-scale value has been applied to each segment. Darker segment means higher density.

Even though any number of segments may be used in the algorithm it is very important not to choose a too small value. Grids with very few cells become very general and the same densities will be seen for many different characters. The grid should not consist of too many segments either. More segments mean more sensitive to variations.

Many different sizes of the grid were tried during the development of the product. A set of different blueprints were processed by the program for every grid to find the one that produced the best result.

The zoning algorithm is not the only algorithm used to classify the characters. When a new character Z is found the aspect ratio is calculated. The character is then normalized to fit the grid.

```
For Each Template Ti
    If |(ASPECT_RATIO(Ti) – ASPECT_RATIO(Z))| < K Then
       MatchCharacters(Ti,Z)
    End If
Next T

K = constant
ASPECT_RATIO(chr) = function that returns the aspect ratio for
character chr
```

The difference of aspect ratios for Z and each Template T_i must be less than the constant K to be processed further. Major reason for this additional test is to increase speed of the algorithm by reducing the number of templates that needs to be processed.

The accuracy of the best match the algorithm could find must be greater than an *acceptance constant* to be classified. If the accuracy is less than this constant the character gets classified as an unknown character. The best match is still stored in the character class. Reason for this is that even though it has a low accuracy it might be useful to use it for word correction algorithms.

Additional information stored for each classified character is the accuracy it was recognized with. This is not used at the moment but is included if needed in future development.

2.3.4 Word building

This is the last step of the OCR algorithm. The word class is used to keep all characters together. The word class itself does not create the word; the characters are added to it by an extern algorithm. The word class does however support several functions that are specific for the word as a unit.

The program builds words of the characters it has found in the image. This means that character segmentation occurs before word segmentation.

The function that creates words first finds a character that, so far, is not part of any other word and adds this it to a new word. There are many conditions that must be fulfilled for the function to accept a new character to be part of the word. When a new character is found and the following criteria's holds, the character is added to the word:

- The first condition is that the smallest x-coordinate of the bounding rectangle of a new character must be greater than then largest x-coordinate of the bounding rectangle of the previous character.

- The second condition is that both the y-components of the centre of mass coordinates for the characters must be approximately equal. This means that they both are on the same row and therefore may be part of the same word.

- The third condition is that the distance between the characters must be relatively small. The average width of the characters in the word is used to decide whether or not a character is close enough to be part of this word. By using the average width of the characters the algorithm gets size invariant.

- The forth and last condition is that the characters must be of approximately equal sizes.

This is a simple method for finding words and has proved to work very well on all the blueprints it has been tested on.

2.3.5 Word correction

Word correction is used to increase robustness. By using a word correction algorithm the word may contain a few character errors and still be read correctly.

The word correction algorithm consists of two different methods.

- Use of familiar patterns and look for deviance.
- Use of a dictionary

Familiar patterns

The function that looks for familiar patterns is divided into two parts; lowercase/uppercase correction and correction of characters with high similarity.

Correction of incorrect character case is done by inspection of the whole word. Some characters look the same in both uppercase and lowercase, and some look totally different and the algorithm makes use of this. Example of characters that look very different are: a and A, b and B, d and D, e and E, g and G. Characters that look the same are: c and C, o and O, s and S, u and U, x and X, z and Z. The function looks for occurrences of one or more of these characters. For example if the character 'a' is found, its height is then used as a measure of a typical lowercase character height.

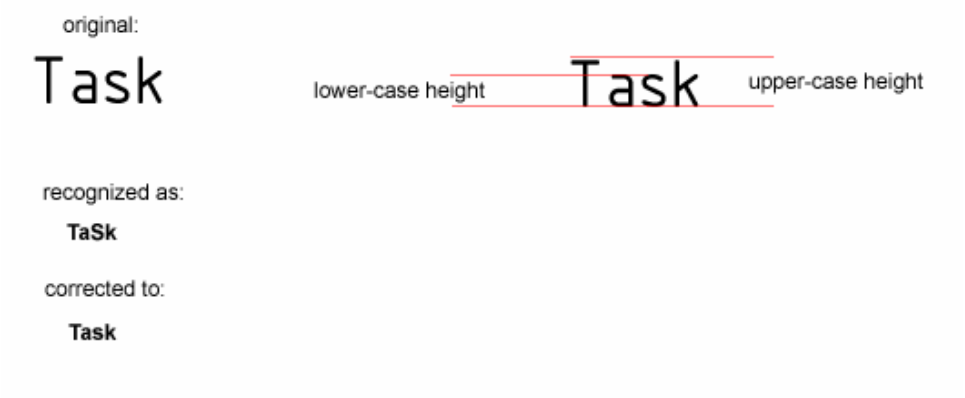


Figure 2.6 Case-height example

Figure 2.6 shows how the character 's' is read as an uppercase 'S' but is corrected in the word correction algorithm. The algorithm used the lowercase 'a' to find a typical lowercase character height and found that the 'S' must be in lowercase.

Some characters look very much alike and therefore got a higher possibility to be incorrectly classified. At the moment the algorithm only focuses on misread '0', 'O' and 'o' characters. If an 'O' is found the nearby characters are inspected. If they are numbers it is most likely that the 'O' in fact should be a '0'. Of course this might not always be the case, but it has proved to work very well on the blueprints that have been tested. It is very uncommon to find characters in between numbers.

Other characters that are often misread (but are not currently being handled by the program) are for example 'S' and '5', 'B' and '8', '1' and 'I'.

Dictionary

A dictionary with commonly used words in blueprints has been created. Every word is matched against the words in the dictionary. A similarity measure is calculated for each word in the dictionary. If this similarity measure is greater than a predefined threshold K (i.e. 90%) the original word gets replaced by the word in the dictionary.

The algorithm does at the moment only calculate similarity measures for words of equal length. This means that if the OCR algorithm happened to include some noise pixels as a character (making the word length longer) or miss a character (making the word length shorter) the word cannot be corrected by dictionary matching.

2.3.6 Category classification

The Optical Character Recognition is a very important feature in the program. Without it the program would not work. The same goes for category classification.

A big problem with category classification is the different names that are used for the same category in the blueprints. A synonym dictionary with all the categories is used to solve this problem. The synonym dictionary contains all the category names that need to be registered. A list of all synonyms to this category name is also stored. The program uses this synonym list to find the different categories that are present in the image. The synonym list is very important when the program is looking for categories.

Another classification problem is that some blueprints do not contain names for the different categories. It is often quite easy for a human to find out which category some text belongs to without a category name specified in the blueprint. On the other hand, it is hard for a computer program to understand what it is reading. Some fields are usually printed in a certain pattern. For example the date is often on the form YYYY-MM-DD where YYYY is the year, MM is the month and DD is the day. The program may look for such patterns and use that for classification. It is however not currently implemented in the program.

To make the program as flexible as possible, two options for category classification are provided; automatic and manual.

Automatic

First all the occurrences the different categories need to be found. The synonym dictionary is used for this.

The program checks all words it got from the OCR algorithm and tries to classify them to one of the found categories. If not appropriate category is found the word is simply discarded.

Some categories may only contain one word; like date and identification number. The algorithm makes sure that one and only one word is added to those categories.

Manual

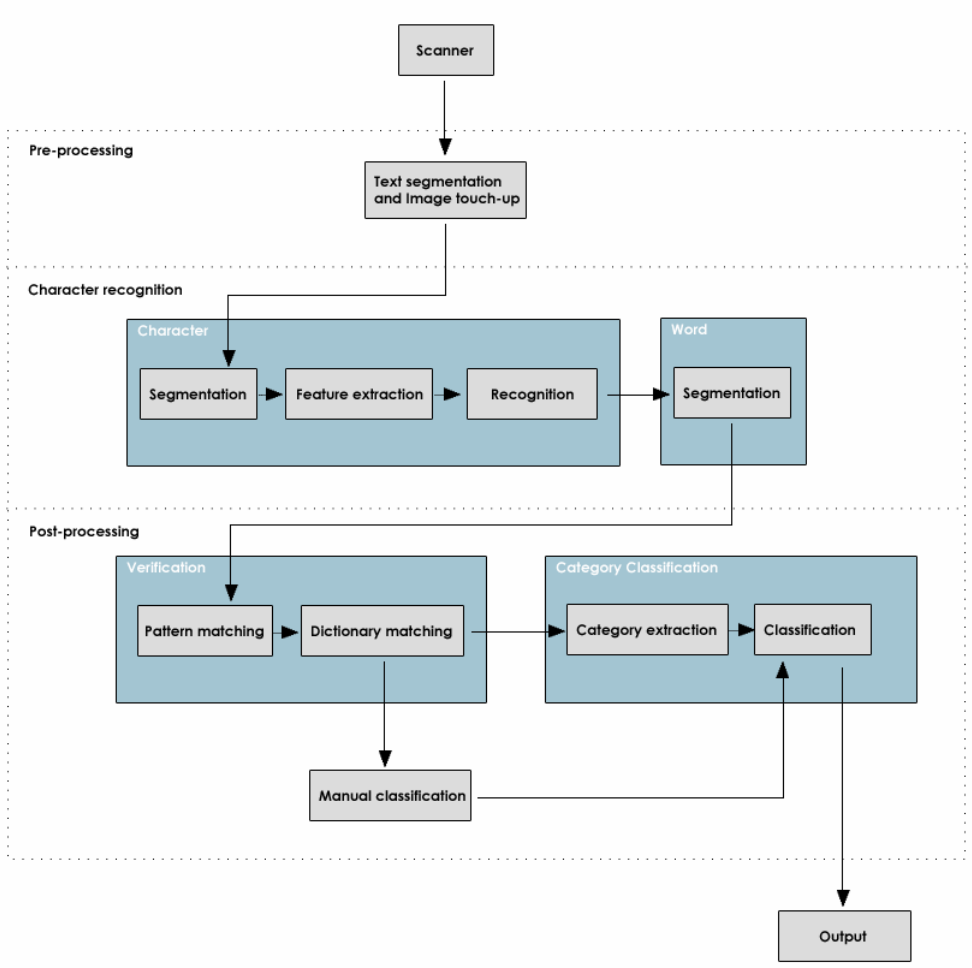
Words can also be manually added to a specified category. The automatic classification can only classify words to categories it has found. This method comes in handy when there are words that the automatic classification has missed.

Automatic and manual classification can both be used on the same image. Words can be manually added to categories where the automatic classification also has added words to. The exception is the one word categories. If a new word is manually added to such a category the old word will be discarded.

3 Approach

This chapter briefly describes the different parts the program is made up of and what each part is for.

3.1 System architecture diagram



3.2 Class overview

The program is divided into many different classes that each has a specific task. A class diagram can be found in figure 3.2.

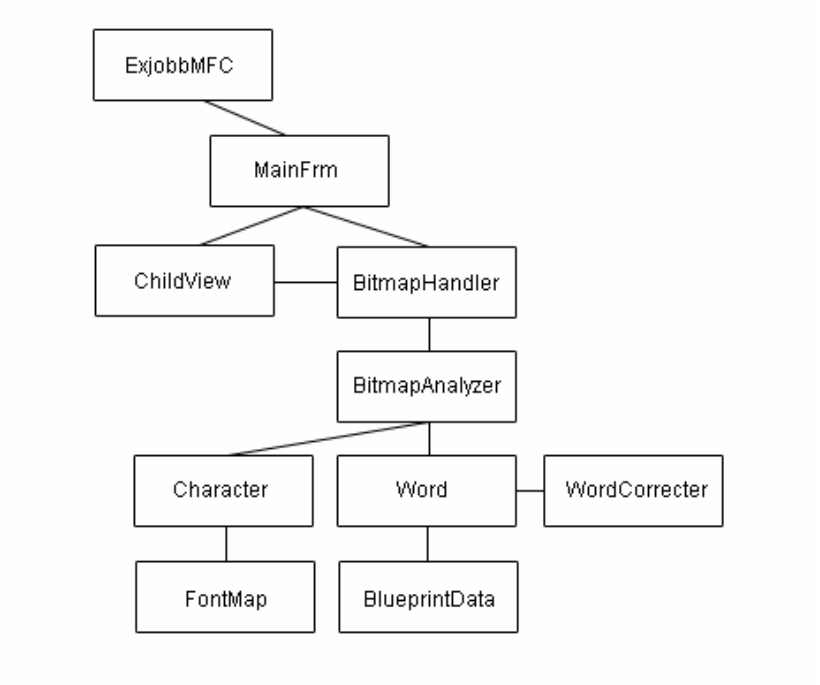


Figure 3.2. Class diagram

Each class is briefly described to get an overview of the system:

- **ExjobbMFC**

This is a very small class that initiates the program and creates a window. It also handles some window messages sent by the main window.

- **MainFrm**

It handles the actual program window. It contains all the buttons, menus and text areas. It also contains the drawing area that is used to display the blueprints. The drawing area is a child window itself and is an instance of the class ChildView.

- **ChildView**

This class handles the drawing area of the program. It is however not this class that performs the actual drawing to it. BitmapHandler contains all information about the opened image and also performs the drawing. ChildView receives messages from the OS when the mouse is clicked and moved and also when the drawing area needs to be refreshed. When an update of the drawing area is requested by the OS, ChildView calls the drawing function in BitmapHandler which in turn renders the image onto the ChildView window.

Zooming in the image is also handled by the ChildView class. Since the drawing area is resizable the image must be resized to fit. ChildView only knows the coordinates of the zooming rectangle on its own drawing area. These coordinates are sent to BitmapHandler that translates them into pixel coordinates in the original image and a correct resize is calculated and the image is redrawn.

- **BitmapHandler**

The BitmapHandler class loads an image and retrieves all information about it, such as height, width, number of total bits, number of bits per pixel, color information etc.

BitmapHandler is used for rendering the image as well as zooming in it. It is also used to return and set pixel colors.

The OCR algorithm assumes the image is a grey scale image. Therefore a desaturation algorithm has been added to its functionality. Even though it is rare, some blueprints have dark background with white text, and the OCR algorithm assumes a bright background with dark text. To resolve this issue, an invert function which calculates the inverted color for each pixel is included in the BitmapHandler class.

- **BitmapAnalyzer**

BitmapAnalyzer's most important function is to find characters and create words. BitmapAnalyzer makes sure the image is in grey-scale (by calling the Desaturate function in BitmapHandler). It also takes care of the text/background segmentation.

- **Character**

This class holds all kinds of information about a character; such as total amount of pixels, density in different regions of the character (number of pixels in the region divided by the total amount of pixels), aspect ratio, height, width and the center of mass coordinate.

After a character has been classified, this class stores the ASCII value of the recognized character. When a character is classified a measure of certainty that the classification is correct is calculated. This value is also stored in the Character class.

- **Word**

The Word class is a set of characters. A word is created from one or more characters. After a word has been closed, the program checks it for errors.

The Word class has got functions to get the center of mass coordinate for the first character, last character and for the whole word which is used for category classification.

- **Wordcorrecter**

WordCorrecter is a class that it used to correct words. To accomplish this it uses a dictionary constructed of words that are commonly used in blueprints. In addition to correcting words it also tries to find characters that have accidentally been classified lower-case when it in fact should be upper-case and vice versa.

A function that finds characters with a high frequency of being misread is also included.

- **Fontmap**

This is the class that performs the actual Optical Character Recognition algorithm. Different font maps can be loaded and used for character recognition matching.

- **BlueprintData**

The purpose of this class is to automatically classify the recognized words into different categories. The user of the program got the opportunity to force a word to a special category if necessary.

3.3 Image viewer

Blueprint images are much larger than the display area of the program. To be able to view the full image it needs to be scaled before it is drawn onto the screen.

The program automatically scales the image to a size that fits the program window. The aspect ratio of the blueprint image needs to be preserved and if it does not equal the aspect ratio of the window, the image will be scaled even smaller to make sure the aspect ratio is not changed. The image is centered on both the vertical and horizontal axis.

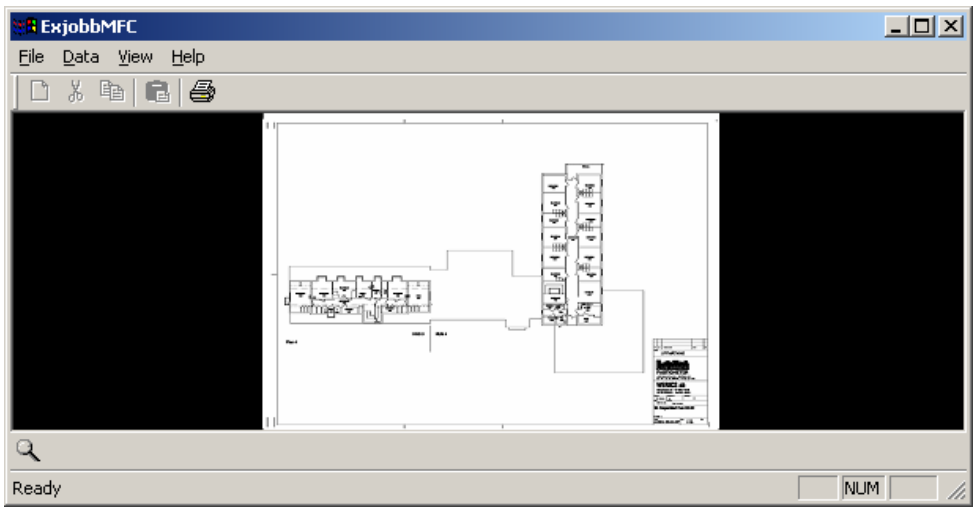


Figure 3.3 Resized image

Figure 3.3 shows how the image is resized, but still preserves the original aspect ratio. It has also been aligned to the center of the window.

3.4 Design

The design is simple and the program is easy to learn. Figure 3.4a shows how the program looks just when it has been started. There are four different drop-

down menus that each handles different tasks. The file menu is used for opening images and exiting the program. The view-menu consists of an option to enable and disable taskbars in the program.

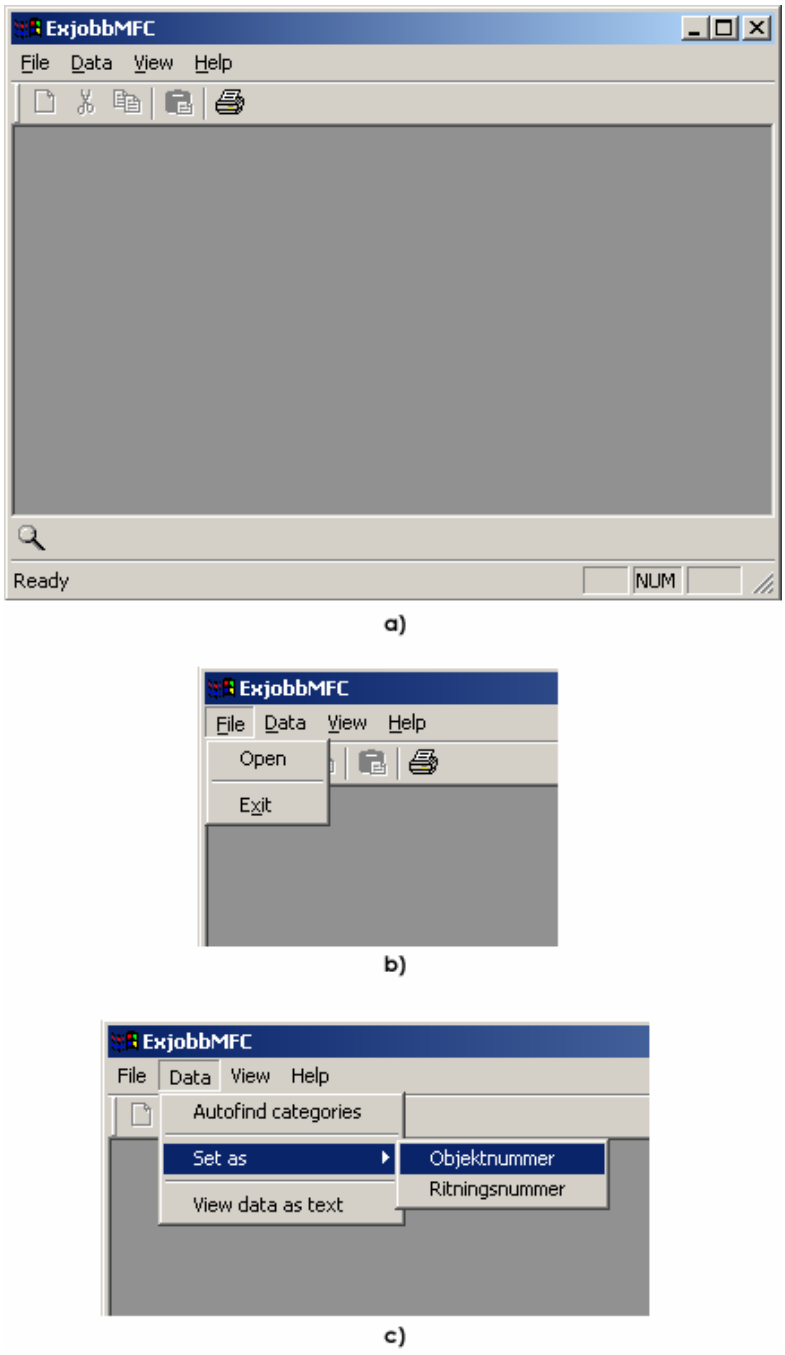


Figure 3.4 a) overview of the program b) The file menu c) The data menu

The most important menu is the data menu: it is used for all the OCR and category classification options. It consists of the following options:

- **Auto-find categories**
  Optical Character Recognition is performed on the image and the program tries to automatically find which category the text belongs to.

- **Set as**
  First a word needs to be selected in the image. By clicking on this menu option another list is displayed. This list contains all the different categories that are being used for classification. The selected word (in the image) can be forced to one of these categories.

- **View data as text**
  Opens a text file that contains the recognized and classified text.

3.5 Usage

This section briefly describes how the program is used.

Zooming

Zooming is performed by using the mouse. The left mouse button is held down as a zooming rectangle is drawn as the mouse is moved. The zooming completes as the left mouse button is released.

Figure 3.5 shows the zooming rectangle. The semi-transparent grey zooming rectangle represents the area that will be zoomed. This zooming rectangle updates automatically as the mouse is moved.

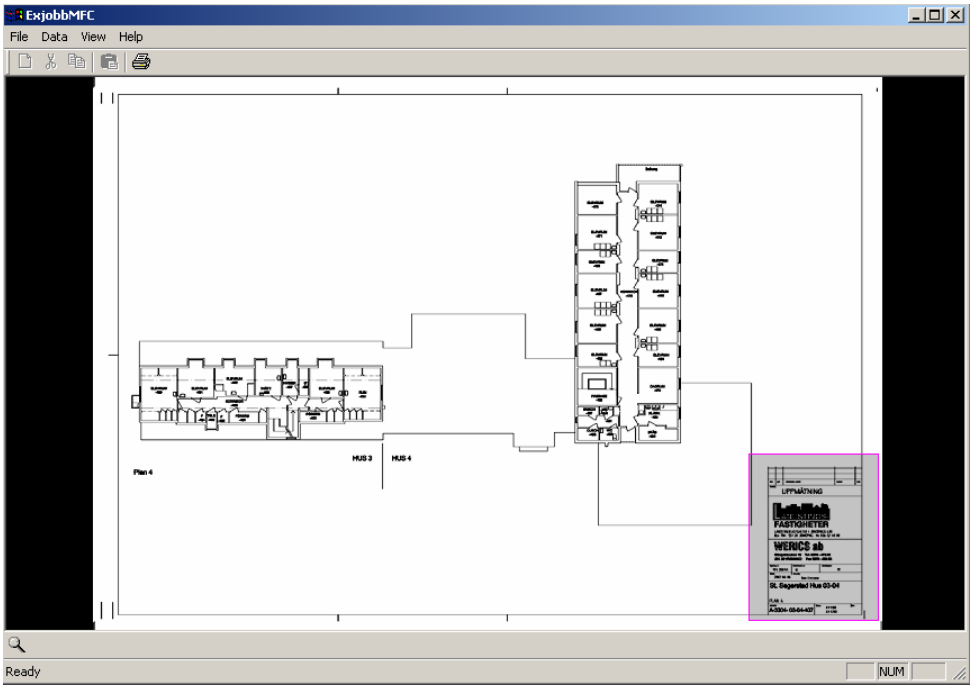


Figure 3.5 Blueprint image

The program does not (in the current version) automatically find where in the blueprint the metadata that needs to be registered can be found. The area where this data is found must therefore be zoomed by the user. The menu option "Autofind categories" searches only the zoomed area for words and categories, which makes it important that the correct area is zoomed.

Classification

Classification may be performed in two different ways.

- The first is to let the program find the categories which the different words belong to. This method is at the moment most of the time not able to find all categories that need to be registered.

- The second is to manually select words in the image and then tell the program which category they belong to. This can alone be used to classify all the categories or be used as a complement to the automatic classification function.

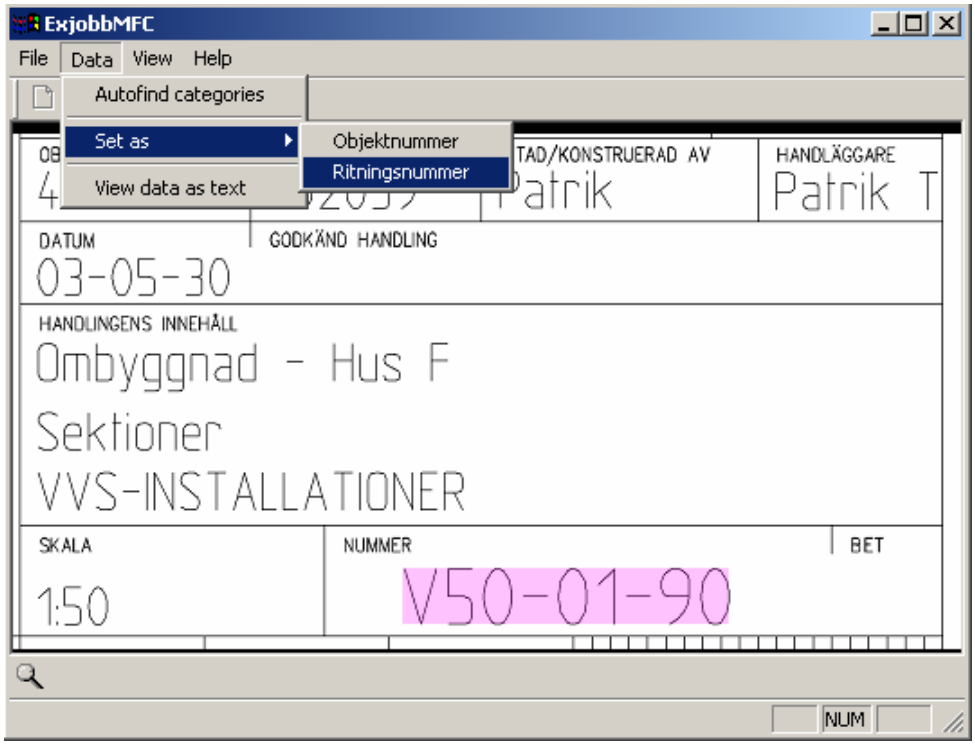


Figure 3.6 A selected word

An example of a selected word is shown in figure 3.6. 'V50-01-90' is the identification number for this blueprint. The menu option 'Set as' is used to classify the selected word.

Words are selected by clicking the mouse button on text in the image. Every OCR recognized word in the image has a smallest bounding rectangle associated with it. These rectangles are used to determine which word (if any) that will be selected. The selected words are highlighted by a semi-transparent purple rectangle around it as shown in figure 3.6.

4 Experimental Results

A few tests have been performed to determine the accuracy of the program in its current state.

4.1 Test description

The focus has been on four different categories: correct words, correct characters, unknown characters and erroneous characters. The difference between unknown and erroneous characters is that unknown characters are not classified at all. They will be displayed as an '?' and the fact that they are unknown may be used in correction algorithms. The erroneous characters are those that have been classified to a different character, and the program is not aware that these characters are faulty.

Blueprints vary a lot in quality and therefore has the text, that is being tested, been categorized into three different quality groups. Statistics for each of the quality settings are computed.

A last note on the parenthesis found in this table. It is the measure of accuracy if all the unknown characters are corrected in a correction algorithm. Even erroneous characters may be corrected, but unknown characters are easier and are therefore included in this table.

4.2 Results

	Good quality	Medium quality	Low quality
Correct characters	97,13%	91,76%	60,93%
Correct words	90%	58% (70%)	8% (22%)
Unknown characters	0%	3,94%	16,49%
Character errors	2,87%	4,30%	22,58%

As seen in the table, the program recognizes the good quality setting very well. It does contain a few errors and this is mainly among characters that look much alike (like S and 5).

Medium quality is still quite a high accuracy in correct characters. There is however very seldom multiple errors within the same word, which means that errors are spread out on many words. This makes the percentage of correct words decrease drastically compared to the good quality. If the unknown characters are corrected the accuracy is increased to 70%.

The low quality setting contains a very large number of errors. Almost 61% of the characters are in fact classified correctly, but only 8% of the words came out correct. There are however a large number of unknown characters (16,5%) and if these are corrected the correct word accuracy is increased to 22%.

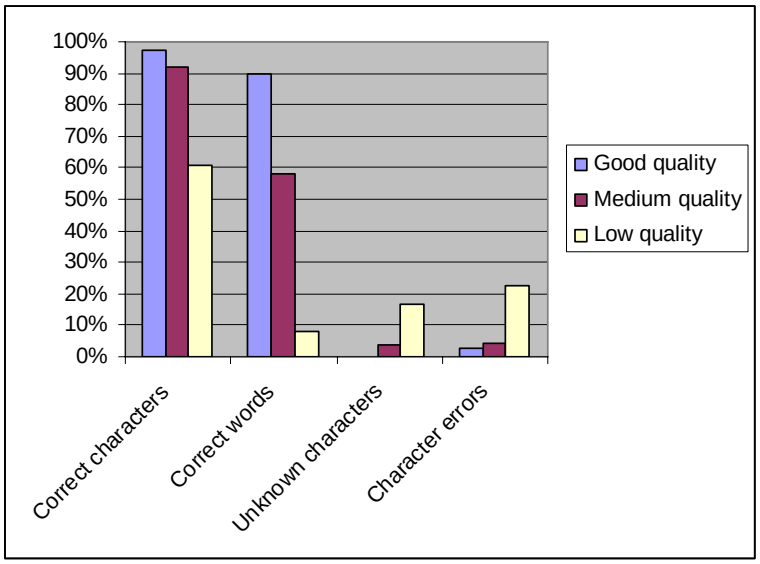


Figure 4.1 Result diagram

5 Conclusions

This chapter contains a summary and conclusions of the paper, the different limitations of the program and what the focus will be on future development of the product.

5.1 Summary

The purpose of this master thesis is to try to develop and evaluate the possibilities of a product that can automatically or semi-automatically register blueprints.

The developed prototype product can read text and to some extent classify it to different categories. Registration of blueprints is extremely important to be error free, especially in the identification numbers. The program does not at the current version provide the needed accuracy, in either Optical Character Recognition or category classification, to be used.

A conclusion that has become clear during the development of this program is that it is most likely too difficult to rely solely on automatic category classification since no errors may be present. However, automatic classification together with manual classification is a good compromise. This means a user always supervise the classification and may correct errors. The supervisor manually adds words, that the automatic classification algorithm cannot classify, by selected them by a single mouse click. This semi-automatic method will be significantly faster than typing every word by hand.

5.2 Limitations

The main limitation of this product is that the accuracy is not high enough for the program to be used at the moment.

5.3 Future plans

The most important feature to improve is the OCR algorithm. Without a solid OCR algorithm the other parts of the program will not be successful either. The current OCR algorithm performs very well in good quality blueprints but the accuracy decreases too much as the quality gets lower. This would not be a problem if it were only to recognize neat machine written text but blueprints are often of bad quality. Especially the recognition of numbers must be very high, since it is impossible for a correction algorithm to find errors in numbers.

Future work on this program will also be to focus on a semi-automatic registration process instead of a fully automatic. The semi-automatic is considered to have the most potential, and is also very flexible.

6 References

[1] Trier, O.D., Jain, A.K., & Taxt, T. Feature Extraction Methods For Character Recognition – a Survey. *Pattern Recognition*, 29 (4), 641-662, 1996.

[2] Hu, M.K. Visual Pattern Recognition By Moment Invariants. *IRE Trans. Information Theory*, IT-8, 179-187, 1962.

[3] Hse, H., & Newton, A.R. Sketched Symbol Recognition Using Zernike Mo*ments.* Technical report, EECS, University of California, 2003.

[4] Teague, M.R. Image Analysis via the General Theory of Moments. *Journal of the Optical Society of America,* 70 (8), 920-930, 1980.

[5] Mori, S., Suen, C.Y., & Yamamoto, K. Historical review of OCR research and development, *Proceedings of the IEEE,* 80, 1029-1058, 1992.

[6] CSIRO Machine Vision. *CSIRO Machine Vision: Expertise: OCR.* [www document]. (2003-10-01). URL http://vision.cmit.csiro.au/expertise/ocr/, visited 2005-03-11.

[7] Tsai, C.M., & Lee, H.J. Binarization of Color Document Images Via Luminance and Saturation Color features. *IEEE Trans. On Image Processing*, 11 (4), 434-451, 2002.

[8] Qixiang, Y., Gao, W., & Huang, Q. Automatic Text Segmentation from Complex Background. *IEEE International Conference on Image Processing (ICIP 2004)*, 2905-2908, 2004.

[9] NeuroSolutions. *Neuro Solutions: What is a Neural Network?* [www document]. URL http://www.nd.com/neurosolutions/products/ns/whatisNN.html, visited 2005-03-11.

[10] Pratt, W.K. *Digital Image Processing*. New York: John Wiley & Sons, second edition, 1991.

[11] Bin, Y., & Jia-Xiong, P. *Invariance analysis of improved Zernike moments*, 2002

[12] Tubbs, J.D. A Note On Binary Template Matching. *Pattern Recognition*, 22 (4), 359-365, 1989