TAMPERE UNIVERSITY OF TECHNOLOGY

TAPIO MANNINEN

COMPUTER VISION AIDED PRINT PATTERN GENERATION IN INKJET PRINTED ELECTRONICS

Master of Science Thesis

Examiners: Heikki Huttunen, D.Sc. (Eng.)
Risto Rönkkä, M.Sc. (Eng.)
Professor Ari Visa
Examiners and topic approved in the Faculty of Computing and Electrical Engineering Council meeting on August 19, 2009

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Automation
MANNINEN, TAPIO: Computer vision aided print pattern generation in inkjet printed electronics
Master of Science Thesis, 69 pages, 9 Appendix pages
December 2009
Major: Signal Processing
Examiners: Heikki Huttunen, D.Sc. (Eng.), Risto Rönkkä, M.Sc. (Eng.), Professor Ari Visa
Keywords: inkjet printed electronics, computer vision, dynamic print pattern correction

Inkjet printed electronics is one of the new promising electronics manufacturing techniques out there. It has become a widely adopted manufacturing method especially in the field of low-cost electronics.

This thesis considers an application of inkjet printed electronics where conductive ink is used for printing the connections between the components on a single unit called a module. The base module is fabricated by molding the components together such that the connection points of the components form a level surface. After this, the wiring is printed on top.

Because of the inaccuracies in the fabrication process, there is often a mismatch between the designed print pattern and the target module. The purpose of this thesis is to introduce an online print pattern generation system that uses computer vision to detect the locations of the module components and then modifies the print pattern accordingly.

By integrating the print pattern generation system as a part of the manufacturing process, not only is it possible to print functioning modules but also multiple modules can be printed at the same time. This way the capabilities of inkjet printed electronics can be more efficiently harnessed.

The experiments prove that the developed print pattern correction system together with the proposed imaging setup are able to produce desired results in practice. In addition to successfully printing ten modules at once, it is also shown that the developed system is robust and generalizes well for different types of modules.

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Automaatiotekniikan koulutusohjelma
MANNINEN, TAPIO: Computer vision aided print pattern generation in inkjet printed electronics
Diplomityö, 69 sivua, 9 liitesivua
Joulukuu 2009
Pääaine: Signaalinkäsittely
Tarkastajat: TkT Heikki Huttunen, DI Risto Rönkkä, Professori Ari Visa
Avainsanat: tulostettava elektroniikka, konenäkö, dynaaminen tulostuskuvion korjaus

Tulostettava elektroniikka on yksi lupaavimmista menetelmistä nykyajan elektroniikkatuotannossa. Tulostettavaa elektroniikkaa käytetään laajalti etenkin edullisen elektroniikan tuottamisessa.

Tämä opinnäytetyö käsittelee tulostettavan elektroniikan sovellusaluetta, jossa elektroniikkamodulin komponenttien väliset kytkennät luodaan sähköä johtavaa mustetta tulostamalla. Komponenttien muodostama yksikkö valmistetaan valamalla komponentit yksittäiseksi moduliksi, jossa komponenttien kytkentäpisteet sijaitsevat samassa tasossa. Tämän jälkeen modulin päälle tulostetaan johtimet.

Modulien valmistustavasta johtuen tulostettavan johdinkuvion ja komponenttien kytkentäpisteiden välille syntyy usein virhettä. Tämän opinnäytetyön tarkoituksena on ratkaista edellämainittu ongelma käyttämällä automaattista tulostuskuvion korjausmenetelmää. Aluksi komponenttien kytkentäpisteiden sijainnit selvitetään konenäköä apuna käyttäen, minkä jälkeen tulostuskuvio muokataan vastaamaan todellista tilannetta.

Tulostuskuvion korjausjärjestelmän integrointi osaksi tuotantoprosessia mahdollistaa virheettömien modulien tulostamisen. Tämän lisäksi on mahdollista tulostaa useita moduleja samanaikaisesti, jolloin tulostettavan elektroniikan vahvuudet tuotantomenetelmänä tulevat tehokkaammin hyödynnettyä.

Työssä tehtävät käytännön kokeet osoittavat, että tulostuskuvion korjaamisen sekä esitetyn kaltaisen kuvantamisjärjestelyn avulla voidaan saavuttaa tavoitteiden mukaisia tuloksia. Sen lisäksi, että on samanaikaisesti mahdollista tulostaa kymmenen modulia, kokeet osoittavat, että kehitetty järjestelmä on robusti ja yleistyy hyvin eri tyyppisille moduleille.

PREFACE

This thesis is written based on the work that was mainly conducted between March, 2007 and December, 2008 while working as a research assistant at the Department of Signal Processing at Tampere University of Technology (TUT). The goal of the project funded by Nokia Research Center (NRC) and Tekes FinNano program was to improve manufacturing methods in inkjet printed electronics. The work was a success resulting in one US patent, several inventions, multiple publications, and two master's thesis including the one you are currently reading. The acknowledgments go to the head of the department, Professor Ari Visa, who awarded the scholarship for writing this thesis. I would also like to thank the Department of Electronics, especially Ville Pekkanen, M.Sc. (Eng.), for providing the equipment, facilities, and staff for running the experiments.

My sincerest thanks go to the project team I had during the NRC project. These people include Heikki Huttunen, D.Sc. (Eng.), Risto Rönkkä, M.Sc. (Eng.), Kalle Rutanen, M.Sc. (Eng.), and Pekka Ruusuvuori, M.Sc. (Eng.). Without the technical expertise of Huttunen and the out-of-the-box thinking of Rönkkä (the masterminds) the project would not have been the success it was. Having Rutanen working in the same office saved me a lot of time and effort during the time we were implementing the print pattern correction system. It turned out that no matter what the question or problem was, Rutanen always seemed to have an answer or algorithm to solve it. I would also like to thank Hannu Oinonen, M.Sc. (Eng.), whose ini-file system is still in use and has spread into other projects as well.

Finally, I would like to thank my spouse along with our four-legged friends. Their tolerance during the programming intensive nights that I had before each printing day was admirable. And when the time came to relax for a while, it took them less than a blink of an eye to turn my thoughts away from work.

Tampere, November 29, 2009

Tapio Manninen
email: tapio.manninen@tut.fi
tel: +358509184455

CONTENTS

ABBREVIATIONS AND NOTATION

$\mathbf{1}$	n-dimensional vector of ones
a	Slope parameter of the logistic function
$\mathbf{a}$	n-dimensional example point or vector
A	Example set of n-dimensional points
$\mathbf{A}$	Point matrix corresponding to set A
b	Bias of a neuron
B	Binary image
d_i	Desired output of the i^{th} output neuron
$\mathbf{d}_i$	Desired output of NN corresponding the data sample $\mathbf{x}_i$
$\mathbf{D}$	$n \times n$ diagonal singular value matrix
δ	Local gradient of a neuron
δ_c	Chernoff upper bound
η	Learning-rate parameter of the back-propagation algorithm
ε	Confidence level of the Chernoff bound
$\varphi(\cdot)$	Activation function of a neuron
$\mathbf{I}_n$	$n \times n$ identity matrix
I	Gray-level intensity image
$J(\cdot)$	Cost function
$L(\cdot)$	Logical threshold function
$\boldsymbol{\mu}$	Weighted centroid of an n-dimensional point set
μ_w	Average intensity of the image window W
$O(\cdot)$	Big O notation for determining upper bounds for algorithm complexities
p	Stroke width
p_c	Classification performance
$\widehat{p_c}$	Maximum likelihood estimate of p_c
P, Q	Sets of n-dimensional points
$\mathbf{P}, \mathbf{Q}$	Point matrices corresponding the point sets P and Q, respectively
$\mathbf{R}$	Orthogonal rotation matrix of size $n \times n$
$\mathbb{R}^+$	Set of positive real numbers
s	Real valued positive scaling term
$\mathbf{S}$	Conditionally modified identity matrix of size $n \times n$
t	Threshold value
$\mathbf{t}$	n-dimensional translation vector
T	Gray-level template image
$\mathbf{U}, \mathbf{V}$	$n \times n$ orthogonal matrices
v	Activation potential of a neuron
w_{ij}	Weight of the synapse from neuron i to neuron j

$\mathbf{w}$	Weight vector
Δw_{ij}	Update of the weight w_{ij} determined by the delta rule
W	Local image window
$\mathbf{W}$	Diagonal matrix with vector $\sqrt{\mathbf{w}}$ as its diagonal
x_i	The i^{th} input of a neuron
$\mathbf{x}_i$	A NN training data sample
y	Output of a neuron
IC	Integrated circuit
MCM	Multi chip module
NN	Neural network
NRC	Nokia Research Center
PC	Personal computer
PPM	Point pattern matching
RGB	Red, green, blue (color space)
SDK	Software development kit
SIFT	Scale invariant feature transform
SVD	Singular value decomposition
TUT	Tampere University of Technology

1. INTRODUCTION

In 1965, Gordon E. Moore published his statement about the exponential growth of the transistor count on an integrated circuit (IC) [1]. Nowadays, this statement is known as the world-famous *Moore's law* and it is applied on almost every field involving some form of the exponential rate of change. In the electronics manufacturing industry, Moore's law has become a yardstick in power consumption, component size, computation performance per unit cost, etc.

Researchers from Intel forecasted in 2003 that somewhere around the year 2020 the conventional electronics manufacturing methods would meet their wall in the sense of the Moore's law [2]. Also Moore himself said in 2005 that the law cannot be sustained forever [3]. On the other hand, most radical futurists like Raymond Kurzweil believe that alternative technologies will arise and ultimately lead to the *technological singularity* [4], i.e., to the point where artificial intelligence surpasses the human intelligence recursively developing until collapsing into something called the *superintelligence*.

Whatever the future will bring us, it is obvious that new technologies are constantly developed in order to overcome the limitations of the conventional electronics. One of the most promising electronics manufacturing methods developed to maintain the advancement of technology is *printed electronics*. Printed electronics is a common name for methods used for creating electronic devices by means of printing. These methods include, e.g., screen printing, rotogravure, offset printing, nanoimprint lithography, and inkjet. Many of these are familiar from the graphics printing industry. The technological contributions of printed electronics reside more on cost efficient manufacturing than on a pursuit of high performance (although in nanoimprint lithography it is possible to fabricate high performance transistor structures by not restricting ourselves into the conventional two dimensions but by using three dimensional designs).

Like implied, printed electronics is often described as low-cost and low-end method when compared to the conventional high-cost and high-end electronics. Applications of printed electronics include e-paper, OLED (organic light-emitting diode) displays, RFID (radio-frequency identification) tags, and batteries. One of the e-paper applications already adopted by vast masses is the e-book reader *Amazon Kindle*.

Printed electronics often offers cheaper and more flexible, scalable, and environmentally friendly way to produce already existing technology. It also enables completely new solutions, e.g., via the diverse choice of materials. The markets for printed electronics are

expanding in an increasing rate. However, investments on research and development are needed in order for printed electronics not to fall behind other technologies.

In this thesis we will concentrate on electronics manufacturing on a certain field of *inkjet printed electronics*. The application is described more thoroughly in section 1.1. The purpose of this thesis is to introduce a dynamical, online, print pattern generation system. It has been proven out in practice that this kind of system is necessary for the mass production use of the inkjet application we are focusing on. The reasons are discussed in the problem statement in section 1.2.

The outline of the rest of the thesis is the following: Chapter 2 defines the theoretical and mathematical preliminaries that are needed to understand how the introduced print pattern generation system works. The concept as well as the details of the system are described in chapter 3. Chapter 4 conducts the validation of the print pattern generation system, and the results are further on analyzed in chapter 5. Finally, chapter 6 concludes the thesis giving some ideas for future work.

1.1 Inkjet printed electronics

In inkjet printed electronics, the idea is to use an inkjet printer in order to create electromagnetic, optical, and/or electromechanical structures that eventually form a working electronic device. This is done by printing liquid materials (inks, that is) with various properties like conductivity or dielectricity.

Inkjet printing is environmentally friendly process as it is *additive*, i.e., material is only added where it is needed unlike in the etching process of conventional electronics manufacturing. Savings in material consumption along with the low initial costs make inkjet printing a flexible manufacturing method. Inkjet printing is also highly scalable to different circuit sizes.

At this point the practical applications of inkjet printed electronics focus on printing of simple structures like wiring and antennas. Other applications in the near future could include, e.g., manufacturing of efficient and cheap solar cells for energy production. As the technology is developing, also more complex structures become possible.

In this thesis we will concentrate on a particular field of inkjet printed electronics. In our application, surface mounted passive components and ICs are first molded in a background material such that the material together with the contact areas of the components form a level surface like seen in Figure 1.1. The complete unit with the components molded together is called a *module*. After the background material has been hardened, an inkjet printer is used for printing the wiring that creates the connections between the components. After an arbitrary amount of subsequent printing with conductive and dielectric inks the result is a functioning electronic device.

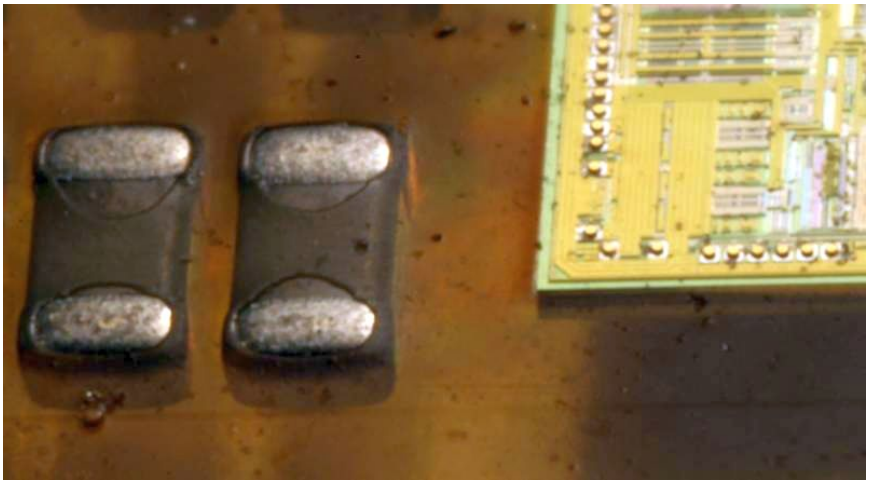


Figure 1.1: Two passive components and an IC molded in the background material.

1.2 Problem statement

Like in any new manufacturing method, there are problems that need to be solved before the application becomes practical enough for actual production use. In this thesis, the problem of *connection point misalignment* is solved. This problem is caused by errors between the locations of the designed components and the locations of the components in practice. Connection point misalignments occur due to things like thermal expansion, deformations in the module background material (component drifting), and inaccuracies in the component placement. This causes the subsequent wiring printed on top of the module to become misaligned with the component connection points. As each component is uniquely displaced, the problem can not be solved by simply realigning the print pattern.

The purpose of this thesis is to introduce a computer vision aided print pattern generation system that detects the errors in the locations of the component connection points. After detecting the errors, the system modifies the designed wiring scheme such that it matches with the actual component layout. By printing the modified design instead of the original one, the resulting module has no missing or false contacts. Using the print pattern generation system also allows the printing of multiple modules at once. Conventionally, it has been possible to print only one module at a time.

Figure 1.2 visualizes the problem of connection point misalignment as well as our solution to the problem. In the ideal situation (Figure 1.2a), there is no error in the locations of the components before the designed wiring is printed on the module. The printer prints according to the original design data. After the printing, all wire ends meet their targets. In a real situation (Figure 1.2b), the components are not in their assumed locations. When the printing is done with unmodified design data, the wire ends do not meet their targets and the module becomes useless. The proposed solution (Figure 1.2c) adds an extra processing block prior to the printing. In this block, the locations of the components are detected and the design data is modified accordingly. The printing result is flawless.

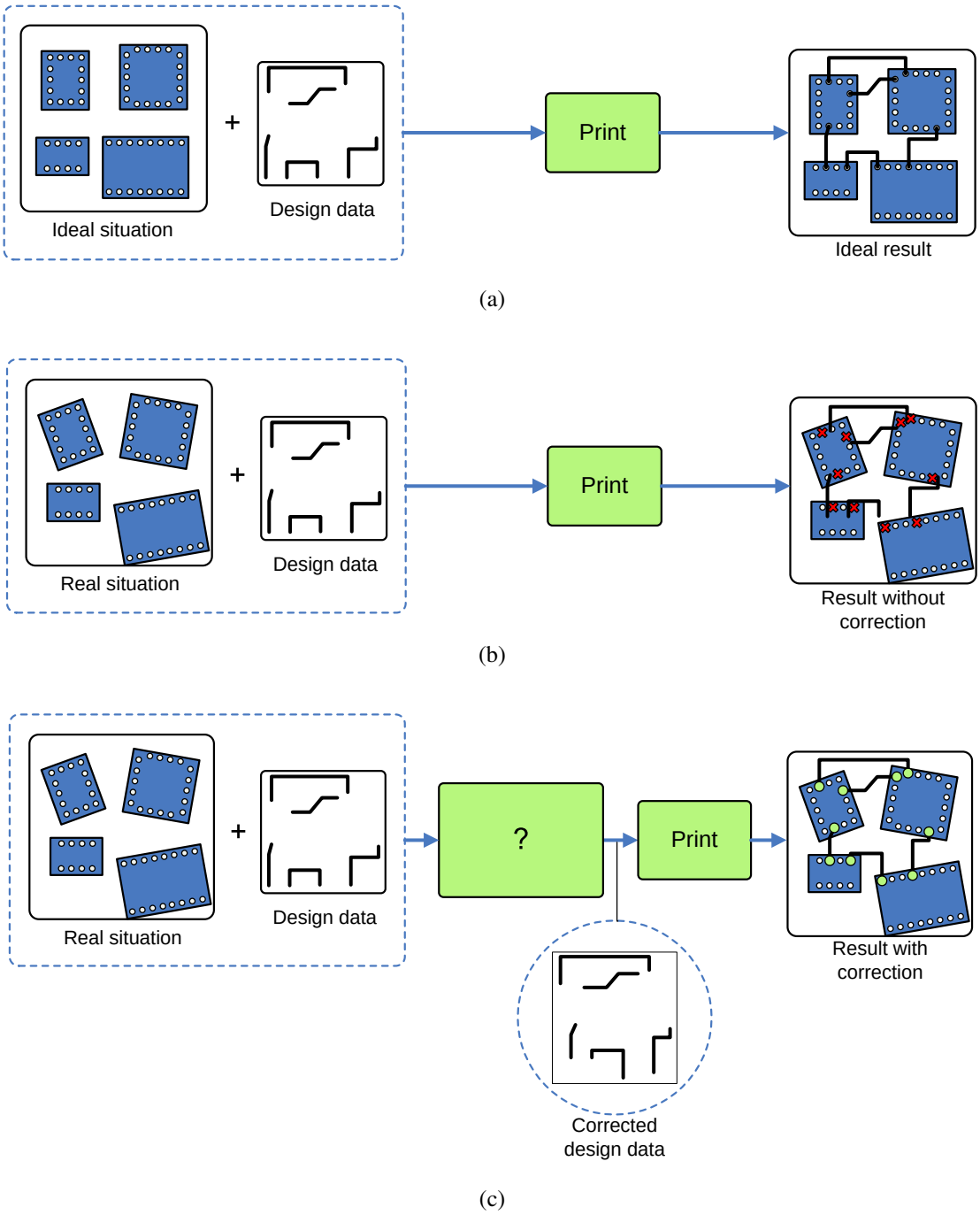


Figure 1.2: Diagrams of the ideal printing situation (a), the real life printing situation (b), and the real life printing situation where there is an extra processing block to solve the problems due to connection point misalignments (c).

The introduced correction system is the first of its kind. Parts of the system have been published in our previous publications [5; 6; 7]. A high-level technique to manipulate design wiring bitmaps through vectorization has been developed in the master's thesis by Rutanen [8]. In this thesis, the pieces of the correction system are put together and explained in more detail. Also validation of the system is conducted in practice.

2. THEORETICAL BACKGROUND

This chapter introduces the main concepts and methods that are put into use in the print pattern generation system. During the development of the system also a vast number of other techniques were used but then discarded as better alternatives were discovered. In order to keep the scope of the thesis in leash, only those techniques that the final version of the system involves are described here.

The structure and motivation of the theory chapter is the following: Section 2.1 introduces the concept of template matching. Template matching is used, e.g., in the detection of the ICs in the module image. Section 2.2 deals with point pattern matching, which is needed when associating the module design data with the detected features in the module image. Sections 2.3 and 2.4 describe logical level thresholding and neural networks, respectively. These techniques are needed in the detection of the IC connection points.

Although the content of the theory chapter is based on practical needs, the approach tries to be rather general. This is to prevent overlap with the content of chapter 3, whose role is to put the theory into practice and describe the applications in detail.

2.1 Template matching

Image registration and any kind of detection of objects in images are often encountered problems in the applications of computer vision. A summary of different registration methods is available, e.g., by Zitová and Flusser [9]. Probably the most popular modern method is *Scale Invariant Feature Transform* [10] or commonly known as SIFT. The method is robust, efficient, and can handle different kinds of transformations in 2-d and 3-d. Classical and simpler methods include, e.g., correlation based methods like *template matching*, which is introduced right after establishing the mathematical notation used for images in this thesis.

Definition 1. *An image I having M rows and N columns is defined as a mapping $I : A \mapsto B$, where $A = \{(x,y) \mid x \in \{0,1,\ldots,N-1\},\ y \in \{0,1,\ldots,M-1\}\}$. Range B can be $\{0,1\}$ or some subset of $\mathbb{R}$ or $\mathbb{R}^3$, depending whether the image is binary, gray-level, or color image, respectively.*

The convention of Definition 1 is used throughout this thesis.

Template matching is an image registration method where there are two gray-level images: the actual image I, or the *scene image*, and a *template image* T, which is usually

much smaller in size than the scene image. The idea is to find coordinates (x,y) that give the best fit between the template image T and the scene image I. [see 11, chap. 4.6.4]

One way to find an optimal fit is to find the point (x,y) that maximizes the cross-correlation of I and T, i.e.,

$$\sum_{(x_t,y_t)\in D} [I(x+x_t,y+y_t)T(x_t,y_t)], \tag{2.1}$$

where D is the domain of the template T. Finding the maximum correlation is easy to implement and efficient to compute (like convolution). However, using correlation has also some disadvantages, especially from the image processing point of view. Clearly, as the pixel values of I get larger, also the correlation increases. This can easily result in the maximum correlation being in bright areas of I. Also the different lighting conditions between the template and the image could mess up things. These problems can be solved – at least to some extent – by applying normalization to the images. This is usually done by subtracting their means from the images and then dividing the images by their standard deviations.

Another approach in addition to the maximum correlation is to find (x,y) that minimizes the squared difference between I and T, i.e.,

$$\sum_{(x_t,y_t)\in D} [I(x+x_t,y+y_t)-T(x_t,y_t)]^2, \tag{2.2}$$

or, equivalently,

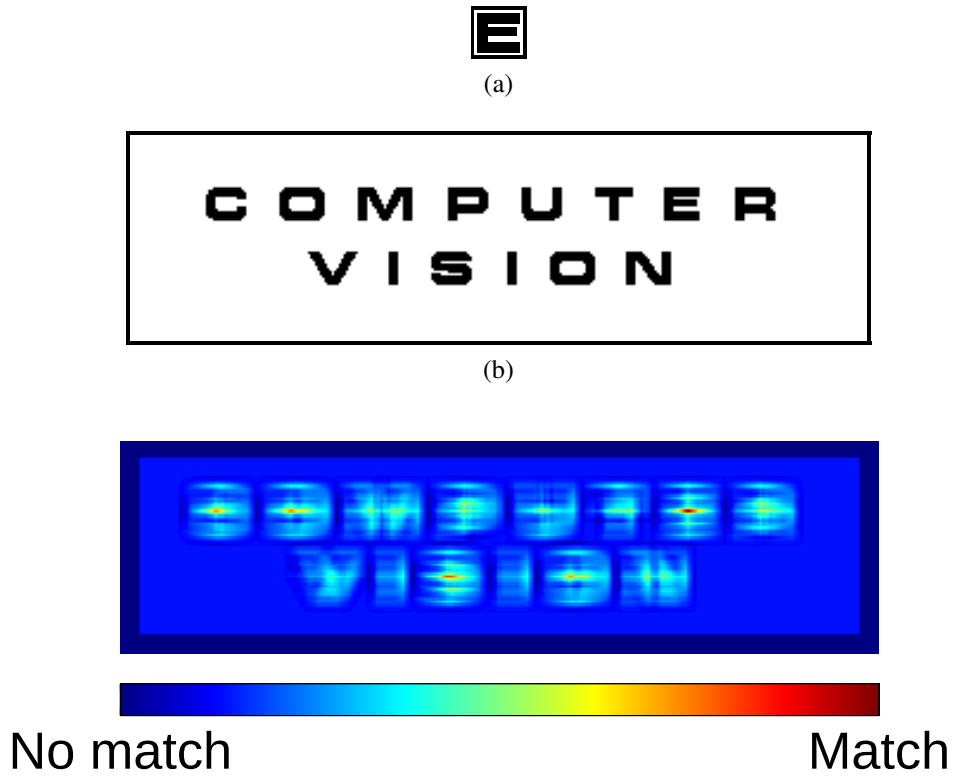
$$\sum_{(x_t,y_t)\in D} I^2(x+x_t,y+y_t) - 2\sum_{(x_t,y_t)\in D} [I(x+x_t,y+y_t)T(x_t,y_t)] + \sum_{(x_t,y_t)\in D} T^2(x_t,y_t). \tag{2.3}$$

As the last term in Formula 2.3 doesn't depend on (x,y), the problem of minimizing the square error can be rewritten as maximizing

$$2\sum_{(x_t,y_t)\in D} [I(x+x_t,y+y_t)T(x_t,y_t)] - \sum_{(x_t,y_t)\in D} I^2(x+x_t,y+y_t). \tag{2.4}$$

The first term in Formula 2.4 is two times the correlation presented in Formula 2.1. The second term prevents the criterion to grow in bright image areas, which was one of the problems when using simple correlation. As convolution can be used in the implementation of Formula 2.4, computing becomes more efficient with respect to using Formula 2.2.

There is a toy example of template matching in Figure 2.1. The template in Figure 2.1a showing a letter 'E' is matched against the scene image in Figure 2.1b. It can be seen in the result image in Figure 2.1c that the maximum match is right on the center of the letter 'E'. Notice that letters resembling 'E', like 'S', also produce a relatively good match.



(a)

(b)

(c)

Figure 2.1: A template matching example. The template (a) is matched against the scene image (b). The color of the result image (c) varies according to the square difference between the template and the scene. Red indicates zero error while blue is for maximal error. Maximal error is assigned where the template would exceed the scene image borders.

Template matching is efficient and straightforward method for finding the template from the scene image. The method can be extended to work on color images, e.g., by performing the matching separately for all the color channels, adding up the criterion values, and then finding the optimal value. However, template matching can only handle image translations. If the template is rotated or is in a different scale than the scene image, more advanced methods like SIFT or point pattern matching (see section 2.2) has to be used.

2.2 Point pattern matching

Point pattern matching (PPM), like template matching in the previous section, is a method for finding a correspondence between two objects. In PPM, the general idea is to find a transformation between two n-dimensional point sets

$$P = \{\mathbf{p}_k \in \mathbb{R}^n \mid k = 1, \ldots, m_p\} \tag{2.5}$$

and

$$Q = \{\mathbf{q}_k \in \mathbb{R}^n \mid k = 1, \ldots, m_q\}. \tag{2.6}$$

where m_p and m_q are the number of points in sets P and Q, respectively. In the applications of computer vision, the dimension n is usually two or three. In two dimensional cases, where the matched points represent control points of images, PPM can be used in a similar manner to template matching. However, PPM can handle more general transformations than simple translation.

Like in template matching, the point set to be transformed is often considered as the *pattern* which is matched to a larger point set, i.e., the *scene*. In some applications the sets are about the same size and it makes no difference whether we match the first set to the other one or the other way around.

In this thesis we will only concentrate on PPM that produces one global *similarity* or *rigid* transformation between the two point sets. Similarity and rigid transformations are both linear transformations. Point $\mathbf{y} \in \mathbb{R}^n$ that is a similarity transformed version of the point $\mathbf{x} \in \mathbb{R}^n$ can be described by the equation

$$\mathbf{y} = s\mathbf{R}\mathbf{x} + \mathbf{t}, \tag{2.7}$$

where $s \in \mathbb{R}^+$ is a scaling factor, $\mathbf{R} \in \mathbb{R}^{n \times n}$ is an orthogonal rotation matrix, and $\mathbf{t} \in \mathbb{R}^n$ is a translation vector.

As can be seen from Equation 2.7, similarity transformation consists of *scaling*, *rotation*, and *translation*. Rigid transformation, in turn, is a special case of similarity transformation with scale parameter equal to one. Both similarity and rigid transformations are special cases of an *affine* transformation, respectively. Affine transformation consists of scaling, rotation, translation, and *shearing*. There is an example body transformed with different kinds of transformations in Figure 2.2.

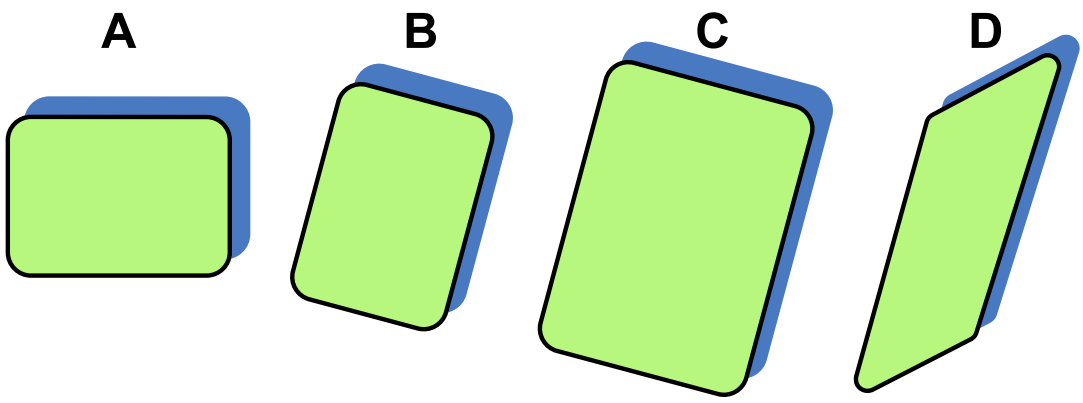


Figure 2.2: Different types of transformations in 2-d. Block A is the original one. Blocks B, C, and D are results of a rigid, similarity, and affine transformations, respectively.

Similarity transformation preserves the shape of the transformed object. This makes it ideal especially for computer vision related applications, which often involve, e.g., registration of physical bodies whose shape remain unchangeable. Rigid transformation comes handy in cases where we have special knowledge about the scale. If we, e.g., know that the scale of our image registration template is the same as the scale of the image that we are matching the template for, using rigid transformation gives probably better results than using similarity transformation.

What makes finding the right transformation in PPM a non-trivial problem in practice are things like noise in the point locations, unknown correspondence between the points in both of the sets, and possible missing or extra points in either one of the sets. Finding a solution to the PPM problem can be divided into two parts. These parts are considered separately in the following two sections. Section 2.2.1 considers finding the correspondence between the points of the two point sets while in section 2.2.2 we determine the actual parameters of the transformation that transforms the set into the other set.

2.2.1 Coupling points

The hardest part in the whole PPM problem is to figure out, which points in the first set correspond to the points in the second set. At this point, some kind of algorithm is obviously needed.

A good summary of PPM algorithms based on different kinds of methods is available by Li et al. [12]. These methods include, e.g., relaxation [13], graphs [14; 15; 16; 17], iterative approaches [18], and clustering [19; 20; 21]. The choice of the PPM algorithm for determining the point correspondence depends solely on the application at hand. Things that can influence the selection are, e.g., cardinality and dimension of the point sets, type of transformation to use, possibility of outlier points, and the amount of noise in the point locations.

A PPM algorithm that suits the needs of this thesis is the one by Chang et al. [21]. Chang's method can find correspondence between similarity transformed point sets in 2-d. Although not the most efficient one, Chang's method is robust and works also when the point sets have both missing and extra points. Because the Chang's algorithm has one major drawback, an improved version of it is introduced next.

Consider the point sets P and Q given in equations 2.5 and 2.6, respectively. Let $n = 2$ and $m_p \leq m_q$, i.e., the point sets are two-dimensional and there are less or equal amount of points in set P when compared to set Q. Chang's PPM method tries to match the point set P with the point set Q. The algorithm is based on clustering of the scale-rotation pairs, which are achieved by calculating scales and rotations between each of the point pairs $(\mathbf{p}_i, \mathbf{p}_j)$ in set P and $(\mathbf{q}_u, \mathbf{q}_v)$ in set Q. This produces a total of $mn(m-1)(n-1)/4$ scale-rotation pairs. The algorithm then finds an area with large amount of scale-rotation pairs densely packed. The average scale and rotation of the detected cluster is approximately

the scale and rotation of the transformation between the two sets. After this the translation term can be calculated like will be shown in Equation 2.28 on page 17.

After determining the parameters of a similarity transformation that transforms set P approximately into set Q, the actual point correspondence can be determined by transforming P with the approximate transformation and pairing each point to the closest point in Q. If no pair is found within a given radius, the point is considered an outlier and discarded.

Chang uses a fixed size accumulator array to collect each scale-rotation pair and then finds the element of the accumulator array with the most hits. This is equal to finding a fixed size rectangle that includes most scale-rotation pairs in a discrete scale-rotation space. This creates a behavior that treats the scale changes on the interval $[1.0, 1.1]$ equally to the scale changes on the interval $[1000.0, 1000.1]$ as these intervals are of the same length. However, it is obvious that the first interval is relatively far more significant than the latter one. In addition, the borders of the accumulator array bins may split the cluster in half. Using a discrete accumulator array causes also memory issues as the array may have to be very large in order to attain adequate resolution. Also the scale axis, which in theory is the interval $(0, \infty)$, has to be truncated at some point.

An improved version of the Chang's accumulator array is now introduced to overcome the problems related to using a discrete accumulator array. The basic idea is for each scale-rotation pair to individually form a rectangle defining the error bounds for that particular scale-rotation pair. The target cluster is then considered as the area where most rectangles intersect each other in the *continuous* scale-rotation space. Dimensions for each rectangle are derived from the given error bounds of each point in the following manner.

Consider points $\mathbf{p}_i, \mathbf{p}_j \in P$ and $\mathbf{q}_u, \mathbf{q}_v \in Q$ such that $i \neq j$ and $u \neq v$. The problem is to find a scaling parameter $s \in \mathbb{R}^+$ and a rotation parameter $\theta \in [0, 2\pi)$ that transform the vector $\mathbf{p}_j - \mathbf{p}_i$ such that it equals the vector $\mathbf{q}_v - \mathbf{q}_u$. Assume that, when observing the points $\mathbf{p}_i$, $\mathbf{p}_j$, $\mathbf{q}_u$, and $\mathbf{q}_v$, instead of getting their true locations, we only get the noisy locations

$$\mathbf{p}'_i = \mathbf{p}_i + \mathbf{n}_i, \tag{2.8}$$

$$\mathbf{p}'_j = \mathbf{p}_j + \mathbf{n}_j, \tag{2.9}$$

$$\mathbf{q}'_u = \mathbf{q}_u + \mathbf{n}_u, \tag{2.10}$$

and

$$\mathbf{q}'_v = \mathbf{q}_v + \mathbf{n}_v, \tag{2.11}$$

where $\mathbf{n}_i$, $\mathbf{n}_j$, $\mathbf{n}_u$, and $\mathbf{n}_v$ are random noise terms with symmetric distributions. These points are called the measured points.

Given a certain confidence level (that depends on the distribution of the noise), the

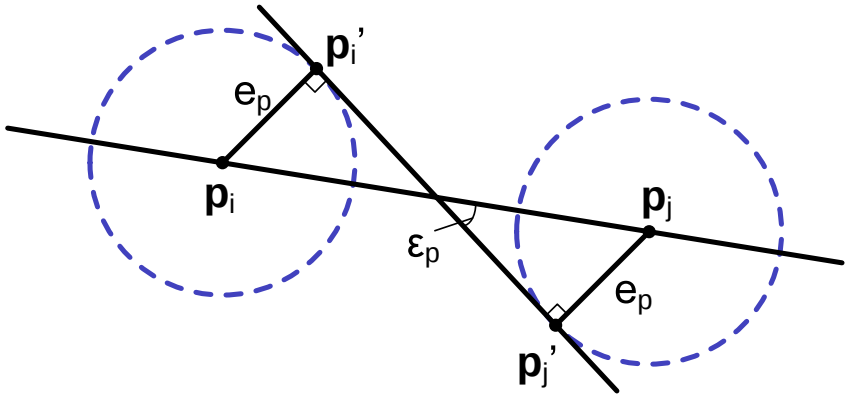


Figure 2.3: Given the maximum error e_p that the observed points $\mathbf{p}'_i$ and $\mathbf{p}'_j$ can have with respect to their true locations $\mathbf{p}_i$ and $\mathbf{p}_j$, this figure illustrates the maximum error ε_p that can occur in the observed orientation of the point pair $(\mathbf{p}_i, \mathbf{p}_j)$.

arbitrary measured points $\mathbf{p}'$ and $\mathbf{q}'$ are found inside circles that are centered to the true point locations $\mathbf{p}$ and $\mathbf{q}$ and have fixed radii $e_p, e_q \in \mathbb{R}^+$ that indicate the maximum error the points are assumed to have, respectively. It is clear that the distance $d_p \in \mathbb{R}$ between the points $\mathbf{p}_i$ and $\mathbf{p}_j$ is now somewhere between the limits

$$||\mathbf{p}'_j - \mathbf{p}'_i|| - 2e_p \leq d_p \leq ||\mathbf{p}'_j - \mathbf{p}'_i|| + 2e_p, \tag{2.12}$$

and, similarly, distance $d_q \in \mathbb{R}$ between the points $\mathbf{q}_u$ and $\mathbf{q}_v$ is somewhere between the limits

$$||\mathbf{q}'_v - \mathbf{q}'_u|| - 2e_q \leq d_q \leq ||\mathbf{q}'_v - \mathbf{q}'_u|| + 2e_q. \tag{2.13}$$

It follows from the inequalities 2.12 and 2.13 that the true scale parameter $s = d_q/d_p$ is somewhere on the interval

$$\frac{||\mathbf{p}'_j - \mathbf{p}'_i|| - 2e_p}{||\mathbf{q}'_v - \mathbf{q}'_u|| + 2e_q} \leq s \leq \frac{||\mathbf{p}'_j - \mathbf{p}'_i|| + 2e_p}{||\mathbf{q}'_v - \mathbf{q}'_u|| - 2e_q}. \tag{2.14}$$

This gives the error bounds of the scaling parameter.

Bounds for the rotation parameter θ can be established in a similar manner. Figure 2.3 illustrates the worst case scenario from the rotation point of view. Both of the points $\mathbf{p}'_i$ and $\mathbf{p}'_j$ have the maximum error e_p in their locations. The direction of the error is such that the angle $\varepsilon_p \in [0, \pi/2]$ is maximized. The true orientation θ_p of the vector $\mathbf{p}_j - \mathbf{p}_i$ is now somewhere between the limits

$$\theta'_p - \varepsilon_p \leq \theta_p \leq \theta'_p + \varepsilon_p, \tag{2.15}$$

where θ'_p is the angle of vector $\mathbf{p}'_j - \mathbf{p}'_i$. Maximum error ε_p can be calculated as

$$\varepsilon_p = \tan^{-1} \frac{2e_p}{||\mathbf{p}'_j - \mathbf{p}'_i||}, \tag{2.16}$$

where $\tan^{-1}(\cdot)$ denotes the arcus tangent. Respectively, for the point pair $(\mathbf{q}_u, \mathbf{q}_v)$ we get

$$\theta'_q - \varepsilon_q \leq \theta_q \leq \theta'_q + \varepsilon_q, \tag{2.17}$$

where θ'_q is the angle of vector $\mathbf{q}'_u - \mathbf{q}'_v$ and

$$\varepsilon_q = \tan^{-1} \frac{2e_q}{||\mathbf{q}'_v - \mathbf{q}'_u||}. \tag{2.18}$$

Inequalities 2.15 and 2.17 result in the true rotation parameter $\theta = \theta_q - \theta_p$ being on the interval

$$(\theta'_q - \theta'_p) - (\varepsilon_p + \varepsilon_q) \leq \theta \leq (\theta'_q - \theta'_p) + (\varepsilon_p + \varepsilon_q). \tag{2.19}$$

These are the error bounds for the rotation.

Given now a scale-rotation pair, instead of accumulating a corresponding bin in an uniform accumulator array, we construct a rectangle, whose dimensions are given by inequalities 2.14 and 2.19. The closer the points $\mathbf{p}'_i$ and $\mathbf{p}'_j$ or the points $\mathbf{q}'_u$ and $\mathbf{q}'_v$ are to each other the more dominant the error terms e_p and e_q become, thus, resulting in larger rectangles. Notice that the inequalities 2.16 and 2.18 assume that $||\mathbf{p}'_j - \mathbf{p}'_i|| > 2e_p$ and that $||\mathbf{q}'_i - \mathbf{q}'_j|| > 2e_q$. This is reasonable as this would otherwise allow the noise to be so powerful that some of the points in the sets would be able to change places. No point pattern matching method could survive that.

The final step is to find the scale-rotation cluster suggesting the scale and rotation of the transformation that we are ultimately searching for. As the scale-rotation pairs now each form a rectangle indicating its error bounds, the problem – instead of just finding the maximum from an accumulator array – becomes finding the area where most rectangles intersect. This is done by a sweep line algorithm, familiar from the field of computer graphics (*scanline rendering* [22]). The principle of the sweep line algorithm is simple: The scale-rotation space is sweeped with a horizontal scanline that moves in y-direction simultaneously keeping track on rectangles, which intersect the scanline. The area where most rectangles are intersecting at the same time is stored in the memory. After scanning all the rectangles, the area with the most rectangles intersecting has been determined.

As can be figured out from the huge amount of point pairs to process, the Chang's method is computationally heavy. Chang et al. ease the calculation by testing some early exit conditions, which break the execution as soon as an adequate amount of scale-rotation pairs appear in some cluster. Nevertheless, the complexity of the algorithm remains at

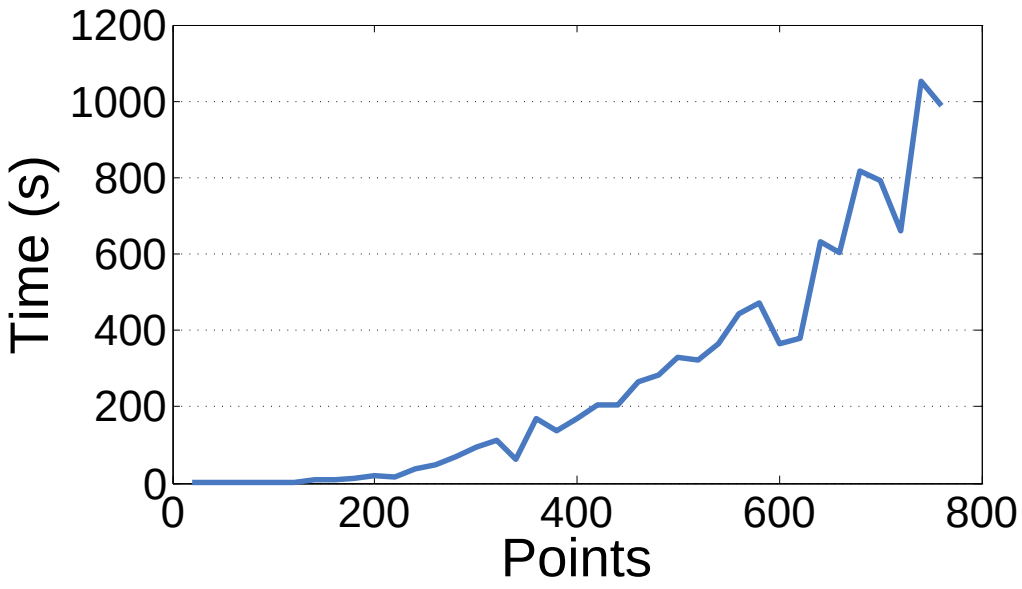


Figure 2.4: The execution time of the improved PPM algorithm of Chang et al. [21] was tested with respect to the number of matched points by generating random, uniformly distributed point sets from the interval $\{(x,y) \mid -1 < x < 1, -1 < y < 1\}$. *The point sets were matched against sets that were derived from the original ones by shuffling the points in a random order, transforming them with an arbitrary similarity transformation, and adding normally distributed noise with standard deviation* $\sigma = 0.0001$. *The number of points in the sets was gradually increased in steps of 20. Execution times of eight iterations per set size were averaged to get better results.*

$O(n^4)$, where n is the number of points when the sets are about the same size.

The described improvement further on increases the worst case execution time into $O(n^4 \log n)$ as the sweep line algorithm needs to do sorting. Fortunately, the additional logarithmic term is quite marginal when compared to the original complexity. Pairing 150-200 points typically takes about one second on an unoptimized platform. However, if the number of points in the sets get larger, the execution times of both the Chang's original and the improved algorithm rapidly increase, as illustrated in Figure 2.4. In practice, the execution time highly depends on the number of outliers, particularly in set P. This results in fluctuations in the actual computation times. However, the overall trend stays the same.

Despite the high complexity of the Chang's PPM algorithm, the method still takes its place amongst the other algorithms. This is because of the properties the Chang's method has that were discussed earlier. The developing of the improved algorithm can be easily justified by a simple example considering the memory requirements of the original algorithm versus the improved algorithm:

Assume that you have a point set having points very close to each other as well as points very far from each other. This requires that you use a rather dense accumulator array in the Chang's PPM algorithm in order for wrong scale-rotation pairs not to fall into the bin of the correct scale and rotation. Let say that you need to determine the rotation

term θ is with a precision of 0.01 degrees and the scaling term s with a precision of 0.001. After assuming that the right scale s is less than 100, the searched scaling term is limited into the interval $(0, 100]$. With these requirements, the accumulator array has to be of size 36000×100000 bins. Assume that you are pairing two sets with both having 200 points. This creates $200 \cdot 200 \cdot (200-1) \cdot (200-1)/4 = 396010000$ scale-rotation pairs, which requires (in theory) a 32 bit accumulator array to be used. Hence, the total amount of memory required for the accumulator array would be $36000 \cdot 100000 \cdot 32\text{b} \approx 13\text{GB}$.

In the improved algorithm, storing the coordinates of all the 396010000 rectangles with a typical 64b precision would require $4 \cdot 396010000 \cdot 64\text{b} \approx 12\text{GB}$ of memory, i.e., almost as much as with the old algorithm. However, Chang's method operates by fixing one point from both of the sets P and Q at a time and then forms all the possible scale-rotation pairs. This limits the amount of scale-rotation pairs at once to be only $(m-1)(n-1)$. Hence, in the example case, memory required to store the coordinates of the rectangles in the improved algorithm is at most $4 \cdot (200-1) \cdot (200-1) \cdot 64\text{b} \approx 1.2\text{MB}$, i.e., about 10000 times less than with the original algorithm. In addition, no compromises regarding to the resolution of the scale-rotation pairs need to be done.

2.2.2 Determining transformation parameters

The correspondence between the n-dimensional point sets P and Q defined in equations 2.5 and 2.6 has now been determined. Next, for more convenient notation, P and Q are redefined to be ordered sequences of n-dimensional points such that their elements form subsets of the old P and Q, i.e.,

$$P = \{\mathbf{p}_k \in \mathbb{R}^n\}_{k=1}^{m} \tag{2.20}$$

and

$$Q = \{\mathbf{q}_k \in \mathbb{R}^n\}_{k=1}^{m}, \tag{2.21}$$

where $m \leq \min\{m_p, m_q\}$ is the number of found point pairs. Further, for all $i \in \{1, \ldots, m\}$ point $\mathbf{p}_i$ corresponds to the point $\mathbf{q}_i$. We call the new P and Q as "ordered sets" or just as "sets".

Next, a transformation of some kind needs to determined in order to transform set P as close to set Q as possible. It is desirable to use a transformation that minimizes the mean squared distance between the two point sets. We would also like to end up with a closed form solution for determining the optimal transformation parameters.

We will now concentrate on finding the optimal rigid and similarity transformations between sets P and Q. In other words, we need to find the parameters s, $\mathbf{R}$, and $\mathbf{t}$ (as

described in Equation 2.7) that minimize the sum of the squared errors

$$J(s,\mathbf{R},\mathbf{t}) = \sum_{k=1}^{m} \|\mathbf{q}_k - (s\mathbf{R}\mathbf{p}_k + \mathbf{t})\|^2. \tag{2.22}$$

Several results for finding the optimal transformation parameters have been published. A good summary of optimal closed form solutions for various transformation types is available by Škrinjar [23]. The solution for optimal similarity transformation can be found from Umeyama's paper [24]. The optimal method corrects a minor flaw in the previous work by Arun and Horn [25]. The method uses *singular value decomposition (SVD) of the covariance matrix* of the point location data. The optimal method works for any dimensional point sets and can also be used for finding the rigid transformation.

In addition to the method using SVD (which is usually solved by using iterative algorithms like the one implemented in the linear algebra library LAPACK [26]), there are also simple closed form solutions for special cases available. One such solution, presented by Chang et al.[21], is for similarity transformations in 2-d. Their solution is achieved by solving a simple linear optimization problem. Surprisingly, using the Chang's method for finding the solution for the corresponding rigid transformation results in a constrained non-linear problem. This problem can be solved by using Lagrange multipliers. This is done in Appendix A. Another way to achieve the optimal rigid transformation is to first calculate the optimal similarity transformation, which has the same rotation parameter as the rigid transformation. This fact follows from Theorem 1, which is to be introduced later. The translation parameter, in turn, can be calculated by using a result that will be stated in Lemma 1.

Despite the already available solutions, we will introduce a theorem that can be useful in many applications. The theorem is a generalized version of the Umeyama's result giving the transformation that minimizes the *weighted sum of the squared errors*, i.e.,

$$J_w(s,\mathbf{R},\mathbf{t}) = \sum_{k=1}^{m} w_k \|\mathbf{q}_k - (s\mathbf{R}\mathbf{p}_k + \mathbf{t})\|^2, \tag{2.23}$$

where $w_1,\ldots,w_m \in \mathbb{R}$ are non-negative weights such that at least one of them is non-zero. From this point on we are assuming that the weights sum up to one. Notice that the weights can always be scaled this way without affecting the minimization problem.

To be able to use matrix algebra, we will next introduce the formulation of *point set matrix*.

Definition 2. *Given point set* $A = \{\mathbf{a}_k \in \mathbb{R}^n\}_{k=1}^m$, *the* $n \times m$ *matrix* $\mathbf{A} = [\mathbf{a}_1,\ldots,\mathbf{a}_m]$ *is called the point set matrix corresponding to set A.*

Next, let matrices $\mathbf{P}$ and $\mathbf{Q}$ be point set matrices of sets P and Q, respectively. The weighted square error in Equation 2.23 can now be written in matrix notation as

$$J_w(s,\mathbf{R},\mathbf{t}) = \left\|\mathbf{QW} - \left(s\mathbf{RP} + \mathbf{t}\mathbf{1}^T\right)\mathbf{W}\right\|_F^2, \tag{2.24}$$

where $\mathbf{1} = [1,\ldots,1]^T$ and $\mathbf{W} = \mathrm{diag}(\sqrt{w_1},\ldots,\sqrt{w_m})$. Notation $\|\cdot\|_F$ denotes the Frobenius norm, which for matrix $\mathbf{A} \in \mathbb{R}^{n\times m}$ is defined as the square root of the sum of its squared elements [27, p. 55] or, equivalently,

$$\|\mathbf{A}\|_F = \sqrt{\mathrm{tr}(\mathbf{A}^T\mathbf{A})}. \tag{2.25}$$

Before stating the theorem that gives the optimal transformation parameters for the minimization problem in Equation 2.24, let's introduce the definition of *weighted centroid* and one important intermediate result.

Definition 3. *Given the non-negative weights $w_1,\ldots,w_m \in \mathbb{R}$ (that sum up to one), the weighted centroid $\boldsymbol{\mu}_w \in \mathbb{R}^{\mathrm{n}}$ of point set $A = \{\mathbf{a}_k \in \mathbb{R}^{\mathrm{n}}\}_{k=1}^m$ is defined as*

$$\boldsymbol{\mu}_w = \sum_{k=1}^{m} w_k\mathbf{a}_k = \mathbf{Aw},$$

where $\mathbf{A}$ is the point set matrix of set A and $\mathbf{w} = [w_1,\ldots,w_m]^T$.

Weighted centroid is analogous to the center of gravity in physics.

By using the definition of the weighted centroid, we now state an intermediate result that reveals a nice property of the optimal transformations.

Lemma 1. *Given the non-negative weights $w_1,\ldots,w_m \in \mathbb{R}$ (that sum up to one), if there is a rigid or a similarity transformation that transforms point set $P = \{\mathbf{p}_k \in \mathbb{R}^{\mathrm{n}}\}_{k=1}^m$ in such a way that the weighted mean square error between transformed P and another point set $Q = \{\mathbf{q}_k \in \mathbb{R}^{\mathrm{n}}\}_{k=1}^m$ is minimized, transformed P and set Q then have the same weighted centroid.*

Proof. The proof for Lemma 1 can be achieved by noting that the Equation 2.23 is quadratic with respect to the translation parameter $\mathbf{t}$. Hence, the minimum is where the derivative is zero. [28] By calculating the derivative, setting it to zero, and solving for $\mathbf{t}$, it can be shown that the optimal translation translates the weighted centroid of point set P into the weighted centroid of set Q. □

Now we are ready to state the theorem, which gives the optimal weighted similarity or rigid transformation between two n-dimensional point sets.

Theorem 1. *Consider point set matrices $\mathbf{P}_w = (\mathbf{P} - \boldsymbol{\mu}_p\mathbf{1}^T)\mathbf{W}$ and $\mathbf{Q}_w = (\mathbf{Q} - \boldsymbol{\mu}_q\mathbf{1}^T)\mathbf{W}$, where $\mathbf{P},\mathbf{Q} \in \mathbb{R}^{n\times m}$ are point set matrices and $\boldsymbol{\mu}_p,\boldsymbol{\mu}_q \in \mathbb{R}^{\mathrm{n}}$ are the weighted centroids corresponding to sets P and Q, respectively. Matrix $\mathbf{W} \in \mathbb{R}^{m\times m}$ is a diagonal weight matrix. Now, the orthogonal rotation matrix $\mathbf{R} \in \mathbb{R}^{\mathrm{n}\times\mathrm{n}}$, the scaling factor $s \in \mathbb{R}^+$, and the*

translation vector $\mathbf{t} \in \mathbb{R}^n$ *of the similarity/rigid transformation that transform set P such that the weighted mean square error between set Q and transformed P is minimized are*

$$\mathbf{R} = \mathbf{U}\mathbf{S}\mathbf{V}^T, \tag{2.26}$$

$$s = \begin{cases} 1, & \textit{for rigid transformation} \\ \frac{\mathrm{tr}(\mathbf{D}\mathbf{S})}{\|\mathbf{P}\|_F^2}, & \textit{for similarity transformation} \end{cases}, \tag{2.27}$$

and

$$\mathbf{t} = \boldsymbol{\mu}_q - s\mathbf{R}\boldsymbol{\mu}_p, \tag{2.28}$$

where $\mathbf{U}\mathbf{D}\mathbf{V}^T$ *is the singular value decomposition of* $\mathbf{Q}_w\mathbf{P}_w^T$ *and*

$$\mathbf{S} = \begin{cases} \mathbf{I}_n, & \textit{if } \det(\mathbf{Q}_w\mathbf{P}_w^T) \geq 0 \\ \mathrm{diag}(1,1,\ldots,1,-1), & \textit{otherwise} \end{cases}, \tag{2.29}$$

where $\mathbf{I}_n$ *is* $n \times n$ *identity matrix.*

Proof. To prove Theorem 1, Lemma 1 can be used for subtracting the weighted centroids $\boldsymbol{\mu}_p$ and $\boldsymbol{\mu}_q$ from both of the sets P and Q, respectively. This way the optimal translation parameter becomes zero without affecting the rotation and scaling term. The minimization problem in Equation 2.24 is now reduced to minimizing

$$J_w(s,\mathbf{R}) = \|\mathbf{Q}_w - s\mathbf{R}\mathbf{P}_w\|_F^2. \tag{2.30}$$

A formal proof that $\mathbf{R}$ and s from Equations 2.26 and 2.27 will minimize the criterion in Equation 2.30 can be found from Umeyama's paper [24]. Finally, as the optimal rotation and scaling parameters are known, it is a straight consequence of Lemma 1 that the optimal translation will take the form seen in Equation 2.28. □

The essence of Theorem 1 (which is to introduce weighting into the minimization problem when seeking for the optimal rigid and similarity transformations) has also been reported by Schaefer et al. [28] and Müller et al. [29]. However, they do not take into account the condition in Equation 2.29, which was introduced by Umeyama [24] (already 15 years earlier). This can sometimes result in invalid rotation matrices that perform reflection instead of rotation.

2.3 Generalized logical level thresholding

Thresholding is an image processing task, where objects of interest are to be extracted or segmented from the background. Usually gray-level images are considered. Pixels that have a value exceeding a given threshold are considered as object pixels and pixels that

have a value below the threshold are considered as background (or – depending on the application – vice versa). Hence, the resulting image is binary valued.

The problem in thresholding is to determine the right threshold value. Simple methods use a global threshold. One of the most popular global thresholding techniques is *Otsu's method* [30]. It chooses the threshold by maximizing the between-class variance of the pixel values below the threshold and the pixel values above the threshold.

Global methods have restrictions like inability to cope with changes in the lighting conditions. This can be handled by using local methods that choose the threshold value adaptively for each pixel. Another way to enhance the thresholding is to somehow incorporate structural information about the objects that need to be extracted. One such method is the *logical level thresholding technique* [31] that was originally developed for detection of characters from grayscale documents.

Logical level technique uses knowledge about the stroke width $p \in \mathbb{R}^+$. It assumes that all the lines in the image are drawn with a same kind of pencil and that the stroke width of the pencil is known. The method considers eight neighboring windows W_i around each image pixel (x,y) like illustrated in Figure 2.5. Each window is of size $(2p+1) \times (2p+1)$ pixels and their center points are located one stroke width away from the pixel under consideration. This makes the windows to overlap each other.

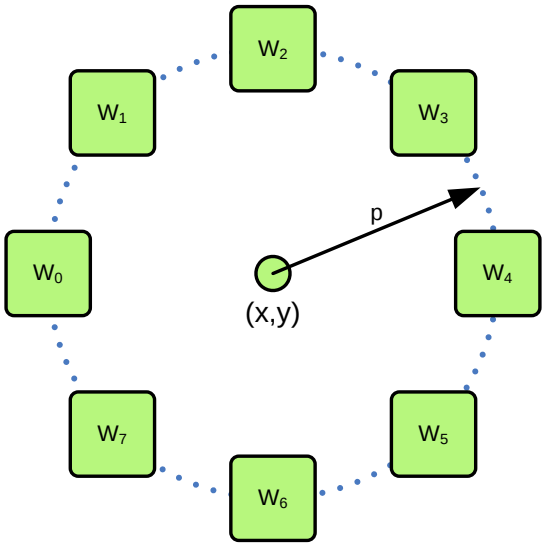


Figure 2.5: Logical level technique considers eight neighboring windows W_i of each image pixel (x,y). Each window is of size $(2p+1) \times (2p+1)$ and located one stroke width p from the center pixel. The layout of this figure originates from Kamel et al. [31].

Logical level technique seeks for points that are part of a dark line on a bright background. This is done by checking whether there are consecutive windows on opposite sides of the pixel with brighter average intensity than the pixel has. Mathematically, this is described such that each pixel of the output image B is assigned a value

$$B(x,y) = \begin{cases} 1, & \text{if } \bigvee_{i=0}^{3} \left[L(W_i) \wedge L(W_{i+1}) \wedge L(W_{i+4}) \wedge L(W_{(i+5) \bmod 8}) \right] \\ 0, & \text{otherwise} \end{cases}, \tag{2.31}$$

where $\vee$ and $\wedge$ denote logical OR and AND operations, respectively, and

$$L(W) = \begin{cases} \text{true}, & \text{if } \mu_w - I(x,y) > t \\ \text{false}, & \text{otherwise} \end{cases}. \tag{2.32}$$

In Equation 2.32, μ_w denotes the average pixel value of window W, $I(x,y)$ is the intensity value of the pixel (x,y), and t is a global threshold. Whenever the average value inside the window W is at least t units greater than the intensity of the pixel (x,y), $L(W)$ is true. Further on, pixel (x,y) is considered as an object pixel, i.e., $B(x,y)$ is true, whenever at least two subsequent windows and the windows to their opposite all set L as true. Using smoothed version of the image I in Equation 2.32 gives robustness against noise. Signal-dependent noise, nonuniform lighting conditions, shadows, low contrast, etc., can be handled by using an extended method that chooses t adaptively [32].

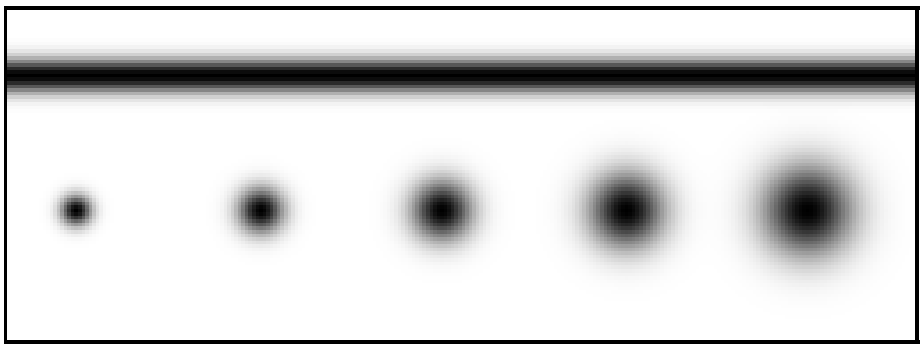
Logical level thresholding is computationally efficient. All possible local averages μ_w can be calculated at the same time in linear time (unlike convolution in general). After that, finding the right neighboring windows for each pixel (x,y) is just a matter of indexing the averaged image. The logical operations are naturally easy to compute as well.

The thresholding criterion in Equation 2.31 is designed to detect dark lines that go across the pixel (x,y). By replacing the criterion with a different kind of logical expression, the method can be extended also to detect other kinds of structures. By changing the direction of the inequality in Equation 2.32, bright objects against dark background can be detected instead of dark objects against bright background. We call the method that results after making these enhancements the *generalized logical level thresholding*.

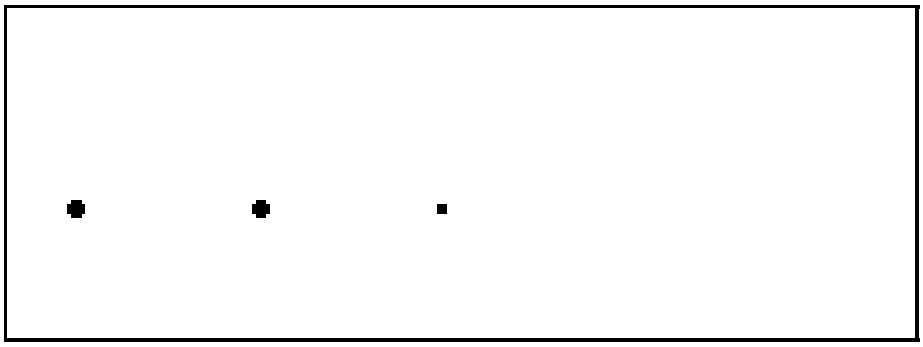
One application of the generalized logical level thresholding technique is to use it as a blob detector. In order to do this, the thresholding criterion in Equation 2.31 is altered such that

$$B(x,y) = \begin{cases} 1, & \text{if } \bigwedge_{i=0}^{7} L(W_i) \\ 0, & \text{otherwise} \end{cases}. \tag{2.33}$$

The criterion now requires that all the neighboring windows need to have higher intensity than the centering pixel before the pixel is considered as an object pixel. This results in a heuristics that detects point-like objects and uses the stroke width p to control the upper bound of the diameter of the detected objects. This is illustrated in Figure 2.6, which shows some blurry blobs and a line thresholded according to this heuristics. The choice of the parameters makes three of the smallest blobs to be detected while the line and the



(a)



(b)

Figure 2.6: The example image (a) shows a blurry line and blobs with different sizes. The objects have Gaussian profiles with standard deviations of 3.0, 4.4, 5.9, 7.3, and 8.7 for the blobs and 3.3 for the line. Figure (b) shows the results after applying the thresholding heuristics presented in Equation 2.33. Exceptionally, black color is indicating true and white false. The parameters used were $p = 7$ *and* $t = 0.5$ *while the pixel values of the original image are on the interval* $[0, 1]$.

larger blobs are discarded. Notice that the discarded line is almost as thin as the smallest blob.

2.4 Neural networks

Artificial neural networks, in this thesis referred to as *neural networks (NN)*, are machines that are designed to model the way how the human brain performs a function of interest. Similar to the brain, NNs consist of *neurons*, units of computation, which are connected to other neurons via *synapses*.

The idea of NNs as computing machines was introduced as early as 1943 by McCulloch and Pitts [33]. Later on in 1958 Rosenblatt used the McCulloch-Pitts neuron model to introduce the *perceptron* [34]. Perceptrons became the first models capable of *learning*. Other early stages of NNs include ADALINE (*adaptive linear neuron*) that was introduced in 1960 by Widrow and Hoff [35]. ADALINE deviates from the perceptron only by its learning characteristics.

In addition to learning, concepts that are associated with NNs are, *nonlinearity*, *adap-*

tivity, *complexity*, *parallelism*, and *robustness*. NNs have applications in fields like modeling, time series analysis, signal processing, and pattern recognition. Especially the last one is of interest in this thesis. The content of this section is mostly based on the book by Haykin [36].

2.4.1 Neuron

First, let's look at the structure of a single neuron. This is illustrated in Figure 2.7. A neuron defines an input-output relationship between its n inputs $x_1, \ldots, x_n \in \mathbb{R}$ and its output $y \in \mathbb{R}$. This relationship is characterized by the *synaptic weights* $w_1, \ldots, w_n \in \mathbb{R}$ and the *bias* $b \in \mathbb{R}$ of the neuron. The bias is summed together with the inputs of the neuron, which are weighted according to the synaptic weights. This creates an *activation potential* $v \in \mathbb{R}$. The activation potential is fed into the *activation function* $\varphi : \mathbb{R} \mapsto \mathbb{R}$ giving the output of the neuron.

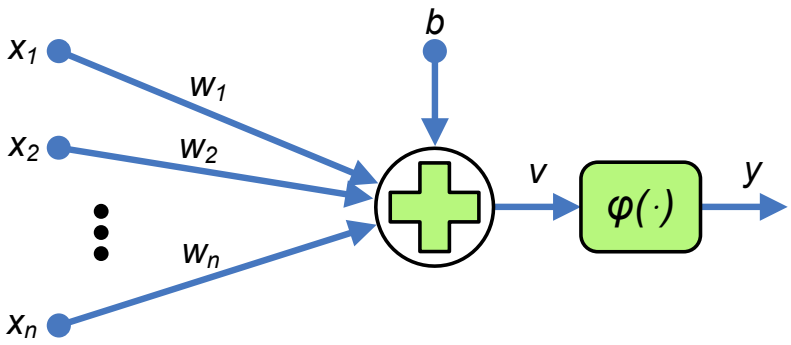


Figure 2.7: A neuron has n inputs $x_1, \ldots, x_n$, which are multiplied by the corresponding synaptic weights $w_1, \ldots, w_n$ and summed together with the bias term b. This results in the activation potential v, which goes into the activation function φ. The output y of the activation function is also the output of the neuron.

Mathematically, a neuron is described as

$$y = \varphi\left(\sum_{k=1}^{n} w_k x_k + b\right). \tag{2.34}$$

For more convenient notation, the bias can be regarded as one of the weights by setting $w_0 = b$ and $x_0 = 1$. In matrix notation, Equation 2.34 can now be rewritten as

$$y = \varphi\left(\mathbf{w}^T \mathbf{x}\right), \tag{2.35}$$

where $\mathbf{w} = [w_0, \ldots, w_n]^T$ and $\mathbf{x} = [x_0, \ldots, x_n]^T$.

The activation function is usually chosen to be nonlinear. This makes the neuron, and eventually the whole network, nonlinear. In addition to bringing in nonlinearities, the purpose of the activation function is to limit the activation potential into some interval.

Sigmoid functions are commonly used as activation functions as they are differentiable (which is essential in the learning phase of the NN). An example of a sigmoid function is the *logistic function*

$$\varphi(x) = \frac{1}{1+e^{-ax}}, \tag{2.36}$$

where $a \in \mathbb{R}$ is the slope parameter. Other common activation functions are the *Heaviside function*

$$\varphi(x) = \begin{cases} 0, & \text{if } x < 0 \\ 1, & \text{otherwise} \end{cases} \tag{2.37}$$

and the *piecewise-linear function*

$$\varphi(x) = \begin{cases} 0, & \text{if } x \leq -\frac{1}{2} \\ x, & \text{if } -\frac{1}{2} < x < \frac{1}{2} \\ 1, & \text{otherwise} \end{cases} . \tag{2.38}$$

The three introduced activation functions map the activation potential of a neuron into the interval $[0,1]$. Unlike real-life neurons, artificial neurons can also have a negative output. The logistic function can be scaled and biased to produce values from the interval $[-1,1]$. This results in a sigmoid function defined as

$$\varphi(x) = 2\frac{1}{1+e^{-ax}} - 1 = \frac{1-e^{-ax}}{1+e^{-ax}} = \tanh(x), \tag{2.39}$$

i.e., the *hyperbolic tangent*.

2.4.2 Feedforward networks

When the neurons connect with each other, i.e., the outputs of neurons become inputs of other neurons, they form a NN. Different network architectures can be conveniently described by *directed graphs*. A graph of one of the most common NN architectures is shown in Figure 2.8. This is the *layered feedforward* NN. Other NN architectures include *recurrent* NNs that have at least one *feedback* loop. Recurrence highly affects the learning and performance characteristics of the NNs. We will now solely focus on feedforward networks as they offer the most interesting applications from the viewpoint of this thesis.

In a layered architecture the neurons are organized as layers. Between the input and the output layer there are a number of *hidden layers*. Each neuron on a particular layer has connections only between neurons of the previous and the following layer. When all the possible connections exist, the NN is *fully connected*. When the direction of each connection is towards the output layer, the NN is said to be feedforward. Hence, graphs of feedforward NNs are *acyclic*. As multilayer NNs that have neurons with linear activation

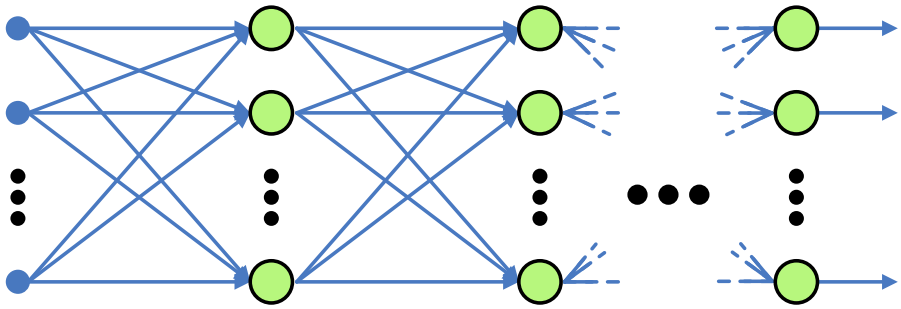


Figure 2.8: A fully connected, layered, feedforward NN. The NN has an input layer, arbitrary number of hidden layers, and an output layer.

functions are always reducible to single layer NNs, multilayer NNs are sensible only when they are nonlinear.

Feedforward NNs are commonly used in pattern recognition as *1 of M classifiers*. In pattern classification the input vector is classified into one of the M classes. The output layer of the NN has M neurons, each giving a *confidence level* of the input vector belonging to the corresponding class.

An interesting fact is that if the activation functions of the NN are logistic functions, the number of training data used for training the NN is large, the training samples are identically and independently distributed, and the learning has finished successfully, the outputs of the NN asymptotically approximate the *a posteriori* probabilities of the input vector belonging to each class [37]. However, there are usually a number of tricks that are done in the learning phase, e.g., to improve the numerical stability of the learning algorithm. These modifications can change the basic assumptions that lead into the property of the network outputs being probabilities. Hence, the outputs of the NN should be considered something else, e.g., confidence levels.

Whatever the outputs of the NN are called, the classification rule can be made simple: the input vector is classified into the class with highest output. In two-class problems only one output neuron is adequate. Negative output indicates the first class while positive output indicates the other one (in the case the output neuron has an odd activation function like hyperbolic tangent).

2.4.3 Learning

Regardless of the network architecture, the most important feature of NNs is their ability to *learn*. In this thesis only *supervised* learning is considered. In supervised learning, the NN has a *teacher* showing the NN pairs of *training data* defined as a set of labeled examples $\{(\mathbf{x}_i, \mathbf{d}_i)\}_{i=1}^{N}$, where $\mathbf{x}_i \in \mathbb{R}^n$ are input data vectors and $\mathbf{d}_i \in \mathbb{R}^m$ their desired outputs such that n is the amount of inputs in the neurons on the first layer and m is the amount of neurons on the output layer. The NN then learns to map the input data

according to the desired outputs. However, the ultimate goal is that the NN would also *generalize* to handle unseen input data with a proper way.

Learning algorithms are used to teach NNs. The purpose of a learning algorithm is to find the biases and the synaptic weights of the NN that minimize a given cost function like the mean square error between the network output and the desired output. In case of multilayer feedforward NNs the learning algorithm used is usually some variant of the *back-propagation algorithm*. The basic principle of the back-propagation algorithm is now discussed more thoroughly.

For each training data sample $(\mathbf{x}_i, \mathbf{d}_i)$, the back-propagation algorithm has two stages. First, the output of the NN is calculated with the input sample $\mathbf{x}_i$. Second, each synaptic weight $w_{ij} \in \mathbb{R}$ (denoting the weight of the synapse from neuron i to neuron j, w_{0j} being the bias term of neuron j) is updated according to the *delta rule* [35] (also known as the *LMS algorithm*). The delta rule is defined as

$$\Delta w_{ij} = \eta \delta_j y_i, \tag{2.40}$$

where $\eta \in \mathbb{R}^+$ is the *learning-rate parameter*, $\delta_j \in \mathbb{R}$ is the *local gradient* of neuron j, and $y_i \in \mathbb{R}$ is the output of neuron i (one of the inputs of neuron j, that is). The local gradient δ_j is calculated by using the *back-propagation formula*

$$\delta_j = \varphi_j'(v_j) \sum_k \left(\delta_k w_{jk}\right), \tag{2.41}$$

where φ_j' denotes the derivative of the activation function of neuron j with respect to the activation potential v_j of the same neuron. The summation goes through all the synapses that the neuron has to the following layer. In case the neuron is on the output layer, δ_j is calculated as

$$\delta_j = \varphi_j'(v_j)(d_j - y_j), \tag{2.42}$$

where d_j is the desired output and y_j is the true output of the j^{th} neuron on the output layer. The latter one is calculated in the first stage of the algorithm.

Notice, that in Equation 2.41 the local gradient of the neuron depends on the local gradients of the neurons on the following layer. That is why the updating of the synaptic weights starts from the output layer and then proceeds layer by layer towards the input layer. Thus, the name back-propagation algorithm. For more information on the back-propagation algorithm see the corresponding chapter from Haykin's book [36, chap. 4.3].

The updating of the synaptic weights continues iteratively until some stopping criteria are met. These criteria can incorporate, e.g., error between the outputs and the desired outputs, classification error with a validation data, the gradient of these errors, time elapsed since the start of the training process, and times that the training data has fully been intro-

duced to the network (*epochs*, that is).

Once the pattern recognition NN has learned, its performance can be tested with a set of *testing data* similar to the training data set. The NN is fed with N test samples $\mathbf{x}_i$ fully independent of the used training data to see how well it can handle unseen data. This is measured by classification performance $p_c \in \mathbb{R}$, $0 \leq p_c \leq 1$. A maximum likelihood estimate $\widehat{p_c}$ of the classification performance can be calculated as

$$\widehat{p_c} = \frac{A}{N}, \tag{2.43}$$

where A is the number of correct classifications when classifying the N testing samples.

Assuming that the actual classification performance p_c (the probability of a right classification) is the same for each input-output sample, an upper bound $\delta_c \in \mathbb{R}^+$ for the probability that the error of the estimate $\widehat{p_c}$ is larger than a given tolerance $\varepsilon \in \mathbb{R}^+$ is given by the *Chernoff bound* [38]

$$P(|\widehat{p_c} - p_c| > \varepsilon) < 2e^{-2\varepsilon^2 N} = \delta_c. \tag{2.44}$$

By solving N, we notice that if we want to have an estimate of the classification performance that has error not more than ε with a confidence of $1 - \delta_c$, a test data of

$$N = \frac{1}{2\varepsilon^2} \ln \frac{2}{\delta_c}. \tag{2.45}$$

samples is guaranteed to be adequate. For example, if one would want to estimate the classification performance with an accuracy of one percent ($\varepsilon = 0.005$) and with a confidence of 95% ($\delta_c = 0.05$), one would need approximately 7400 testing samples. In practice, as the amount of available data is limited, some of the data is used for training and what remains is used for the testing of the NN.

In cases where there is low amount of available training data compared to the number of free parameters (weights and biases) of the NN, there is a possibility of *overfitting* the data. In overfitting, the NN learns to describe the noise in the training data instead of the underlying phenomena. This leads to poor generalization.

To avoid overfitting, a portion from the training data can be taken aside as *validation data*. The training of the NN is then performed with the remaining training data and, as the learning goes on, validated by using the validation data. Normally, the performance with the actual training data keeps rising as the training goes on. However, at some point, overfitting starts to occur and the performance with the validation data begins to deteriorate. The best generalization is achieved when the training is stopped before this happens.

It is good to notice that the testing and validation data should be kept separate. This is because the testing data is fully independent of the training process of the NN, which is a necessary property when measuring the performance of the NN after it has been trained.

2.4.4 Pros and cons

NNs have both advantages and disadvantages. The advantages include the capability to solve highly nonlinear problems in a robust manner. In addition, the implementation can easily exploit from the parallel structure of NNs. One of the key advantages of the NNs is the adaptivity concerning the learning. NNs are easy to retrain online, which brings in their nature of *nonlinear adaptive filters*.

Drawbacks of NNs are mostly related to the learning. The learning algorithms depend highly on the initial state (biases and synaptic weights) of the network and can easily get stuck on local optima or saturate due to numerical inaccuracies. Several tricks are needed to overcome these problems. These include randomizing the order of the training data, normalization of the input data, using a momentum coefficient and a neuron dependent learning-rate parameter in the delta rule of the back-propagation algorithm, and assigning the desired network outputs in some special manner.

One issue is overfitting, which was mentioned earlier. In addition to low amount of training data compared to the complexity of the network, overfitting may also occur when the quality of the training data is bad or when the structure of the NN is unsuitable for the job at hand. Determining the optimal structure of the NN can be tricky as there are no absolute rules for designing NNs. Instead, there are mostly just guidelines and heuristics proven to work in practice. Quoting Haykin [36, see p.178]:

> It is often said that the design of a neural network using the back-propagation algorithm is more of an art than a science in the sense that many of the numerous factors involved in the design are the results of one's own personal experience.

3. IN-PROCESS PRINT FILE CREATION

This chapter describes how the print file creation system works in practice. In section 3.1, we will introduce the basic equipment needed to utilize the system. Section 3.2 gives an overview on how the developed correction process works. In section 3.3 we describe the setup for the imaging and camera calibration. Finally, section 3.4 gives a detailed description of the print file creation algorithm.

3.1 Facilities

One of the major advantages of the developed correction system is that it uses software to do most of the job. The only hardware needed – in addition to the actual printing equipment of course – is a PC (personal computer) and an imaging device, which are both available off-the-shelf. Using these standard equipment together with high-end software instead of custom made hardware makes the system inexpensive, flexible, and easily updateable.

As the system works online during the actual manufacturing process, the imaging device that captures the images of the printed modules has to be installed such that it can

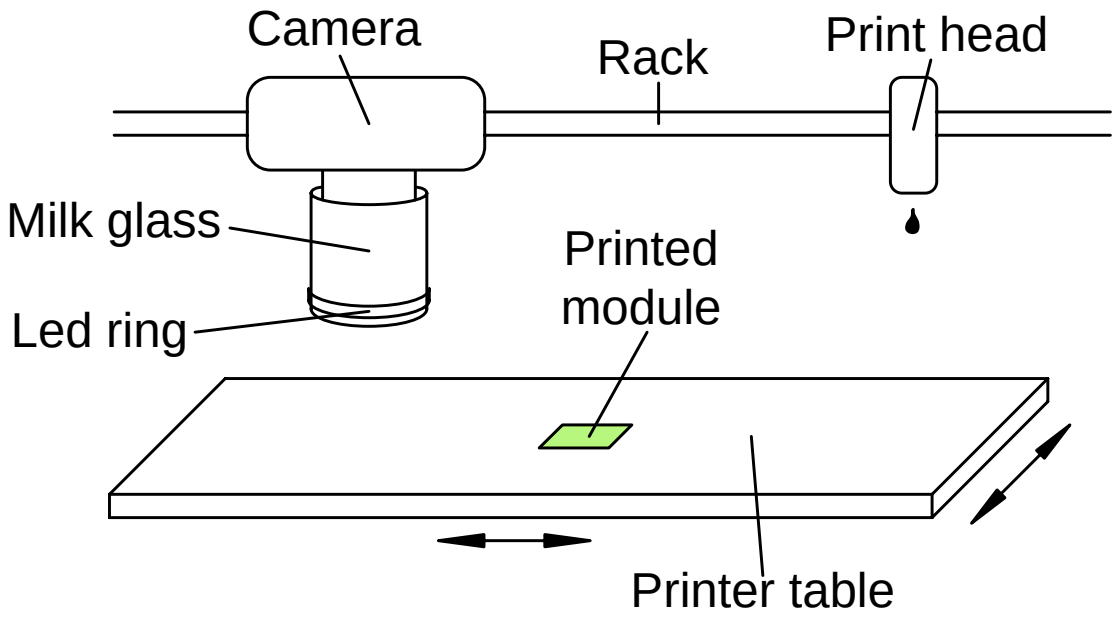


Figure 3.1: An example setup that uses mobile printer table below a static rack that includes the imaging device as well as the printing related hardware like the print head.

easily be moved above the printing site. Figure 3.1 illustrates one possible setup. In the example setup the imaging device is mounted on a static rack that also includes the print head (that holds the print nozzles and ink cartridges), a UV-light source (for curing of the printed material), and other printing related equipment. The printer table below the rack is mobile and can use, e.g., vacuum to hold the printed modules in place. As the offset between the print head and the imaging device is known, the translation between printing and imaging phases can be made automatic.

The real-life testing and validation of the correction system was conducted with hardware setup equal to the one in Figure 3.1. The printer used was iTi (Imaging technology International) XY MDS2.0 located in the Department of Electronics at Tampere University of Technology. The printer includes a mounting rack and a printer table similar to that of Figure 3.1. Canon EOS 5D digital camera with Canon MP-E 65 mm macro lens was used as an imaging device. Vertically adjustable mounting bracket along with the zoom of the objective were used to control focus and magnification of the images. The camera was connected to a PC with a USB (universal serial bus) interface. In addition, a translucent glass with led ring was installed around the space between the objective and the target in order to illuminate the target area and to avoid distracting reflections from the surrounding environment.

3.2 Correction process overview

Assume that a single module (see the definition of a module in section 1.1) has been molded and is ready for printing of the wiring. Also assume that the equipment are all set and that the camera has been calibrated and is ready for imaging (see section 3.3 for camera calibration). First, the module to be printed is photographed and the resulting image is transferred to the PC. The subsequent processing steps are illustrated in Figure 3.2.

The processing starts with image normalization, which eliminates the effects of the lighting conditions and camera automatics to the image. This is done with some standard tricks including white balancing [39] and contrast stretching [11, see p.77]. These topics are beyond the scope of this thesis.

The first actual processing block is the *feature detection*. This is divided into two parts: IC extraction and connection point detection. The feature detection block automatically detects the connection points of each IC. Notice that ICs are rigid bodies and the locations of the IC connection points inside each IC are accurately known by the design. This would suggest that there is no need to detect the connection points at all after the IC itself has been located. However, by considering the connection points individually, the correction algorithm is easily modified to detect also other kinds of control points, e.g., connection points of various discrete components. Further on, the flexible concept of control points makes the detection step portable also into completely other applications of inkjet printing. One such example is a bio application where we would like to detect

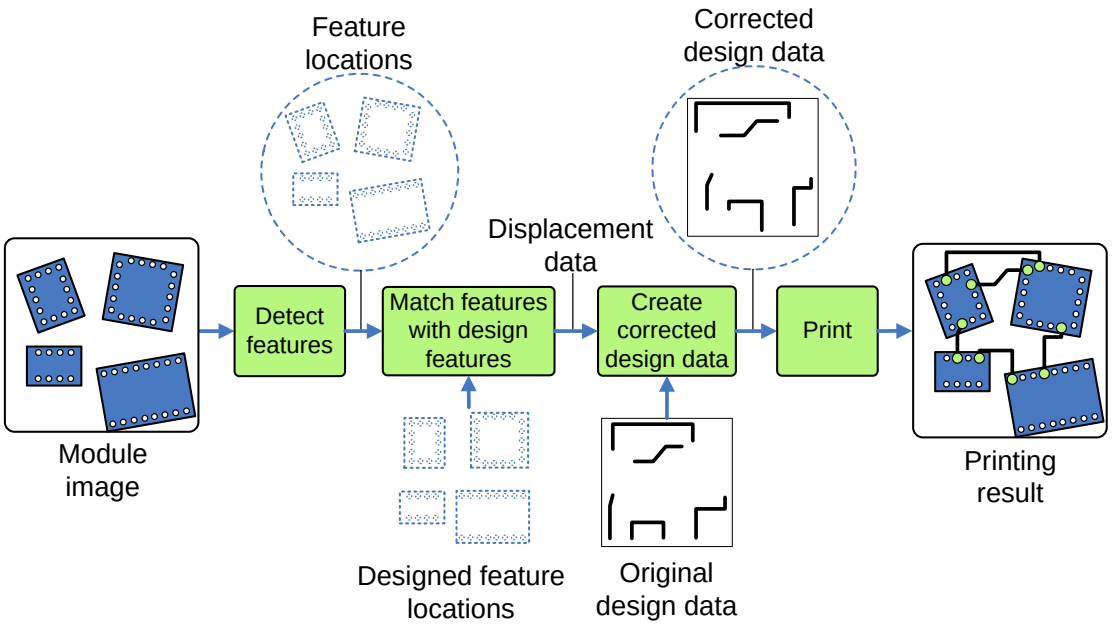


Figure 3.2: Diagram of the correction process.

and print organic cells and structures.

After the first processing block has detected the connection points, the next block *matches the detected points* with the corresponding points in the design data. In this block, the designed connection point locations are paired with their true locations in order to determine the error, i.e., the *displacement data*.

The displacement data is fed into the next block, which *creates a new wiring diagram* by altering the original one. Different methods to produce the corrected design data are considered in section 3.4.4. Finally, the corrected design is passed to the printer for printing. The result is a functioning module. A more detailed description of the correction algorithm is given in section 3.4.

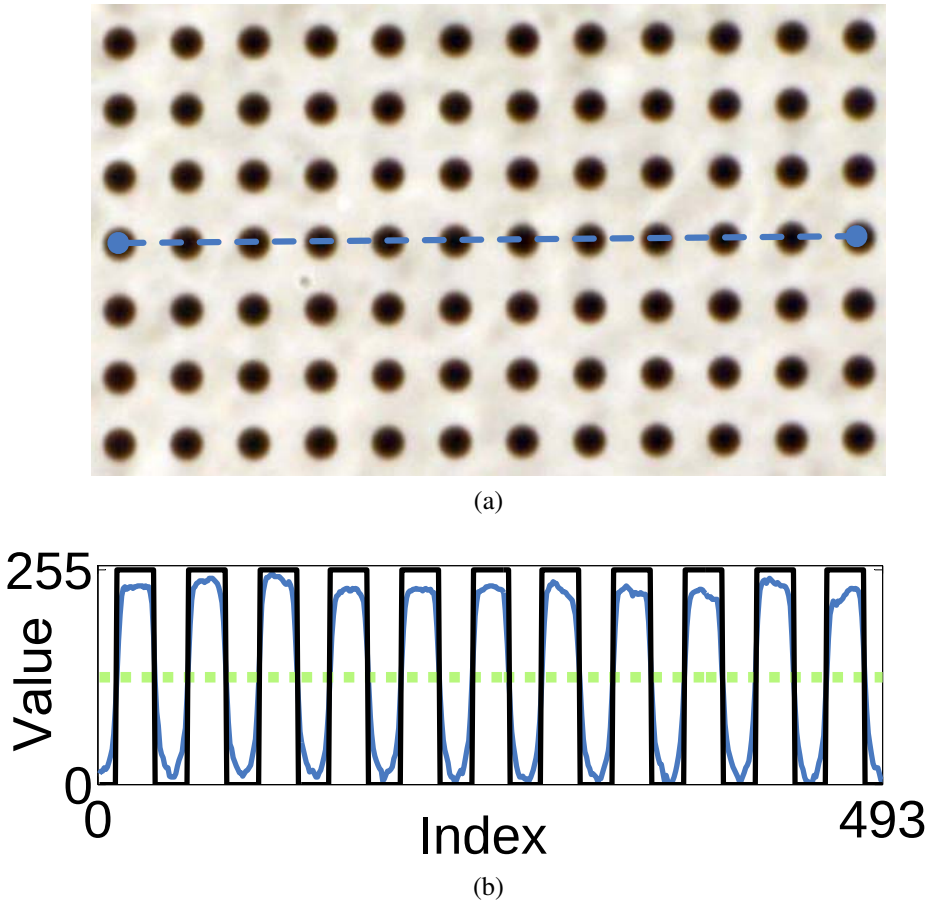
3.3 Camera sensor calibration and imaging

After the camera has been installed to the printer, it has to be calibrated. Calibration is needed only once, prior to the first print. Recalibration is necessary only if the mounting position or the lens setting of the camera changes. Variability in lighting conditions etc. can be handled by software like mentioned in the previous section.

The first thing to do is to manually adjust the focus and the magnification of the optics. The focus should be as sharp as possible and the magnification should be such that a single module is filling the image area as fully as possible, which maximizes the image resolution. As the module surface is not necessarily level, the aperture should be set as narrow as possible to achieve a wide depth of field. The camera automatics can be set to handle the rest of the options. This enables an optimal exposure.

The actual calibration consists of two steps. The first step is to determine the image resolution, i.e., how many pixels does the image have per one millimeter on the module plane. The second step is to determine the camera rotation, which is the angle between a straight line on the printer table and the same line in the image. Depending on the used imaging device and the type of its mounting, also other kinds of calibration procedures may be necessary. These can include, e.g., determining the deviation of the optical axis of the camera from the normal of the printer table. This information can be used in compensating the projection error or modeling the image distortions due to camera optics. These kind of calibration steps are left outside consideration in this thesis.

The image resolution can be determined by using a dimensionally accurate body. Calibration plates designed for that purpose work ideally. Alternatively, a dedicated IC can be used to measure the resolution as IC dimensions are usually quite accurate.



(a)

(b)

Figure 3.3: An example of an ad-hoc method, which can be used for determining the image resolution.

Figure 3.3 shows an example on how the resolution can be determined by using a calibration plate and some ad-hoc methods. Figure 3.3a shows a small portion of a calibration plate by Edmund Optics. On a glass plate there is a grid of chrome dots with spacing of $0.25 \pm 0.001 mm$. When determining the camera resolution, the operator first picks two

dots from the same row. These are shown with the blue dots. Image pixels on the line segment between these dots form a profile. The profile is plotted with blue line in Figure 3.3b. A threshold value (dashed green line) is then calculated with Otsu's method [30] and the profile is thresholded. The thresholded profile is represented by the black line. Finally, the amount of pixels on the profile is divided by $0.25mm$ times the amount of spaces on the profile, which can be calculated from the thresholded profile. In the example case the resulting resolution would be $493/(11 \cdot 0.25mm) \approx 179pix/mm$.

A noteworthy thing when using a calibration plate to measure the image resolution is the effect of the difference in the thicknesses of the calibration plate and the printed modules. When switching the target between a calibration plate and a module this difference creates a change in the distance from the camera to the image plane, thus, changing the focus of the image. Notice, however, that the change in the image resolution is not so significant as macro optics used with the camera makes the image on the camera sensor to appear very close to its natural size regardless of the distance. Further on, the developed correction system is not sensitive to errors in the image resolution. This makes the accurate determination of these errors unnecessary.

The rotation of the camera is trickier to measure than the image resolution. One method is to take multiple images from a static control point such that a small amount of translation to a given direction is applied between each image. The trajectory of the control point in the images forms a straight line. The direction of this line can be compared against the direction of the applied translation. The difference between these two directions gives the camera rotation.

After the camera has been calibrated, a convenient way to control the camera is needed. The images need to be transferred to the PC automatically and without handling the camera anymore. This can be done by using a *software development kit (SDK)*, which enables the integration of camera control mechanisms into the developer's own program code. In the case of the Canon EOS 5D camera, Canon's ED-SDK Version 2.2[1] was used.

As the correction system has been implemented almost entirely on top of MATLAB, which is a programming environment developed by The MathWorks, it was necessary to get the images into MATLAB workspace. The Canon SDK uses C++ as the programming language. Running programs, which have been written with C/C++ in MATLAB is possible by compiling the source code into mex-files (MATLAB executables). Using the mex-interface enabled us to take images and transfer them into MATLAB workspace by calling a single MATLAB function. As the camera generates preview and thumbnail images from every full size image it takes, it is also possible for the user to pass a parameter determining the quality of the image. The preview images are transferred and decoded faster than the full size images. This feature is handy, e.g., when aligning the modules under the camera for the first time.

[1] Available for registered developers in http://www.didp.canon-europa.com/

3.4 Detailed description of the correction algorithm

This section describes in detail the processing blocks seen in Figure 3.2 on page 29 (excluding the printing). Like stated in section 3.2, the algorithm is divided into three stages: feature detection (sections 3.4.1 and 3.4.2), feature matching (section 3.4.3), and print pattern correction (section 3.4.4). Some of the algorithm description is already published material [5; 6; 7].

3.4.1 IC extraction

Generally, by feature detection we mean the detection of the component connection points. The locations of the connection points need to measured in order to be able to correct the print pattern accordingly. In this thesis, only IC connection points are detected as these are the smallest ones and, thus, create missing contacts and short circuits even after a slight IC misalignment.

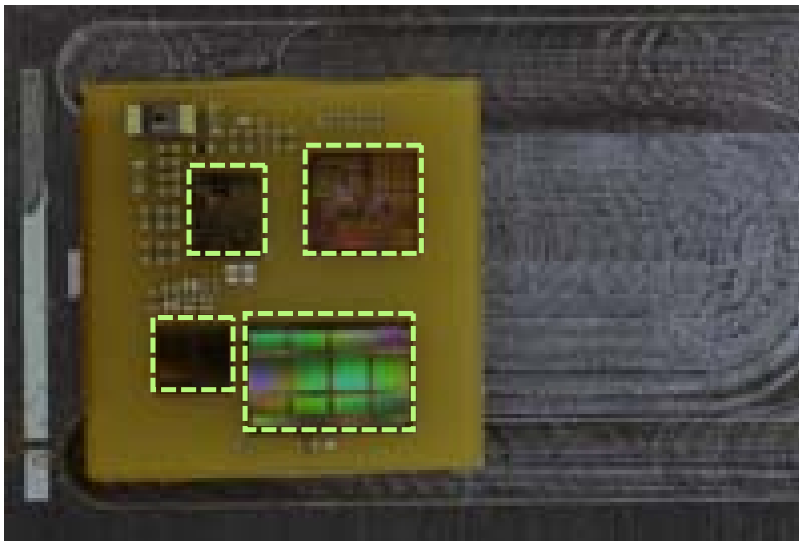
As each IC is uniquely misaligned, the subsequent processing steps need to consider each IC separately in order to determine the displacement errors of the connection points. When the detection of the ICs is done right at the first stage, the problem is divided into smaller pieces. Notice that there is nothing of interest outside each IC area. Extracting the ICs reduces the amount of data, speeds up the processing, and adds the possibility for concurrent computing.

Our initial approach in extracting the ICs was to use thresholding of the saturation component of the module image in HSV (hue, saturation, value) color space [6]. After this, morphological operations were used to segment the IC areas. This method has limitations as it assumes that the background material has a more saturated color than the ICs have. Obviously, whether this assumption holds or not depends on the appearance of the ICs and the molding material in use.

Another method to extract the ICs is to use template matching described in section 2.1. A module layout template is generated in the calibration phase of the correction system. The template is cropped from an image of a dedicated *calibration module* showing the layout of the ICs. Such a template is shown in Figure 3.4a. Figure 3.4b shows the result after matching the template with an image taken from another module. The IC borders can be determined from the corresponding ones in the template image after the template is positioned to the point with the maximum match.

Due to different IC misalignment characteristics between the calibration module and the module under processing, the extraction result can not be made accurate by using the IC template. However, this is not an issue as the exact IC borders need not to be known and the errors in the IC locations occur in significantly smaller scale compared to the module dimensions. The most important thing is that the connection points should remain inside the detected IC areas. This can be ensured by adding a small margin outside

(a)



(b)

Figure 3.4: A module layout template (a) is matched against the downsampled camera image in order to detect the ICs (b).

each detected IC border.

Although template matching can be done very efficiently as described in section 2.1, downsampling is done in order to further on speed up the processing. After all, the original resolution of imaging device is rather large so that the connection points on the sides of the ICs could be detected. Despite heavy downsampling, the module layout stays quite apparent as seen in Figure 3.4b. One advantage when using template matching together with downsampling to detect the ICs is that the method is easily extended into situations where there is no prior knowledge about the location of the module in a large image or into cases where there are multiple modules that need to be detected at once.

The IC extraction method has been tested to handle modules that have been rotated a couple of degrees. In general, sensitivity to image rotation can be regarded as a drawback when using template matching. If more freedom is desired to give for the operator in placing the modules on the printer table or the displacements of the components are extremely heavy, other methods has to be used.

3.4.2 Connection point detection

After the ICs have been extracted, each of them is individually processed in order to detect the connection points. The connection points usually lie on the borders of the ICs. Connection point detection is divided into two parts. First, a low-level segmentation of feasible connection point candidates is done. Second, the detected candidates are classified into objects and non-objects by using a neural network. The purpose of the first step is to bring down the amount of data in order for the classifier to be able to process it in reasonable time. In the following, the two stages of the connection point detection are considered in more detail.

Finding candidates

The low-level part of the connection point detection is based on the visual appearance of the connection points. The whole procedure is illustrated in Figure 3.5. First, the same template matching method that was used in extracting the ICs in section 3.4.1 is applied to the IC image, which is partly shown in Figure 3.5b. The difference to the earlier usage of template matching is that there are now multiple targets to detect. Figure 3.5a shows the template. Like the module layout template in the IC extraction phase, the connection point template can be collected from a dedicated calibration module. The donut-like appearance of the connection points is due to reflection from the camera and led ring above. This gives the connection points a rather unique look, which enhances the performance of the template matching. Notice that the template doesn't have to be shaped like a rectangle. Notice also that as the connection points are symmetric, template matching can be used regardless of the orientation of the ICs.

Matching the connection point template into the IC image produces a gray-level image seen in Figure 3.5c. The match rapidly deteriorates when moving away from the center of a connection point. This creates bright blobs where the connection points are. However, there are high match values also on areas having similar average color than the connection points have.

Common techniques to detect bright blobs like the connection points in Figure 3.5c are, e.g., *Laplacian of Gaussian (LoG)* [11, see p.582] and morphological *top-hat transformation* [11, see p.557]. However, the connection point areas are extracted by using the generalized logical level technique described in section 2.3. The advantage of the logical level technique is that it has the ability to do highlighting and thresholding of the blobs at the same time. Because the camera resolution is known, the stroke width parameter of the logical level thresholding can be adjusted so that the method detects objects having size equal to the known size of a connection point. The resulting image is shown in Figure 3.5d.

Coordinates of each detected connection point candidate are created from the thresh-

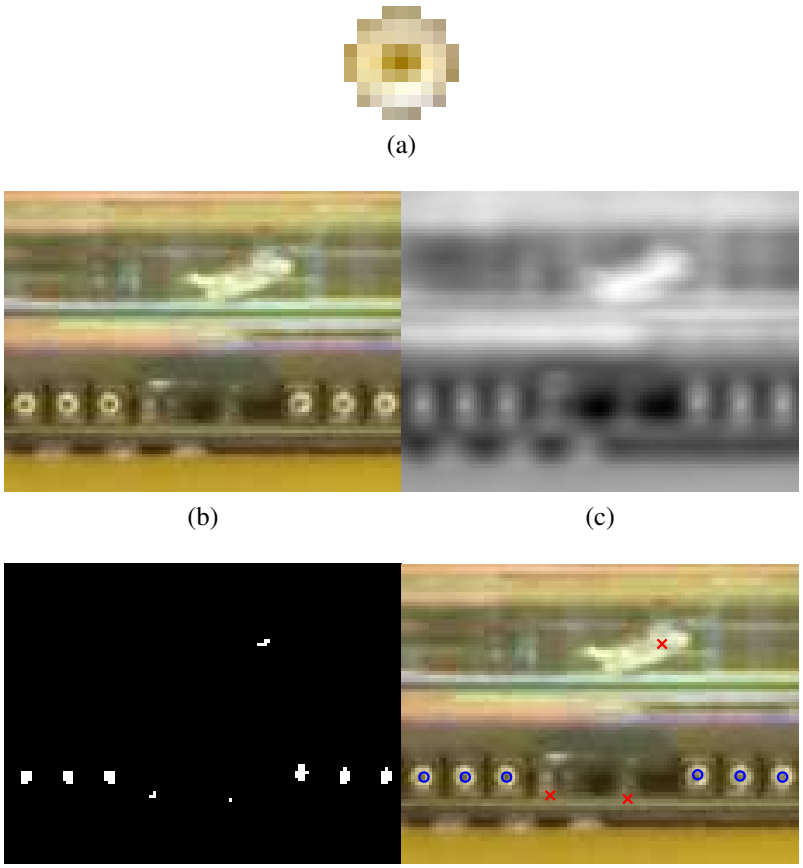


Figure 3.5: Detection of the connection points. A connection point template (a) is matched against the IC image (b) producing a gray-level match image (c). Individual blobs are detected by thresholding (d). Center point of each blob is then determined as the local maximum in the match image (c). Finally, each detected candidate is fed into a classifier, which classifies them to connection points (blue circles) or trash (red crosses) (e).

olded image (Figure 3.5d). The coordinates are not simply taken as the centroids[2] of the detected objects because this is not necessarily the center of the connection point. Instead, for the centroid of each object, a nearest local maximum is iteratively searched from the template match image (Figure 3.5c). If the image resolution is low with respect to the size of the connection points, interpolation can be done in order to find the location of the maximum point with better than one pixel precision. For resolution, which is somewhere around $180 pix/mm$ like in the example of Figure 3.3 on page 30, we use interpolation scale three. This gives the location of the detected connection point candidates with a precision of $\pm 1/(2 \times 3 \times 180 pix/mm) \approx \pm 1 um$ (whereas the diameter of the connection points are around $50 um$). As the maximum point doesn't have to be inside the detected object, it is possible for multiple objects to produce same coordinates. The duplicates are discarded in these cases.

[2] In geometry, object's *centroid* is the average of all the points belonging to the object. The centroid of a physical body with uniform density is equal to its center of mass.

In the first stage of the connection point detection, it is essential that preferably too many, instead of too few, candidates are produced so that the classification step can find the actual connection points amongst the candidates. Figure 3.5e shows the connection point candidates, which have already been classified. Blue circles and red crosses denote, which of the detected objects are classified as connection points and which are not, respectively. Next, we will consider the classification step.

Classification

After a feasible number of connection point candidates have been detected, a higher level method is used to classify the found objects into actual connection points and something else (trash). Neural networks (see section 2.4) are chosen to do the task as they offer a flexible solution to this kind of classification problem and their implementation is straightforward by using the Neural Network Toolbox in MATLAB. First, the construction of the NN is considered. This is assumed to be done offline by using a calibration module.

The NN used in the classification is a fully connected feedforward network with 243-25-1 structure. In other words, the NN has 243 inputs, one hidden layer with 25 neurons, and a single output neuron telling whether the input pattern is a connection point or not. The structure of the NN is illustrated in Figure 3.6.

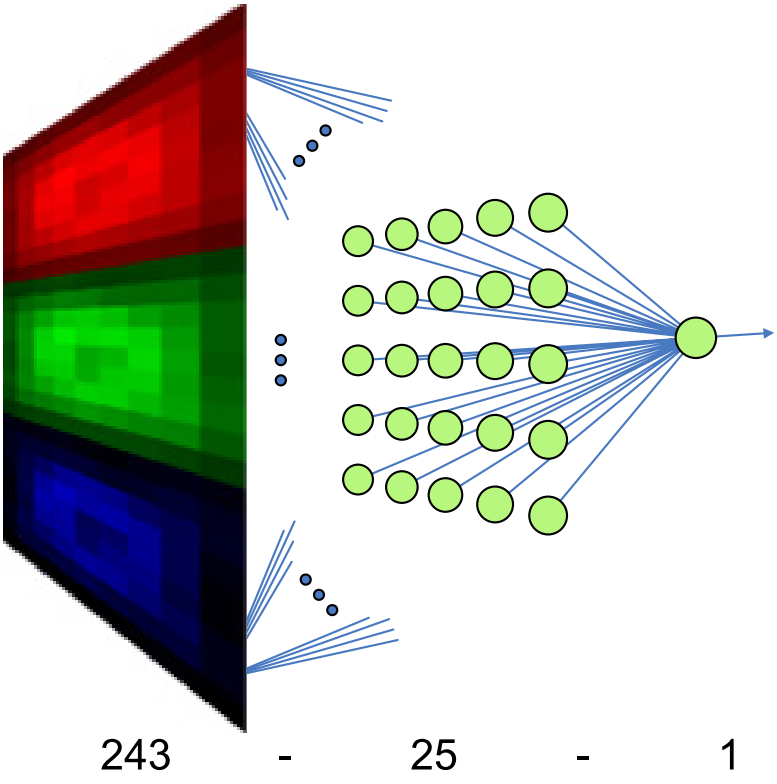


Figure 3.6: The NN used in the classification of the connection point candidates is a fully connected feedforward network with a 243-25-1 *structure.*

Size for the input layer is chosen such that the NN can take in size 9×9 RGB (red, green, blue) images of connection point candidates ($9 \times 9 \times 3 = 243$). As the size of the input layer is fixed, each classified image has to be resized. Image size 9×9 is chosen as it is rather small but can still represent the connection point with an adequate resolution.

A proper size for the hidden layer giving the best performance together with the ability for the NN to generalize was figured out by training the NN multiple times with different amount of neurons on the hidden layer and then measuring its performance. This is a common way to practice *network pruning* [36, see p.218]. The hidden layer in pattern recognition networks can be regarded as a *feature extraction layer*. The fact that the hidden layer only has 25 neurons – while the input layer is much bigger – is justified as the connection point candidate images are rather simple and have plenty of correlations between their pixel values (i.e., have low amount of distinctive features).

After resizing the input images, they are blurred and normalized before feeding them to the NN. Blurring suppresses the noise in the images, thus, giving the NN a better performance and capability to generalize. The blurring is done by convolving the images with a Gaussian convolution kernel. Different standard deviations for the Gaussian were tested starting from no blurring at all. The Gaussian resulting in the best classifier performance turned out to be the one with standard deviation of 0.4 pixels. Such a kernel is plotted in Figure 3.7. The blurring effect is not very drastic as the magnitude of the center weight is rather large, about 0.85. After the blurring, the input images are normalized by scaling the pixel values into the interval $[-1, 1]$.

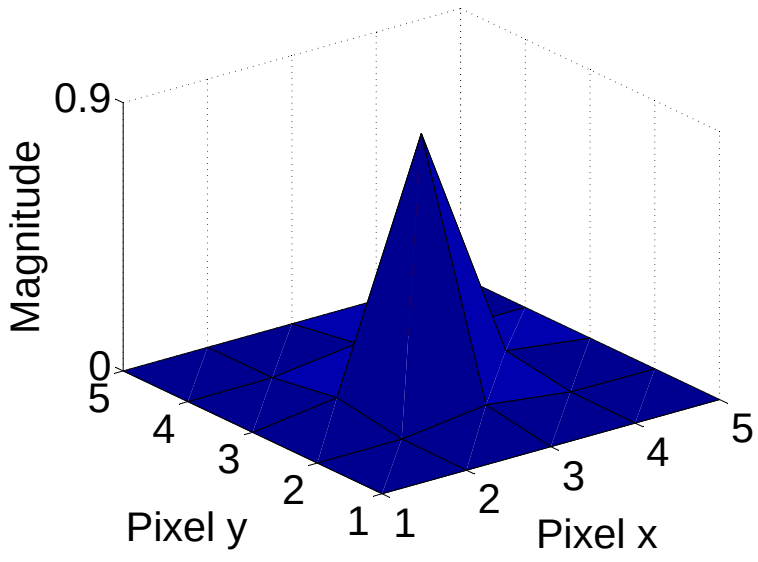


Figure 3.7: The convolution kernel used in the blurring of the NN input images.

Activation functions for the hidden neurons and the output neuron are chosen to be hyperbolic tangents. Hence, the output of the NN is on the interval $(-1, 1)$. Target values are set to be 0.9 in the case of a connection point and -0.9 in the case of trash. It is im-

portant not to set the target values as 1 and -1 as the hyperbolic tangent only attains these values in infinity and minus infinity, respectively. This can make the learning algorithm to saturate due to numerical inaccuracies.

A training method called *scaled conjugate gradient back-propagation* [40] is used to train the NN. Unlike steepest descent methods (like the one described by the delta rule in Equation 2.40 on page 24) conjugate gradient methods do not necessarily move towards the negative of the gradient but, instead, they find a route for much faster convergence. Whereas the standard steepest descent was found out to take about 50 minutes to converge with an optimal learning-rate, the scaled conjugate gradient method takes only about 30 seconds and also trains better performing networks. The Neural Network Toolbox of MATLAB was used in the training and using of the NNs as it includes implementations of various training algorithms along with data structures to store and use NNs.

The NN is designed such that it can be trained with connection point candidates found within a single designated calibration module. This way the calibration phase of the correction system doesn't get too complicated when introducing a new type of module. Typically, there are a couple of hundred connection points on a single module. In addition to these, the detection of the connection point candidates produces trash objects so that the total amount of training data for the NN is usually below one thousand samples. While there are 6126 free parameters in the NN (25×243 synaptic weights and 25 biases on the hidden layer and 25 synaptic weights and one bias on the output layer), there is a true possibility for overfitting the data.

The risk of overfitting is reduced by two tricks. First, the amount of training data is made four times larger by rotating each sample image by 0, 90, 180, and 270 degrees. This can be done as the connection points are rotationally symmetric. These particular angles are used in order not to change the size of the image. Using rotated versions of the same connection point also makes the input of the NN more invariant to the orientation of the connection points in case the rotational symmetry is lost for some reason. Second trick to avoid overfitting is to divide the data into training, validation, and testing data with a ratio of $60/20/20$, respectively. Like described in section 2.4.3, the validation data is used for early stopping of the training and the testing data for measuring the performance of the NN.

The appearance of the connection points slightly varies according to which IC they are from. This is taken into account when dividing the data into training, validation, and testing sets. A set is assigned an equal percentage of connection point and trash samples from the area of each IC. This ensures that the NN learns to classify connection points from every IC. Also the *a priori* knowledge of the ratio of the connection points and trash is incorporated in a proper way. The ordering of the samples inside each IC is otherwise randomized. This prevents the training algorithm from getting stuck on local optima.

Figure 3.8 shows an example of the NN training process. The training is continued

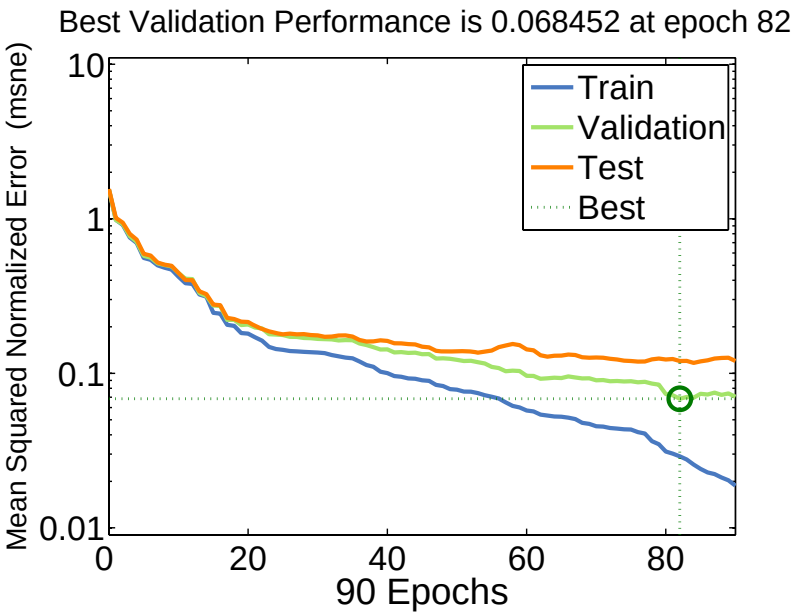


Figure 3.8: An example run of NN training with scaled conjugate gradient method. The training is stopped when the performance with the validation data (green line) stops improving. The error with the training data (blue line) still keeps decreasing due to overfitting. The final performance of the NN is measured by using the test data (orange line).

until eight consecutive epochs occur without improvement in the performance with the validation data. The network is then returned to the state with the best performance, in this example, to the state after the 82^{nd} epoch. The performance is measured by *mean squared normalized error*, which is the standard mean squared error after scaling the network output into interval $[-1, 1]$. This makes the performance comparable with NNs that have different target values (e.g., 0.1 and 0.9).

The NN trained during the calibration step is next used for classifying each of the connection point candidates found in the first stage of the connection point detection. A simple decision rule could be used that classifies each candidate with positive NN output to be connection points and each candidate with negative output to be trash. However, it is essential that there is as low amount of false positives (i.e., trash classified as connection points) as possible before proceeding into the next processing step, in which the detected connection points are associated with the designed connection points. Hence, the decision rule is adjusted by adding an extra requirement that limits the amount of connection point classifications in each IC to be at most a given proportion of the true amount of connection points in that IC. The true number of connection points is known by the design. If there are more candidates having a positive NN output, the ones with the smallest output are discarded. Hence, the remaining candidates are most likely to be connection points.

Figure 3.9 shows an example of the classification result when the number of connection point classifications is limited to be at most 80% of the actual number of connection points on each IC. Justifications for this particular percentage have been presented in our

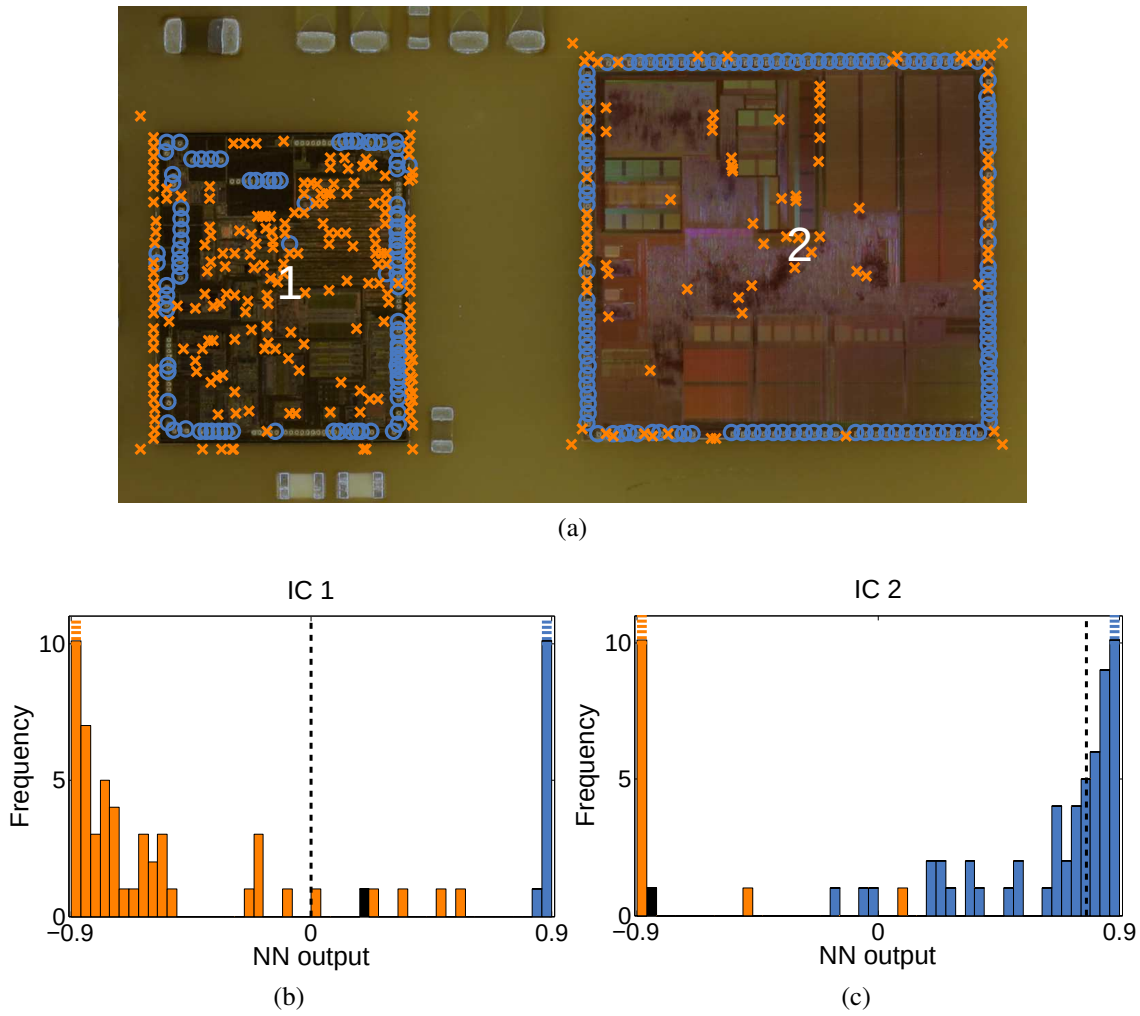


Figure 3.9: An example of the connection point classification. In Figure (a), blue circles denote for connection point classifications and orange crosses for trash classifications. Figures (b) and (c) show the histograms of the NN outputs for two different ICs. Blue and orange bars indicate outputs that are created from true connection points and trash, respectively. Overlapping parts of the orange and blue bars are indicated with black color. The highest bars have been truncated as they have approximately 80% *of all the hits. Black dashed lines indicate the decision boundaries.*

previous paper [6]. Classification results of two different ICs are illustrated in Figure 3.9a. Figures 3.9b and 3.9c show the histograms of the NN outputs together with the decision boundaries. The connection point candidates that produce network outputs above the decision boundary are considered as connection points and those that produce outputs below the boundary as trash.

In the case of the IC number one in Figure 3.9a, many of the connection points have been completely left undetected in the earlier stage. Hence, all the candidates producing positive output are classified as connection points. Like seen on the histogram in Figure 3.9b, This produces seven false positives (trash classified as connection points). All the positive classifications in the case of the IC number two are true positives like can

be seen on the histogram in Figure 3.9c. In this case, only the most obvious candidates are classified as connection points such that the amount of positive classifications doesn't exceed 80% of the true amount of connection points in that IC. There are plenty of false negatives (connection points classified as trash). However, this is not a problem like will be noticed in the next section.

3.4.3 Matching with design data

After the connection points from each of the ICs are detected, they have to be associated with the corresponding connection points in the design data in order to determine their displacements. As the ICs are rigid bodies, misalignments of the ICs can be modeled by a rigid transformation, which consists of rotation and translation like mentioned in section 2.2. Further on, as the connection points lay inside the ICs, a set of connection points belonging to a particular IC can be assumed to have transformed according to the same unknown rigid transformation. This transformation is also the transformation of the IC.

Conveniently, the connection point detection step already divided the detected connection points according to which IC they were detected from. However, we still need to determine, which detected ICs correspond to the ICs in the design. The module on the image is known to have approximately the same orientation as in the design and the misalignment of the IC is assumed not to be so severe that the ICs would change their place. This makes the problem rather trivial as simple ordering of the IC locations can be done in order to pair the detected ICs with the design ICs. A unique ordering in 2-d can be achieved by sorting the IC locations separately in x and y direction and then assigning an identifier for each IC. This results in a two-tuple of identifiers, which uniquely encode the ordering of the ICs on the module.

Next, the design connection points of a single IC need to be associated with the detected connection points of the corresponding IC on the module image. Section 2.2.1 introduced several PPM algorithms, which are potential for this job. However, an important constraining factor is that – like shown in Figure 3.9 – there now exists both extra and missing points. While not the most efficient algorithm, the improved version of the algorithm by Chang et al. [21] introduced in section 2.2.1 is chosen as it can handle these kinds of imperfections. In this case, as the ICs are rigid bodies and the approximate module orientation is known, we can constrict the algorithm to consider, e.g., scales from the interval $[0.9, 1.1]$ and rotations from the interval $[-\pi/2, \pi/2]$ only. This reduces the execution time significantly.

Another factor having a great impact on the execution of the PPM algorithm of Chang et al. is the amount of extra points in the point set being the smaller one of the two sets to be matched (the algorithm always matches the smaller set into the larger one). In this case, the smaller point set would be the detected connection points as in the connection

point detection step the maximum amount of detected connection points was fixed to be some percentage (at most 100%) of the design connection points. Extra points in the set of detected connection points equal to false positive classifications in the detection phase. This is why false positives are considered more harmful than false negatives.

After the PPM algorithm has figured out a pairing between the detected connection points and the designed connection points, the transformation of each IC is determined. First, however, the coordinates of the detected connection points need to be mapped into the design coordinate system. This mapping is called the *alignment transformation* and is done because the detected connection points have their origin on the lower left corner of the camera image, whereas the design has the origin on the lower left corner of the wiring bitmap image.

The alignment transformation is determined by considering the connection points of all the ICs at once and then finding the least squares transformation that transforms the detected connection points as close to the design as possible. This makes the uncorrected wiring scheme to align as well as possible with the distorted module, thus, minimizing the need for correction. If we used conventional fiducial marks to align the design always in a same manner, the correction would become much harder.

The alignment of the design is improved by determining the alignment transformation by using the weighted least squares solution given in Theorem 1 on page 16. This gives us an opportunity to give more weight to connection points, which are tightly pitched. These areas then become more accurately aligned, which further on eases the correction. Areas with sparsely located connection points have greater error. However, there is plenty of space to do the correction on those areas.

After the alignment transformation has been determined, the coordinates of the detected connection points are mapped into the design space. The detected connection

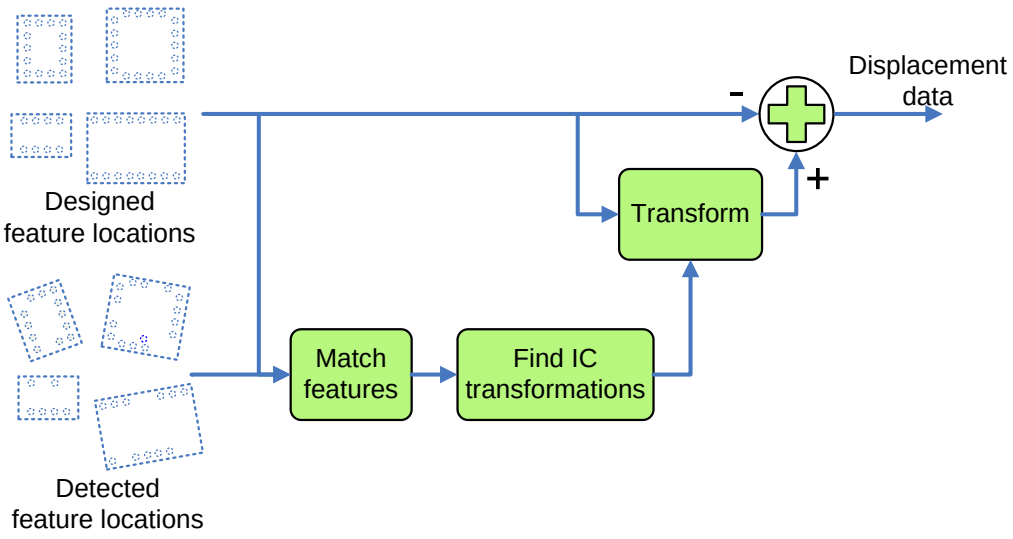


Figure 3.10: Diagram on how the displacement of each connection point is determined.

points that got paired with the design connection points can have slight errors in their locations due to inaccuracies in the detection phase. In addition, they usually do not include all the connection points, as some remain completely undetected. Hence, the actual displacement of the connection points cannot be calculated simply as the difference between the aligned detected connection points and the design connection points. Instead, the displacements are determined like illustrated in Figure 3.10.

First, for each IC, the transformation transforming the designed connection points of the current IC into the corresponding detected connection points is determined. This can be done, again, by using Theorem 1. Weighting is not needed here, so the weights are set equal. After this, the coordinates of the design connection points are mapped with this transformation. The displacement is now calculated as the difference between these transformed design connection point coordinates and the original ones. This results in displacement vectors, which tell how each of the design connection points should be moved in order for them to meet the connection points in the actual module.

3.4.4 Print pattern correction

The final and most important stage in the print pattern correction system is to create the actual printable wiring scheme. The patterns used are simple binary images telling the printer where to deposit a single droplet of ink. The plan is to modify the original wiring bitmap according to the determined displacements of the connection points.

Next, three different approaches for correcting the wiring bitmap are considered. *Redrawing last miles* and *displacement map* rely on directly manipulating the pixels of the wiring bitmap whereas *high level correction* vectorizes the design structures enabling more diverse techniques for correcting the errors.

Redrawing last miles

The first and most obvious correction method just erases some given amount of wiring from each of the wire ends, or *last miles*, and then redraws them to their correct locations. This method is described in one of our earlier publications [5].

Figure 3.11a shows three typical wire ends and their displacements. In Figure 3.11b, the wire ends have been erased by a given erasing radius and then redrawn to the right locations.

Advantages of the erase-and-redraw method include simplicity and therefore computational efficiency. However, the correction is suitable only for small displacements. When the errors are large, short circuits get easily generated. Also the choice of a proper erasing radius is crucial. The erasing radius has to be small enough not to propagate into the tortuous parts of the wiring. On the other hand, small erasing radii create unnecessary corners and might be unable to make the correction.

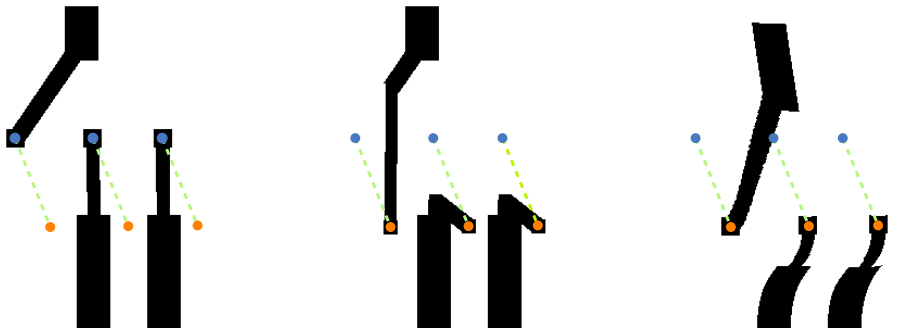


Figure 3.11: Correcting the original wiring (a) by redrawing the last miles (b) and by displacement map (c). Blue dots indicate the intended locations of the connection points while orange dots indicate their actual locations.

Displacement map

The idea of a displacement map is to translate each point in the wiring bitmap with a translation vector. The displacements of the connection points each define a single point in the displacement map. These points can be used to interpolate the displacement map such that its size equals to the size of the whole wiring bitmap. By doing the interpolation wisely, the displacement map can be made continuous and differentiable. Details of the construction of a proper displacement map is described in another of our publications [7].

Figure 3.11c shows the result after the three example wires in Figure 3.11a have been corrected by using a displacement map. The map is generated such that each wire end has an correction radius like in the redrawing of the last miles. The difference is that the correction radius of each wire end is now individual and depends on the amount of correction needed.

Displacement maps can be used to make the correction smooth. This enables correction of large displacements without creating sharp corners like in the simple erasing and redrawing. In addition, when a wire is drawn close to another wire, the displacement map makes the neighboring wire to move aside, thus, avoiding short circuits. Also the structure of the wires are approximately preserved whereas the redrawing of last miles always replaces the erased wire with a line having a fixed thickness.

Generating the displacement map and applying it to the wiring bitmap can be a computationally heavy task as the wiring bitmaps can have over 30 million pixels when printing with 9600 dpi, for example. Usually, however, only small amount of pixels around each connection point actually need to be re-deposited. This fact can easily be exploited in the implementation, which enables the online generation of print patterns.

High level correction

The first introduced correction method, the redrawing of the last miles, moves each wire end to its right location independently of the neighboring wiring. The second method, the displacement map, incorporates a global transformation that is constructed by using information from all the wire ends. In the third correction method, the interactions between the structures in the wiring bitmap are further on improved. The idea is to *vectorize* the wiring scheme, *manipulate* it accordingly, and finally *render* it back into a printable bitmap.

The Master's thesis of Rutanen [8] describes a method, which applies a high level correction for the wiring bitmap according to the given displacements of the connection points. After vectorization of the wiring bitmap the wire ends are moved in short steps towards the connection points until all wire ends meet their targets. The moving is made such that the wires stay intact and avoid overlapping with other wires as well as forming sharp corners.

By using a high level correction, extra knowledge about the design can be used to correct different kinds of structures with a proper way. Some objects, like antennas, are required to stay untouched while some are allowed to move more freely. Detection and correction of short circuits are easy to do as well as validation, which right away tells whether the correction succeeded or not.

The hard parts when making a high level correction would be the vectorization and the rendering as some information is always lost in these steps. The vectorization, however, can be made unnecessary by not using the wiring bitmap as it is. Instead, the original wiring layout of the designer can be used as it probably already exists in vector format.

4. EXPERIMENTS

This chapter performs the validation of the developed print pattern generation system. There are three experiments demonstrating and testing the system. In the first experiment in section 4.1, the print pattern correction is applied in case of a single module having severe component displacements. The second experiment in section 4.2 shows that the correction is easily scaled for different module types. It also introduces the concept of *panel printing* where multiple modules are printed at once by using the correction system to detect the relative locations of the individual modules. In the last experiment in section 4.3, ten modules are printed in a panel. The actual printing procedure is described in detail and the software is benchmarked for analyzing the performance of the correction process and to extract statistics about the errors of the module components.

4.1 Experiment 1

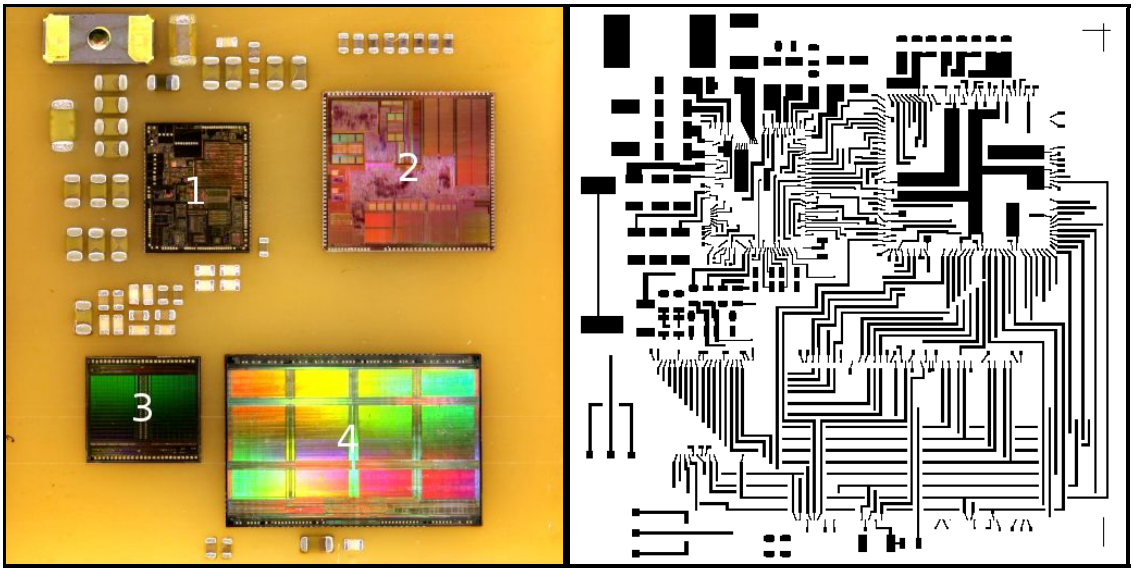
The first experiment demonstrates the correction of the wiring designed to be printed on top of a multi chip module (MCM) consisting of four different ICs and several passive components. The correction is done off-line and the printing result is simulated with and without the print pattern correction.

Figure 4.1a shows the layout of the used module. The size of the module is about $17mm \times 17mm$ and the diameter of the connection points on the borders of each IC is $44\mu m$. Scale this small makes it quite hard to print the wiring (shown in Figure 4.1b) as the diameter of a single droplet of ink is around $40\mu m$. The actual function of the test module is not relevant in this thesis[1].

In this experiment, the print pattern correction system is run in off-line mode. The module images have been taken by using a conventional desktop scanner instead of using a setup like described in section 3.1. The system has been calibrated by using a calibration module. This is the module used to collect the IC layout template for IC extraction as well as to teach the classifier and to create the connection point template for detection of the connection points. The calibration module is different than the module used for testing the correction system.

Various figures are now calculated in order to measure the success rate of the correction system in different stages of its operation. The ultimate measure of the performance of

[1] To please any curious mind, let us reveal that the module was originally used in an ancient mobile device



(a) (b)

Figure 4.1: Figure (a) shows the type of module used in the experiments. In Figure (b) there is the wiring image, which is going to be printed on the module.

the print pattern correction system is whether the correction succeeds or not. There are no intermediate forms as the resulting module is either functioning or is not (assuming that the printing goes successfully). However, we would also like to have information about the sensitivity of the correction algorithm and would like to know, which part of the processing chain is critical when compared to the other parts.

The first part of the correction is the IC extraction explained in section 3.4.1. The step is crucial for the success of the correction because if the ICs are detected from totally false locations, the correction inevitably fails. As the forthcoming processing step detects the locations of the IC connection points, a relevant measure of the success rate of the IC extraction is to count how many of the IC connection points remain inside the detected IC area. In the test case, the IC extraction succeeded perfectly.

The next step is to detect the connection points like explained in section 3.4.2. The results for the performance of the connection point detection are listed in table 4.1. For each IC, the table lists the number of connection point classifications that went right (true positives), the number of non-connection point classifications that went right (true negatives), the number of connection point classifications that went wrong (false positives), and the number of non-connection point classifications that went wrong (false negatives). There is also the number of connection points that was completely missed by the candidate searching step. The enumeration of the ICs is the one seen in Figure 4.1a.

After detecting the connection points, the task is to associate them with the ones in the design data. This is done by using a PPM algorithm like explained in section 3.4.3. For each IC, the algorithm either succeeds in determining the correct pairing or fails in it. In

Table 4.1: Connection point detection rate.

	True Pos.	True Neg.	False Pos.	False Neg.	Undetected
IC1	101	196	1	25	2
IC2	116	70	8	16	23
IC3	37	124	3	13	0
IC4	55	212	6	13	8
Tot.	284	616	43	53	72

the experiment case, the pairing was correctly determined in case of every IC.

When the correspondence between the designed connection points (wire ends) and the actual connection points on the module is known, the print pattern is rotated and translated with an alignment transformation such that the error between the module connection points and the wire ends is minimized in least squares sense like described on page 42. After this, the print pattern is corrected like explained in section 3.4.4.

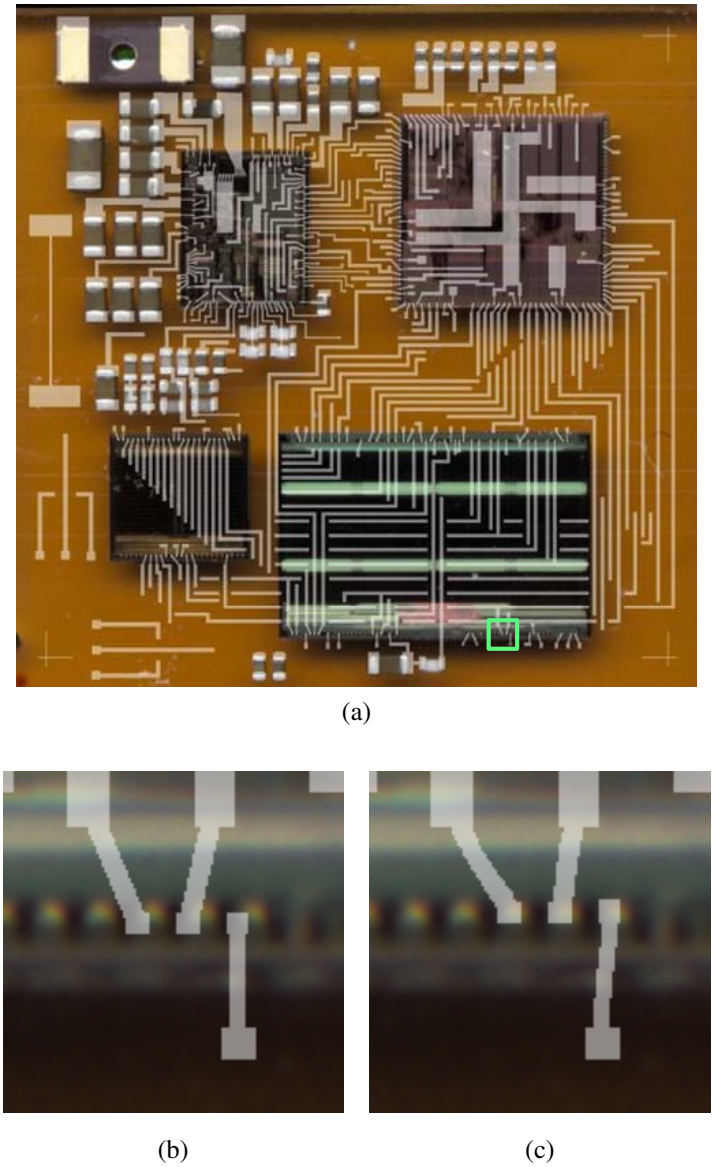
The displacement map approach was used as the correction method in this experiment. The errors in the locations of the module components are rather small with respect to the amount of displacement that the correction could manage. Thus, the corrected print pattern is usable, i.e., there are no short circuits and all the wire ends meet their intended locations.

Table 4.2 shows the displacement transformations of each IC. There are also the average and maximum displacements of the connection points for each IC. The transformation parameters are given with respect to the optimally aligned print pattern such that for each IC the origin (the point that the rotation is applied with respect to, that is) lies on the mid-range (mean of the minimum and maximum coordinate values) of the connection points on that IC. Counterclockwise is considered to be the positive direction of rotation.

Table 4.2: Displacements of each IC and their connection points.

	Rotation (degrees)	Translation X (μm)	Translation Y (μm)	Avg. Error (μm)	Max. Error (μm)
IC1	0.21	-3	6	9	15
IC2	0.15	9	8	13	20
IC3	-0.17	-3	-11	12	17
IC4	-0.22	9	-18	22	35

The printing result is simulated by drawing the optimally aligned wiring on top of the experiment module. The result is shown in Figure 4.2. Figure 4.2a shows the whole module. The wiring has been drawn on top of the module image with white transparent color. A close-up is shown from the area indicated by the green square. Figure 4.2b shows the area without correction and Figure 4.2c shows the area with correction.



(a)

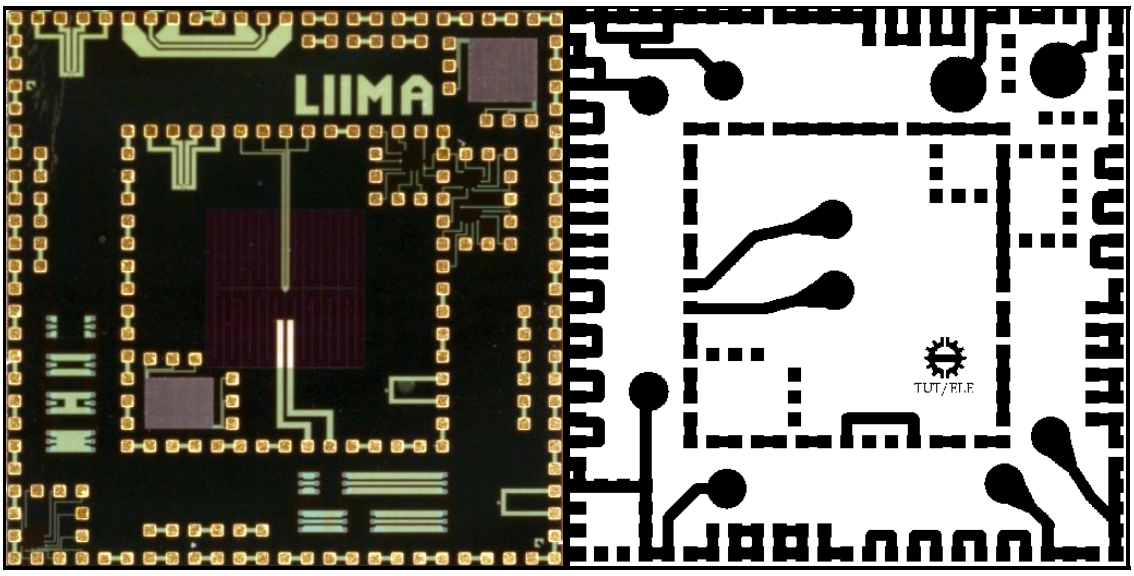
(b) (c)

Figure 4.2: The simulated printing result of the first experiment.

4.2 Experiment 2

This far only a particular type of MCM has been used. The second experiment shows that with minor effort it is possible to change the type of the module that the correction is applied to. In this experiment, actual printing is conducted together with the verification of the electric functionality of the printed modules.

The type of the module used in the second experiment and the corresponding wiring image are shown in Figures 4.3a and 4.3b, respectively. Unlike on the previously used MCM, this module has only a single IC. The connection points differ from the ones seen earlier by their size, shape, and color. They are also located all around the IC, not only on the edges. The diameter of the connection points is about 220μm, which makes them a lot larger than the ones in the MCM.



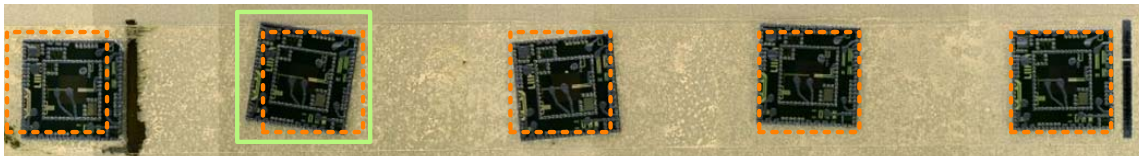
(a) (b)

Figure 4.3: Figure (a) shows the type of module used in the second experiment. In Figure (b) there is the wiring image, which is going to be printed on the module.

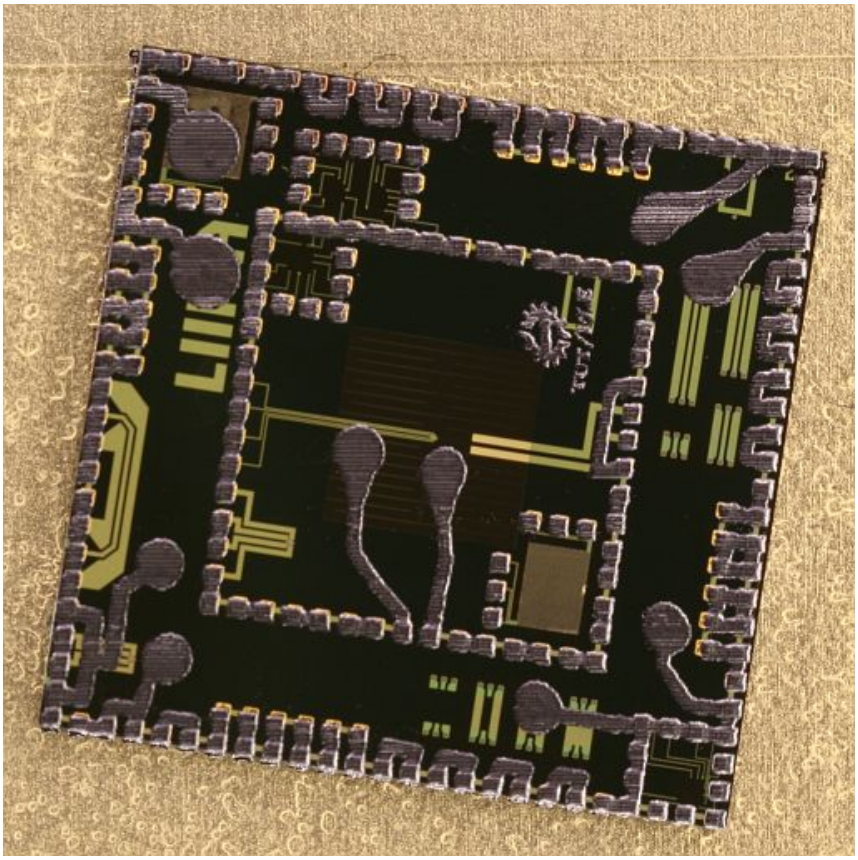
As the test module consists of only one component, there occurs no component drifting that would make the print pattern not to match the module layout. In this experiment, component misalignment is artificially created by not aligning the print pattern optimally on the module like in the previous experiment but by using a fixed point on the background on the printer table to align the print pattern. After the alignment, the displacement map based correction is run on the pattern to make the printed wiring to match the connection points of the module. Notice that it is not possible to use the redraw-last-miles method to correct the wiring as there are no last miles in this design.

In this experiment, five print patterns are corrected and printed on five modules in an online environment like described in section 3.1. The modules have been placed on the printer table such that they are approximately located on the point where the print pattern is going to be aligned. Like mentioned in section 1.1, one of the advantages of inkjet printed electronics as an electronics manufacturing method is the scalability of the printable area. In order to take better advantage of the available printing area that the printer can cover on a single run and to lower the amount of the operator's manual labor, which is needed to set the printer into the printing condition before each run, the five modules are printed at the same time as a panel.

Now that there are multiple modules to print, the print pattern correction is run separately for each module. First, the modules are individually photographed by shifting the printer table below the camera. Then, as the offset between each image is known and the locations of the modules in the images have been detected, the resulting corrected print patterns are put together as a single large pattern.



(a)



(b)

Figure 4.4: Printing result of the second experiment. Figure (a) shows the whole panel. Orange squares show where the modules are expected to be and where the print pattern is initially aligned. The module inside the green square can be seen in detail in Figure (b).

The experiment printing was conducted at the Department of Electronics at TUT on November 4, 2008. Figure 4.4 shows the printing result. In Figure 4.4a, multiple photographs have been catenated as a panorama image showing all the printed modules. The orange squares indicate the places where the print patterns have been aligned to. The error created by the intentionally careless module placement is corrected by detecting the true connection point locations and by using the displacement map correction like mentioned.

The area surrounded by the green square in Figure 4.4a is shown in larger scale in Figure 4.4b. All the printed modules were verified to work.

4.3 Experiment 3

In the third experiment, wiring is printed on 10 modules. The same type of MCM is used as in the first experiment in section 4.1. Like in the second experiment in section 4.2, the modules are printed in a panel. This experiment focuses on describing the actual printing session (section 4.3.1) as well as on software benchmarking (section 4.3.2).

4.3.1 Printing

The printing of the third experiment was conducted at the Department of Electronics at TUT on November 13, 2009. Flow graph of the planned printing session is illustrated in Figure 4.5. The graph is designed such that three human operators are able to work in parallel. The first operator handles software related issues, the second one takes care of the imaging equipment, and the third operator operates the printer. The printing session followed the idea of the flow graph of Figure 4.5 with the exception that only two operators were available instead of the originally planned three. Full session diary can be found enclosed in Appendix B.

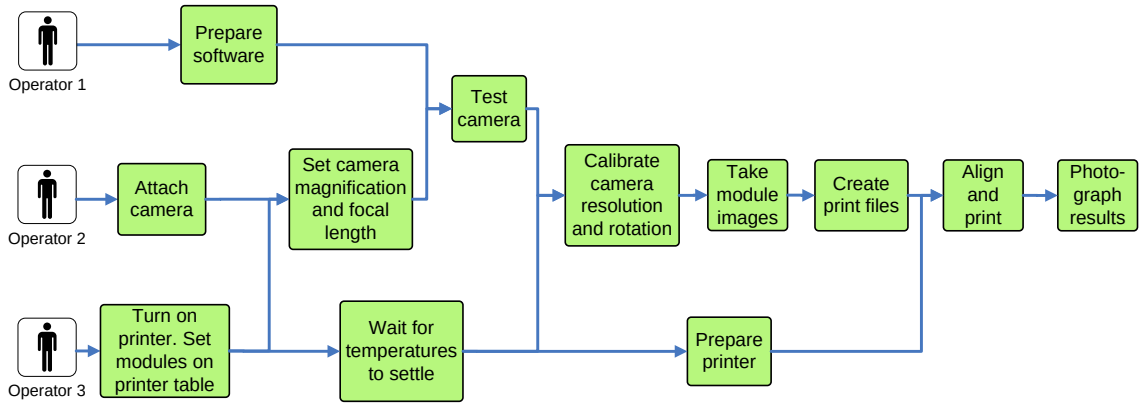
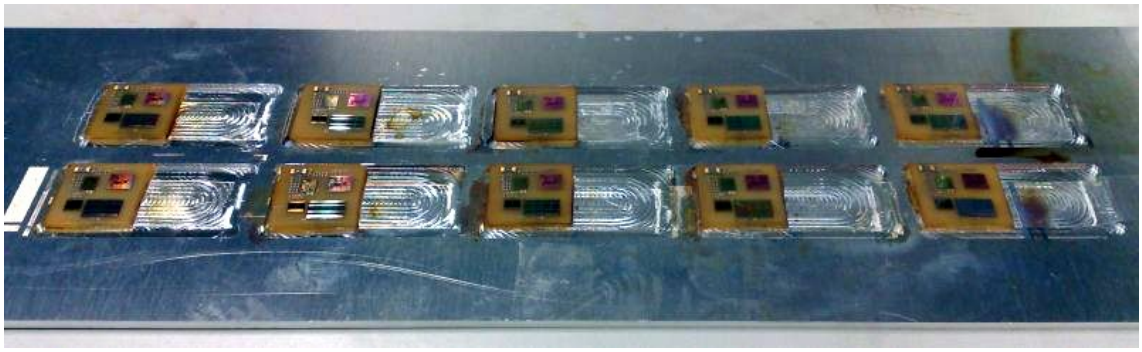


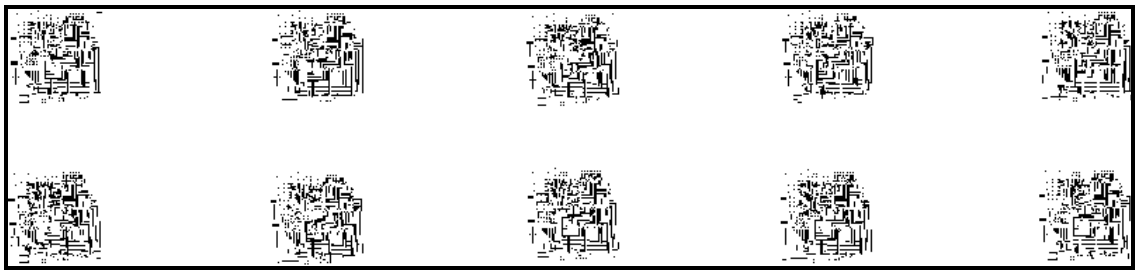
Figure 4.5: Flow graph of the printing event designed for three individual operators.

The printing in a panel was done by using a jig (aluminum plate with slots in it) seen in Figure 4.6a. The jig consists of 10 slots in 2×5 formation. The slots are used for positioning the modules. There is an example print pattern shown in Figure 4.6b, which has been created by individually running the correction system on each module and then catenating the resulting print patterns like described in the previous experiment.

In the actual printing session, the first thing that needs a close attention is the possible changing of the environmental conditions between the taking of the module images and the printing. The most obvious issue of this type is the warming up of the printer, which causes thermal expansion on the printer table and, thus, on the modules and the jig that lay on the printer table. The consequences can be fatal as alignment errors of couple of tens of microns are adequate to make the corrected wiring not to match the displaced modules



(a)



(b)

Figure 4.6: Figure (a) shows a 2×5 panel of modules. In Figure (b) there is the wiring that is going to be printed on top.

anymore. When the printing is done in a panel, the errors are cumulative. This makes thermal expansion even more problematic issue when compared to the situation where only one module is printed.

To get a hold of the amount of error caused by thermal expansion, consider the aluminum jig holding the modules. For aluminum, the magnitude of thermal expansion (in $25°C$) is $23.1\mu m/(m \cdot K)$. The distance between the left edge of the left-most slot and the right edge of the right-most slot on the jig is about $210mm$. Hence, as the temperature of the jig rises from the room temperature ($25°C$) to the operating temperature of the printer table ($60°C$ in this case), the misalignment can be as large as $23.1\mu m/(m \cdot K) \cdot 210 \cdot 10^{-3} m \cdot (60-25)K \approx 170\mu m$. Recall that the diameter of a connection point on the IC is $44\mu m$. Hence, in order for the printing of the panel to succeed, it is vital to wait for the temperature to settle before photographing the modules and creating the print pattern.

At the beginning of the printing session the printer had been in use earlier that day and the operating temperature was already reached. The operator responsible for handling the printer firmly attached the 10 experiment modules on the jig and placed the jig on the printer table.

During the initial tasks of the printer operator the second operator set up the camera and the laptop that were used for taking the module images and for running the print pattern correction software. After the modules were set on the printer table, camera magnification

and focal length were tuned such that the image was sharp. It was then tested that there were no problems in sending the shoot command to the camera and transferring the image to the laptop after taking it. The test image was confirmed to be fine.

After the imaging equipment was all set and it was verified (by using an infrared thermometer) that no thermal expansion was occurring anymore, the camera was calibrated. This was done by using a calibration plate like explained in section 3.3. After the calibration, images of the modules were taken one by one. First, the upper module row was photographed starting from the left-most module. Then, the lower row was photographed starting from the right-most module. This particular order was chosen as it minimizes the distance that it is necessary to move the printer table. Steps that were taken between each image were 43*mm* in processing direction and 26.5*mm* in cross-processing direction (horizontal and vertical direction in Figure 4.6, respectively). The step size is determined from the jig slot spacing.

After taking the module images, the images were fed into the correction software. The template used in the IC extraction (see section 3.4.1) and the connection point template and the classifier used in the connection point detection (see section 3.4.2) had already been created beforehand. This was done by using the same module that in the experiments is on the lower left corner of the panel.

After the corrected print image was ready, the image was aligned with the modules and then printed. The alignment was made by first printing an alignment image same size with the actual print image and then measuring the offset between an alignment point in the alignment pattern printed aside the panel and the corresponding point on the panel.

Due to a bug in creating the print pattern, the first printing had to be canceled and then restarted after quick regeneration of the print image. The second attempt was a success after which the modules were photographed for inspecting the results. The photographing was done in the same manner as before the printing.

4.3.2 Results

In this section, the success rate of the correction system is measured like in the first experiment. Also images of the printing result are presented.

Benchmarking of the correction system starts from the IC extraction. For all ICs on each module the extraction results are perfect, i.e., all the connection points remain inside the detected IC borders.

The results of the connection point detection are listed in table C.1 in Appendix C. The enumeration of the modules goes from left to right, starting from the upper row of modules, when referred to the panel of modules shown in Figure 4.6. The enumeration of the ICs is the one shown in Figure 4.1a.

The binary results of the success rate of the PPM algorithm are listed in table 4.3. A green number one indicates that the correct pairing was found and a red zero indicates

that the correct pairing was not found. A module is considered to be successful if the PPM algorithm has a successful result for each IC on it.

Table 4.3: Success of the PPM algorithm.

	Module									
	1	2	3	4	5	6	7	8	9	10
IC1	1	1	1	1	1	1	1	1	1	1
IC2	1	1	1	1	1	1	1	1	1	1
IC3	1	1	1	1	0	1	1	1	1	1
IC4	1	1	1	1	1	1	1	1	1	1
Tot.	1	1	1	1	0	1	1	1	1	1

Table 4.4 shows the parameters of the alignment transformations of each module. The translation components of the alignment transformations are normalized such that the translation of the upper left module is zero. In case of the other modules, the origins equal to the origin of the upper left corner module such that it has been translated according to the spacing of the jig slots.

Table 4.4: Module alignment transformations.

Module	Translation X (μm)	Translation Y (μm)	Rotation (degrees)
1	0	0	-0.32
2	-214	493	-0.08
3	147	588	0.79
4	8	1043	0.49
5	-273	1519	-0.26
6	-387	1271	-0.37
7	259	780	0.32
8	21	1042	-0.25
9	81	252	-0.51
10	490	-48	0.76

Table C.2 in Appendix C shows the displacement transformations of each IC on every module and the average and maximum displacements of the connection points on each IC. The transformation parameters are given with respect to the optimally aligned print pattern such that the origin of the transformation is on the mid-range of the connection points of the corresponding IC.

The printing result of the third experiment is shown in Figure C.1 in Appendix C. Figure C.1a shows an overview of the whole module panel. The module inside the green square is enlarged in Figure C.1b.

5. DISCUSSION OF THE RESULTS

The purpose of the first experiment in section 4.1 was to demonstrate a case where the modifying of the wiring is necessary in order to get successful printing results. Even after the optimal alignment of the print pattern, the errors between the IC connection points and the wire ends remain at tens of microns like seen in table 4.2. As the diameter of a single connection point is $44\mu m$ and the most severe connection point displacement is $35\mu m$, a bad connection is likely to occur, which makes the module non-functional. This can be verified from the simulated printing result shown in Figure 4.2.

No major conclusions can be drawn about the general performance of the print pattern generation system based on only one test run. However, there are some interesting points that can be noted from the connection point detection results in table 4.1. First, the number of undetected connection points of IC number two seems to be a lot greater than with the other ICs. The results of the connection point detection for IC number two are illustrated in Figure 5.1.

Like can be seen from the image, there are a lot of undetected connection points on the top edge of the IC. The ones that got detected are classified as trash. This is caused by the desktop scanner used to take the module image. The sweep of the scanner sensor with the light creates a reflection on one side of each IC. On this IC, the connection points are so close to the IC edge that the reflection makes the connection points to blend with the edge and the first stage of the detection fails to detect the connection points. The few connection points that get detected are classified as trash because the trained NN obviously has not learned to detect connection points with this kind of appearance.

Because of a different kind of illumination setup, in an online-case there are no problems like the one seen in Figure 5.1. The led ring and milk glass around the camera objective create a uniform illumination from every direction without generating reflections or shadows. The problem created from using the desktop scanner could be solved, e.g., by changing the logical level thresholding conditions introduced in section 2.3. Notice, however, that even though the detection rate of the connection points is poor, the PPM algorithm correctly manages to find the correspondence between the detected and designed connection points.

In the second experiment in section 4.2, it was shown that the correction system can operate regardless of the module type. Even after the layout of the module and the scale and appearance of the connection points is totally different from the ones seen earlier,

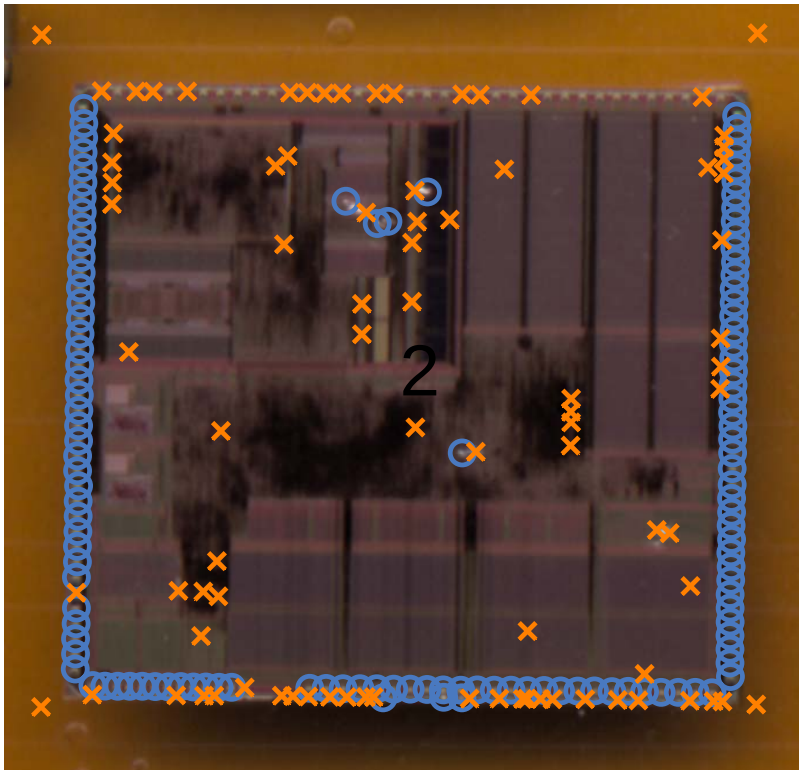


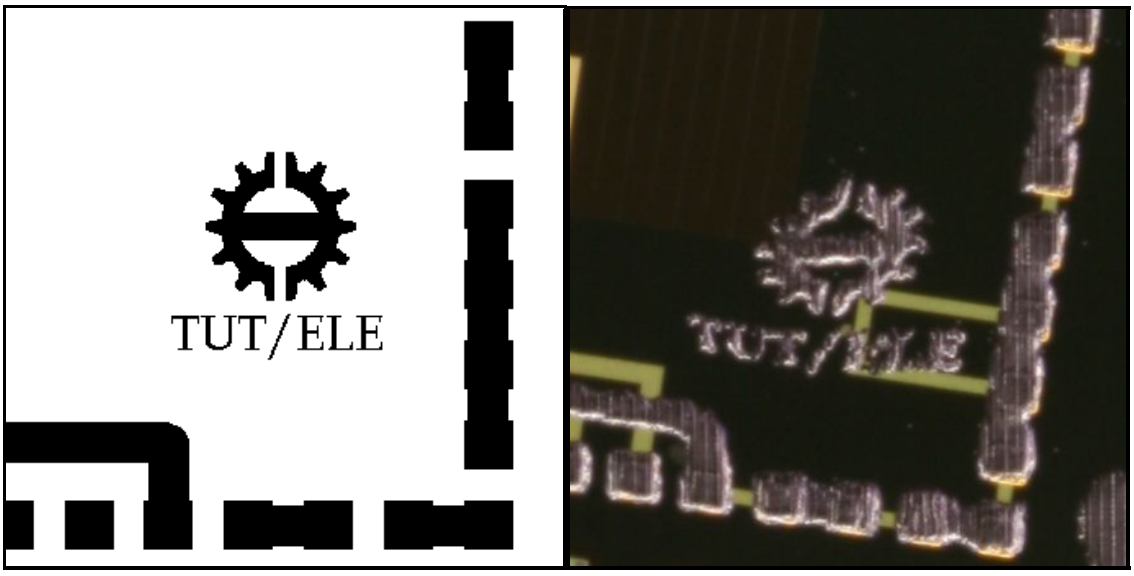
Figure 5.1: Classification results of the connection point detection in IC number two. Blue circles indicate the positive classifications and orange crosses indicate the negative classifications.

there occurs no problems in the processing chain.

In addition to the connection points, also the wiring scheme in the second experiment is different in the way that there are no actual wire ends or last miles. The displacement map type of correction can still be used as it is independent of the structure of the design. To test the correction, a relatively large displacement was intentionally created for each module. The printing result in Figure 4.4b shows that with the displacement map even large errors can be corrected. Notice, how the secondary features have been moved along with the connection point centers, which are the points that define the displacement map. This is illustrated in Figure 5.2, which shows a part from the original print pattern (Figure 5.2a) with a TUT logo and the corresponding part after printing it (Figure 5.2b).

Slight mismatch can be seen on couple of the connection points in the result of the second experiment. This is due to the small differences that can be initially found between the module (Figure 4.3a) and the design (Figure 4.3b) and are not caused by the correction algorithm.

In the final experiment in section 4.3, the correction system was proved to work as a part of an online process. Due to lack of routine the session lasted several hours. However, without the correction system, it would have been impossible to print the ten modules at once and the printing would have taken days instead of hours.



(a) (b)

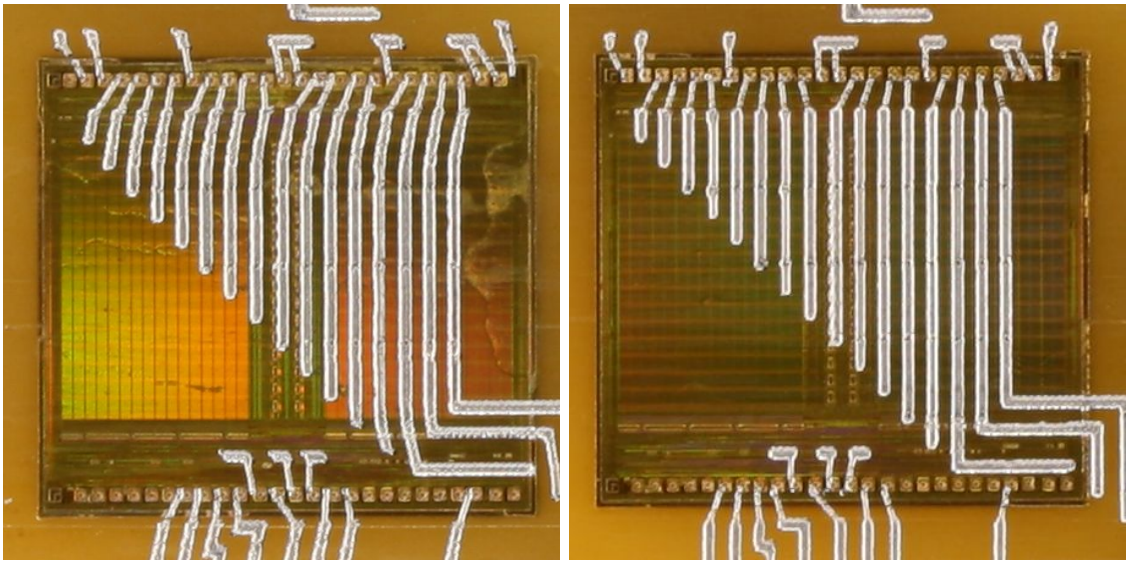
Figure 5.2: Figure (a) shows a small part of the original print pattern. In Figure (b) there is the same area after correction and printing.

The first thing to be noted from the results of the third experiment is the failing of the correction of the third IC on the fifth module. Figure 5.3 shows the printing result of the IC in question both in the unsuccessful and in a successful case. The failure is caused by the PPM algorithm, which has paired each wire end with a wrong connection point lying next to the correct one.

The mistake that the PPM algorithm has committed is not surprising in a case where the point pattern to be matched has the type of symmetry that there is in the third IC. Notice that when there are 50 points equidistantly located in two rows like the connection points in the IC number three, the PPM algorithm can achieve an error free pairing of 48 points by pairing each point with the neighboring point with respect to the correct one. This is 96% of the optimal result, which can be considered a very good match. If the PPM algorithm finds a pairing this good before finding the 100% match, the early stopping mechanism might stop the execution and the correct pairing remains to be unattained.

Although poor results for the PPM algorithm could be expected in the case of all the other modules as well, the pairing of the connection points of the third IC fails only in the fifth module. This is because after the connection point detection step, the coordinates of the detected connection points are nearly in the same order in the computer memory as the coordinates of the corresponding connection points extracted from the design data. This leads to the situation where the PPM algorithm in most cases finds the correct pairing before the wrong one.

In the case of the fifth module, the failing of the PPM algorithm can be traced all the way back to the connection point detection step. The detection results for the problematic



(a) (b)

Figure 5.3: Figure (a) shows the printing result of IC number three of the fifth module, where the wire ends have been redrawn on incorrect connection points. Figure (b) shows the corresponding correct result of module number one.

third IC are illustrated in Figure 5.4, which shows a portion from the upper left corner of the IC. It turns out that the connection points of IC number three are almost square-shaped whereas the ones in the other ICs are round and also a bit smaller. The difference in the appearance is due to the frame around the connection points while, regardless of the IC, the actual contact areas have the shape of a disc with a diameter of $44\mu m$.

The difference in the appearance of the connection points is not that big a problem for the NN doing the classification although from table C.1 in Appendix C it can be seen that the detection rate is consistently a bit poorer in the third IC than it is on IC number two and four (IC number one is discussed soon). The NN still manages reasonably well because in the training step, the classifier has been trained to recognize connection points from every IC like described on page 38. However, before the classification step, in finding the connection point candidates, the round shaped connection point is used as the template in the template matching. This results in the maximum match being slightly displaced in case of some of the connection points, which can also be seen in Figure 5.4.

The described errors in the connection point detection boil down to the failing of the PPM algorithm. This happens because the maximum error in the locations of the detected connection points (see section 2.2.1) is assumed to be smaller ($5\mu m$ in this case) than it actually is. This prevents the PPM algorithm to find a pair for all the detected connection points and ultimately leads to a wrong pairing. The problem could be solved by allowing a greater error. However, a more sustainable solution would be to tune the connection point detection such that the detection result of the connection points is more accurate.

Figure 5.4: Connection point detection results of the upper left corner of the third IC on module number five. Blue circles indicate the positive classifications and orange crosses indicate the negative classifications.

This could be done, e.g., by using a different connection point template for each IC.

Besides the unsuccessfully corrected module number five, another eye catching result is the poor rate of connection point detection in the IC number one. For each module, around 40% of the 128 connection points on the first IC remain completely undetected. As only 60% of the connection points are detected, there is a possibility of about 25 (20% of the 128) false positives to occur in the classification step before the maximum amount of allowed positive classifications (which is 80% of the total amount of connection points like described on page 39) is achieved. In addition, there are plenty of trash objects on the first IC that resemble the connection points. These objects easily turn out as false positives in the classification step.

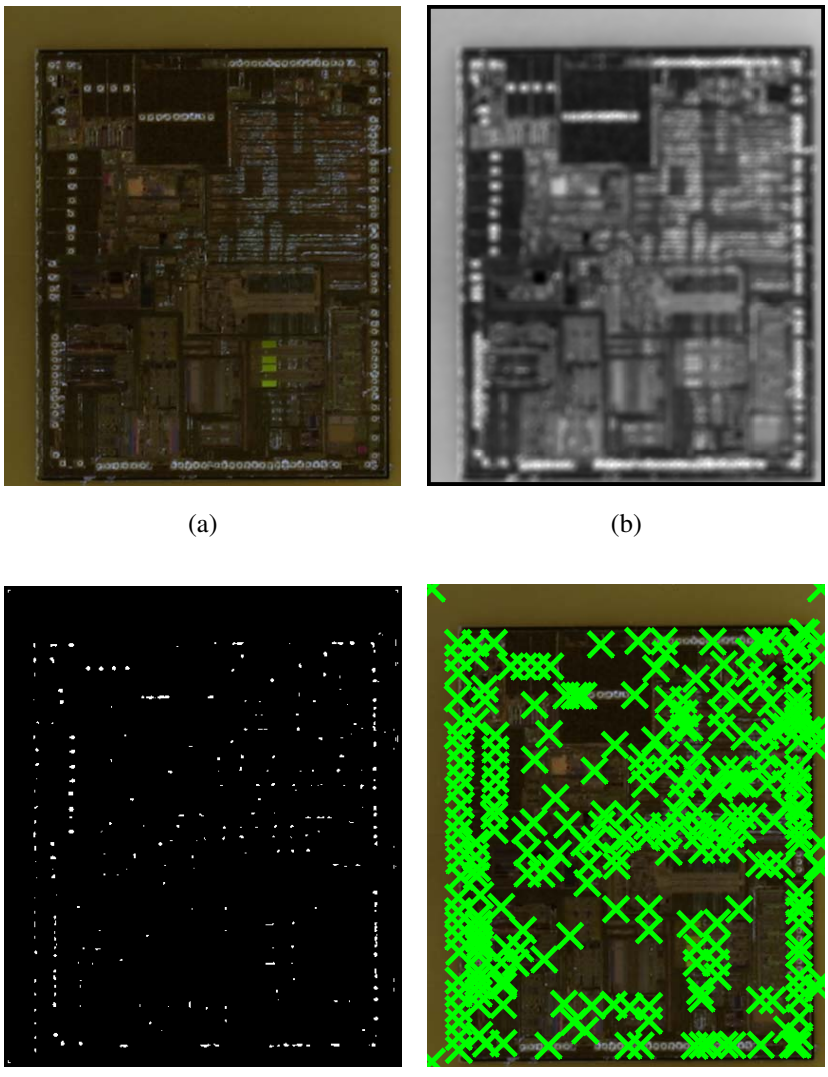
The result of the connection point candidate finding step is illustrated in Figure 5.5. A high number of connection points remain undetected on areas where the points are densely packed. After template matching (Figure 5.5b) the connection points get bulged and finally get thrown away in the thresholding step (Figure 5.5c). The reason for this is the combination of tight spacing of the connection points and the fact that there are remains of old ink from earlier printing tests on the connection points. Traces of the old ink can also been seen in the middle of the IC. Plenty of this ink get detected as connection point candidates as the color of the ink is similar to the color of the connection points.

The problem of the undetected connection points on IC number one could be prevented by using clean modules. This is easy to verify by looking the result of the connection point detection in the first experiment (table 4.1). There are only two connection points undetected on the first IC. Also the parameters of the logical level thresholding could be tuned in order to have better thresholding results.

Like in the first experiment, regardless of the poor detection rate of the connection

points, the PPM algorithm manages to find out the correct pairing between the detected connection points and the designed ones. This makes the correction system robust to these kinds of faults. Problems may arise in case of ICs with particular kind of symmetry in the locations of their connection points like happened in the third IC of module number five.

Another kind of robustness can be noted by comparing the results of connection point detection in case of the calibration module (module number six) with the corresponding results of the other modules. Although the calibration module is the one used to collect the connection point template as well as to train the NN used in the detection of the connection points, the detection results of the calibration module are in fact slightly worse than on



(a) (b)

(c) (d)

Figure 5.5: Finding connection point candidates in IC number one. Figure (a) shows the original IC image, in Figure (b) the image has been matched with a connection point template, and in Figure (c) there is the result after logical level thresholding and shape filtering. Figure (d) shows the detected connection point candidates.

the average. This suggests that – besides the calibration module – the correction system generalizes to work on other modules as well.

Now leaving behind the analysis of the performance of the correction system and moving onto the interpretation of the IC displacement statistics on table C.2 in Appendix C. As can be seen from the table, the displacement errors of the ICs are relatively small. In fact, as the maximum connection point displacement is only $14\mu m$ and the diameter of a connection point is $44\mu m$, one could claim that no correction would have needed for any of the modules. Indeed, the ten modules used in the third experiment are fabricated with a lot better precision than, e.g., the module used in the first experiment. However, by correcting the wire ends even in the case of the smallest displacement error, the quality of the connections is maximized and the possibility of false contacts is minimized. Further on, printing the ten modules at once would not have been possible without the alignment information in table 4.4, which makes it possible to automatically apply the alignment rotations of each print pattern and then catenate the patterns into a single large panel.

Next, consider the printing result shown in Figure C.1 in Appendix C. The quality of the print is not relevant from the viewpoint of this thesis. However, notice the systematic error in the alignment of the print pattern of module eight seen in Figure C.1b. The same kind of error can be seen in the printing result of modules one and five, which were shown in Figure 5.1. A closer inspection of the printing results reveal that the printed pattern of each of the modules has an error of the same magnitude and direction. This confirms that the error is due to inaccurate alignment of the print pattern. The alignment of the print pattern is a procedure done before each printing in order to get the print pattern to be printed in correct location in the xy-plane and, thus, independent of the print pattern generation system.

Finally, a few words related to the accuracy of the results are given. The amount of correction applied to the print pattern (table C.2 in Appendix C) as well as the parameters of the IC alignment transformations (table 4.4) are based on the connection point locations, which have been measured from the camera image. After adjusting the magnification of the camera such that the entire module can be seen in the image as accurately as possible, the resolution of the image is approximately $3120dpi$, i.e., the size of one pixel is about $8.14\mu m$. In the detection of the connection points, the template matched image is interpolated around each pre-detected connection point. This is done by using bicubic interpolation such that the resolution is tripled. Although the amount of information in the image stays the same, the accuracy is approximately tripled and the error in the detected location of a single connection point can be assumed to be approximately $\pm(8.14\mu m/2)/3 \approx \pm 1.36\mu m$.

In the worst case, the errors in the measured locations of all the connection points in a single IC are $1.36\mu m$ both in x and y direction. Hence, the maximum error in the translation components of the IC transformation is $\pm 1.36\mu m$. The worst case with respect to

the rotation parameter occurs in a situation where the errors are such that the connection points have been rotated about the origin of the transformation (mid-range of the connection points, that is). The rotational error is maximized when the connection point furthest away from the origin has the maximum error of $1.36\mu m$ both in x and y direction, i.e., the magnitude of the error is $\sqrt{2} \cdot 1.36\mu m$. The error in the rotation parameter of the transformation is then $2\sin^{-1}(\sqrt{2} \cdot 1.36\mu m/(2l))$, where l is the distance from the origin to the most distant connection point. By doing the math, we find out that the maximum errors in the rotation components of the determined IC transformations are $\pm 0.051^\circ$, $\pm 0.035^\circ$, $\pm 0.053^\circ$, and $\pm 0.027^\circ$ for ICs number one, two, three, and four, respectively.

More optimistic and accurate error bounds could be determined by noting that the error in the measured location of a single connection point can be assumed to be uniformly distributed. This information could be used to derive the distributions of the errors in the calculated parameters of the least squares IC transformations. In order to avoid the excessively complex calculations in deriving these distributions, we settle for the previously derived error bounds as these are adequate for us to confirm that the accuracy of the used imaging device is sufficient for making the correction of the print pattern.

Notice that more significant errors than the ones due to the camera resolution may occur in cases like shown in Figure 5.4, where the inaccuracy is caused by the connection point heuristics itself. These kinds of errors have a more systematic nature and can be considered more harmful than the ones created by the limited camera resolution. Fortunately, these kinds of faults can be handled by further development of the software.

6. CONCLUSION AND FUTURE WORK

In this thesis, a dynamic print pattern correction system for inkjet printed electronics manufacturing was introduced. The developed correction system uses computer vision to detect the connection points of the module components. The print pattern is then modified according to the measured errors in the component locations such that it matches the actual layout of the module, which may differ from the designed layout due to things like component drifting.

The experiments show that the correction system is capable of creating the corrected print pattern like it was intended. The system is shown to work in case of extensive displacements in the component locations. The system also generalizes to work with different types of modules as well as connection points with different kinds of appearance.

The correction of the print pattern is justified even in the case of the smallest displacement error by the fact that this way the quality of the connections is maximized and the possibility of false contacts is minimized. Further on, dynamic correction of the print pattern enables the use of more inaccurate techniques for component placement. This allows less expensive equipment to be used, increase in the life cycle of the existing equipment, shorter lead times, and the printing of larger designs as the errors due to material deformations increase together with the size of the module.

The major advantage in using the print pattern generation system is that it enables the printing of multiple modules with one single print. By detecting the locations of the modules and the components on them, it is possible to automatically apply the alignment rotations of each print pattern and then to catenate the patterns into a single large panel. With the conventional method, each module has to be separately printed together with their own alignment printings. The rotational error has to be compensated by manually rotating the printer table. By using an online print pattern generation system, all the modules can be printed with exactly one alignment print and one actual print without the need for operator dependent manual determination of the errors in the orientation of the modules.

Future work should concentrate on further development of the print pattern correction system. The efficiency of the software should be increased for better integrability into the manufacturing process. This includes the transportation of the software into a platform closer to the hardware. Currently the system runs on MATLAB, which suits better for development purposes. Also memory issues should be considered as the system needs to

handle high resolution camera images and massive print patterns. Using a 64-bit platform instead of a 32-bit would enable a greater amount of memory to be addressed for the correction system, thus, allowing the use of better performing code.

In addition to the hardware, there are plenty of room for optimization on the software level. Development in the individual stages of the processing chain of the whole system should be allocated, e.g., on improving the connection point detection step. In some cases, relatively large amount of connection points remain completely undetected. Fortunately, the experiments show that even in cases where only 60% of the connection points within a single IC are detected, the correction can be successfully accomplished. This is because of the robust PPM algorithm, which is used for finding the correspondence between the detected and the designed connection points.

The experiments revealed a problematic issue concerning the alignment of the print pattern. The alignment is done so that the print pattern gets printed on correct location. As the printing equipment doesn't offer any means for doing the alignment, an alignment image has to printed before the actual printing. Measuring the offset between the printed alignment image and the target area involves manual inspection, thus, allowing operator dependent humane mistakes to take place. Future work could include a development of an alignment method where the printed alignment pattern could be automatically analyzed. This way the print pattern alignment could be made more accurately and without any human intervention.

REFERENCES

[1] G. E. Moore, "Cramming more components onto integrated circuits, reprinted from electronics, volume 38, number 8, april 19, 1965, pp.114 ff." *Solid-State Circuits Newsletter, IEEE*, vol. 20, no. 3, pp. 33–35, Sep. 2006.

[2] V. Zhirnov, I. Cavin, R.K., J. Hutchby, and G. Bourianoff, "Limits to binary logic switch scaling - a gedanken model," *Proceedings of the IEEE*, vol. 91, no. 11, pp. 1934–1939, Nov. 2003.

[3] M. Dubash, "Moore's Law is dead, says Gordon Moore," [WWW], Apr. 2005, [cited on August 12, 2009]. Available: http://www.techworld.com/opsys/news/index.cfm?NewsID=3477.

[4] R. Kurzweil, "The Law of Accelerating Returns," [WWW], Mar. 2001, [cited on August 12, 2009]. Available: http://www.kurzweilai.net/articles/art0134.html.

[5] H. Huttunen, P. Ruusuvuori, T. Manninen, K. Rutanen, and R. Rönkkä, "Dynamic adaptation of interconnections in inkjet printed electronics," in *Proceedings of International Conference on Signals and Electronic Systems, ICSES'08*, Kraków, Poland, Sep. 2008, pp. 449–452.

[6] H. Huttunen, P. Ruusuvuori, T. Manninen, K. Rutanen, R. Rönkkä, and A. Visa, "Object detection for dynamic adaptation of interconnections in inkjet printed electronics," in *15th IEEE International Conference on Image Processing, ICIP 2008*, San Diego, California, USA, Oct. 2008, pp. 2364–2367.

[7] H. Huttunen, T. Manninen, K. Rutanen, P. Ruusuvuori, R. Mäkinen, and R. Rönkkä, "Dynamic correction of interconnections in printed electronics manufacturing," in *International conference on digital printing technologies, NIP25*, Louisville, Kentucky, USA, Sep. 2009, pp. 589–592.

[8] K. Rutanen, "Automated in-process correction algorithm for printed electronics," Master's thesis, Tampere University of Technology, Tampere, Nov. 2008.

[9] B. Zitova and J. Flusser, "Image registration methods: a survey," *Image and vision computing*, vol. 21, no. 11, pp. 977–1000, 2003.

[10] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[11] R. Gonzalez and R. Woods, *Digital image processing*, 2nd ed. Upper Saddle River, New Jersey, USA: Prentice-Hall, 2002, 793 pp.

[12] B. Li, Q. Meng, and H. Holstein, "Point pattern matching and applications-a review," *IEEE International Conference on Systems, Man and Cybernetics, 2003*, vol. 1, pp. 729–736, Oct. 2003.

[13] S. Ranade and A. Rosenfeld, "Point pattern matching by relaxation," *Pattern Recognition*, vol. 12, no. 4, pp. 269 – 275, 1980.

[14] D. Lavine, B. A. Lambird, and L. N. Kanai, "Recognition of spatial point patterns," *Pattern Recognition*, vol. 16, no. 3, pp. 289 – 295, 1983.

[15] W. E. L. Grimson and T. Lozano-Pérez, "Localizing overlapping parts by searching the interpretation tree," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 9, no. 4, pp. 469–482, 1987.

[16] H. Chui and A. Rangarajan, "A new point matching algorithm for non-rigid registration," *Computer Vision and Image Understanding*, vol. 89, no. 2-3, pp. 114–141, 2003.

[17] M. Carcassoni and E. Hancock, "Point pattern matching with robust spectral correspondence," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, 2000, pp. 649–655.

[18] P. Besl and H. McKay, "A method for registration of 3-D shapes," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[19] G. Stockman, S. Kopstein, and S. Benett, "Matching images to models for registration and object detection via clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 4, pp. 229–241, May 1982.

[20] G. Stockman, "Object recognition and localization via pose clustering," *Computer Vision, Graphics, and Image Processing*, vol. 40, no. 3, pp. 361–387, 1987.

[21] S. Chang, F. Cheng, W. Hsu, and G. Wu, "Fast algorithm for point pattern matching: Invariant to translations, rotations and scale changes," *Pattern Recognition*, vol. 30, pp. 311–320, Feb. 1997.

[22] C. Wylie, G. Romney, D. Evans, and A. Erdahl, "Half-tone perspective drawings by computer," in *AFIPS '67 (Fall): Proceedings of the November 14-16, 1967, fall joint computer conference*. New York, New York, USA: ACM, 1967, pp. 49–58.

[23] O. Škrinjar, *Biomedical Image Registration*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2006, vol. 4057, ch. Point-Based Registration with Known Correspondence: Closed Form Optimal Solutions and Properties, pp. 315–321.

[24] S. Umeyama, “Least-squares estimation of transformation parameters between two point patterns,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 4, pp. 376–380, 1991.

[25] K. S. Arun, T. S. Huang, and S. D. Blostein, “Least-squares fitting of two 3-d point sets,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 9, no. 5, pp. 698–700, 1987.

[26] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users' Guide*, 3rd ed. 3600 University City Science Center, Philadelphia, USA, PA 19104-2688: The Society for Industrial and Applied Mathematics, 1999, available: http://www.netlib.org/lapack/lug/lapack_lug.html.

[27] G. Golub and C. Van Loan, *Matrix computations*, 3rd ed. 2715 North Charles Street, Baltimore, Maryland, USA: The Johns Hopkins University Press, 1996, 694 pp.

[28] S. Schaefer, T. McPhail, and J. Warren, “Image deformation using moving least squares,” *ACM Transactions on Graphics*, vol. 25, no. 3, pp. 533–540, 2006.

[29] M. Müller, B. Heidelberger, M. Teschner, and M. Gross, “Meshless deformations based on shape matching,” *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 471–478, 2005.

[30] N. Otsu, “A Threshold Selection Method from Gray-Level Histograms,” *IEEE Transactions on Systems, Man and Cybernetics*, vol. 9, pp. 62–66, 1979.

[31] M. Kamel and A. Zhao, “Extraction of binary character/graphics images from grayscale document images,” *CVGIP: Graphical Models and Image Processing*, vol. 55, no. 3, pp. 203–217, 1993.

[32] Y. Yang and H. Yan, “An adaptive logical method for binarization of degraded document images,” *Pattern Recognition*, vol. 33, no. 5, pp. 787–807, 2000.

[33] W. McCulloch and W. Pitts, “A logical calculus of the ideas immanent in nervous activity,” *Bulletin of Mathematical Biology*, vol. 5, no. 4, pp. 115–133, Sep. 1943.

[34] F. Rosenblatt, “The perceptron: A probabilistic model for information storage and organization in the brain,” *Psychological review*, vol. 65, no. 6, pp. 386–408, 1958.

[35] B. Widrow and M. E. Hoff, “Adaptive switching circuits,” *IRE WESCON Convention Record*, vol. 4, pp. 96–104, 1960.

[36] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed. Upper Saddle River, New Jersey, USA: Prentice-Hall, 1999, 842 pp.

[37] M. D. Richard and R. P. Lippmann, "Neural network classifiers estimate bayesian a posteriori probabilities," *Neural Computation*, vol. 3, no. 4, pp. 461–483, 1991.

[38] H. Chernoff, "A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations," *The Annals of Mathematical Statistics*, vol. 23, no. 4, pp. 493–507, Dec. 1952.

[39] S. Bianco, F. Gasparini, and R. Schettini, "Combining strategies for white balance," *Digital Photography III*, vol. 6502, no. 1, p. 65020D, 2007.

[40] M. Møller, "A scaled conjugate gradient algorithm for fast supervised learning," *Neural Networks*, vol. 6, pp. 525–533, 1993.

A. APPENDIX: OPTIMAL RIGID AND SIMILARITY TRANSFORMATIONS IN 2-D

There exists a generalized solution for finding the optimal rigid/similarity transformation parameters between two n-dimensional point sets when the point correspondence is known. However, in some situations we can benefit from an easily implementable closed form solution that calculates the transformation parameters in a special two-dimensional case. This kind of result can be found from the appendix of the publication of Chang et al. [21]. However, Chang's solution only gives parameters of the similarity transformation, not the rigid.

This appendix introduces a theorem that gives a modified version of the Chang's formula for finding the parameters of optimal similarity transformation. The introduced formula is straightforward and can be used both in similarity and rigid cases. Before stating the theorem, let's define the optimization problem in two dimensions.

Let

$$P = \{(x_{p_k}, y_{p_k}) \in \mathbb{R}^2\}_{k=1}^{m} \tag{A.1}$$

and

$$Q = \{(x_{q_k}, y_{q_k}) \in \mathbb{R}^2\}_{k=1}^{m} \tag{A.2}$$

be two-dimensional point sets such that the k'th point in the first set corresponds to the k'th point in the second set. In order to find the parameters of the optimal rigid/similarity transformation that transforms the point set P into the point set Q in least squares sense, we need to find a scaling factor $\hat{s} \in \mathbb{R}^+$, a rotation parameter $\hat{\theta} \in [0, 2\pi)$, and translation parameters $\hat{t}_x,\ \hat{t}_y \in \mathbb{R}$, which minimize the sum of the squared errors

$$J(s, \theta, t_x, t_y) = \sum_{k=1}^{N} \left\| \begin{bmatrix} x_{q_k} \\ y_{q_k} \end{bmatrix} - \left(s \begin{bmatrix} \cos\theta & -\sin\theta \\ \cos\theta & \sin\theta \end{bmatrix} \begin{bmatrix} x_{p_k} \\ y_{p_k} \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} \right) \right\|^2. \tag{A.3}$$

In the rigid transformation case, s is equal to one.

Now, let's state the solution.

Theorem 2. *The parameters of the rigid or similarity transformation, which transform point set $P = \{(x_{p_k}, y_{p_k}) \in \mathbb{R}^2\}_{k=1}^{m}$ such that the mean square error between another set*

$Q = \{(x_{q_k}, y_{q_k}) \in \mathbb{R}^2\}_{k=1}^m$ *is minimized are*

$$\hat{s} = \frac{1}{c}\sqrt{l_{A+B}^2 + l_{A-B}^2}, \tag{A.4}$$

$$\hat{\theta} = \tan^{-1}\frac{l_{A-B}}{l_{A+B}}, \tag{A.5}$$

$$\hat{t}_x = \bar{x}_q - \frac{1}{c}(\bar{x}_p l_{A+B} - \bar{y}_p l_{A-B}), \tag{A.6}$$

and

$$\hat{t}_y = \bar{y}_q - \frac{1}{c}(\bar{y}_p l_{A+B} + \bar{x}_p l_{A-B}), \tag{A.7}$$

where $(\bar{x}_p, \bar{y}_p)$ *and* $(\bar{x}_q, \bar{y}_q)$ *are the set centroids,*

$$l_{A+B} = \sum_{k=1}^{N}\left[(x_{p_k} - \bar{x}_p)(x_{q_k} - \bar{x}_q) + (y_{p_k} - \bar{y}_p)(y_{q_k} - \bar{y}_q)\right], \tag{A.8}$$

and

$$l_{A-B} = \sum_{k=1}^{N}\left[(x_{p_k} - \bar{x}_p)(y_{q_k} - \bar{y}_q) - (y_{p_k} - \bar{y}_p)(x_{q_k} - \bar{x}_q)\right]. \tag{A.9}$$

Constant $c \in \mathbb{R}$ *is calculated according to the desired transformation type such that*

$$c = \begin{cases} l_A, & \textit{for similarity transformation} \\ \sqrt{l_{A+B}^2 + l_{A-B}^2}, & \textit{for rigid transformation} \end{cases}, \tag{A.10}$$

where

$$l_A = \sum_{k=1}^{N}\left[(x_{p_k} - \bar{x}_p)^2 + (y_{p_k} - \bar{y}_p)^2\right]. \tag{A.11}$$

Proof. First, the optimal rigid transformation is determined. This is done by fixing the scale parameter in Equation A.3 to $s = 1$.

Let

$$\begin{bmatrix} x'_{p_k} \\ y'_{p_k} \end{bmatrix} := \begin{bmatrix} x_{p_k} \\ y_{p_k} \end{bmatrix} - \begin{bmatrix} \bar{x}_p \\ \bar{y}_p \end{bmatrix} \tag{A.12}$$

and

$$\begin{bmatrix} x'_{q_k} \\ y'_{q_k} \end{bmatrix} := \begin{bmatrix} x_{q_k} \\ y_{q_k} \end{bmatrix} - \begin{bmatrix} \bar{x}_q \\ \bar{y}_q \end{bmatrix}, \tag{A.13}$$

where $(\bar{x}_p, \bar{y}_p)$ and $(\bar{x}_q, \bar{y}_q)$ are the set centroids. It follows from the Lemma 1 given on page 16 that the sum of the squared errors in Formula A.3 can be reshaped into a form where the translation parameter is eliminated. The new problem becomes determining only $\hat{\theta}$. This is done by finding the parameter vector $\hat{\mathbf{r}} = [\cos\hat{\theta}\ \sin\hat{\theta}]^T$ that minimizes the

square error

$$J(\mathbf{r}) = \sum_{k=1}^{N} {\mathbf{e}_k}^T \mathbf{e}_k, \tag{A.14}$$

where

$$\mathbf{e}_k = \begin{bmatrix} \cos\theta & -\sin\theta \\ \cos\theta & \sin\theta \end{bmatrix} \begin{bmatrix} x'_{p_k} \\ y'_{p_k} \end{bmatrix} - \begin{bmatrix} x'_{q_k} \\ y'_{q_k} \end{bmatrix} = \begin{bmatrix} x'_{p_k} & -y'_{p_k} \\ y'_{p_k} & x'_{p_k} \end{bmatrix} \mathbf{r} - \begin{bmatrix} x'_{p_k} \\ y'_{p_k} \end{bmatrix}. \tag{A.15}$$

Next, let

$$\mathrm{C} := \begin{bmatrix} x'_{p_1} & -y'_{p_1} \\ y'_{p_1} & x'_{p_1} \\ \vdots & \vdots \\ x'_{p_N} & -y'_{p_N} \\ y'_{p_N} & x'_{p_N} \end{bmatrix} \tag{A.16}$$

and

$$\mathbf{b} := \begin{bmatrix} x'_{q_1} \\ y'_{q_1} \\ \vdots \\ x'_{q_N} \\ y'_{q_N} \end{bmatrix}. \tag{A.17}$$

Equation A.14 can now be written in matrix form as

$$J(\mathbf{r}) = (\mathrm{C}\mathbf{r} - \mathbf{b})^T (\mathrm{C}\mathbf{r} - \mathbf{b}) = \mathbf{r}^T \mathrm{C}^T \mathrm{C}\mathbf{r} - 2\mathbf{b}^T \mathrm{C}\mathbf{r} + \mathbf{b}^T \mathbf{b}. \tag{A.18}$$

Before minimizing $J(\mathbf{r})$, an important constraint is needed to point out. Namely, in order for $\mathbf{r} = [\cos\theta \ \sin\theta]^T$ to be valid, it needs to fulfill the trigonometric identity, i.e., $\mathbf{r}^T\mathbf{r} = 1$. The problem now becomes a constrained minimization problem, which can be solved by using Lagrange multipliers. The function to be minimized becomes

$$\Lambda(\mathbf{r}, \lambda) = J(\mathbf{r}) + \lambda(\mathbf{r}^T\mathbf{r} - 1), \tag{A.19}$$

where $\lambda \in \mathbb{R}$ is a Lagrange multiplier. Calculating the partial derivatives with respect to $\mathbf{r}$ and λ yields in

$$\frac{\partial \Lambda(\mathbf{r}, \lambda)}{\partial \mathbf{r}} = J'(\mathbf{r}) + 2\lambda\mathbf{r} = 2\mathrm{C}^T\mathrm{C}\mathbf{r} - 2\mathrm{C}^T\mathbf{b} + 2\lambda\mathbf{r} = 2(\mathrm{C}^T\mathrm{C} + \lambda\mathrm{I})\mathbf{r} - 2\mathrm{C}^T\mathbf{b} \tag{A.20}$$

and

$$\frac{\partial \Lambda(\mathbf{r},\lambda)}{\partial \lambda} = \mathbf{r}^T\mathbf{r} - 1 \tag{A.21}$$

The optimal parameter vector $\hat{\mathbf{r}}$ can now be solved from Equation A.20 by setting it equal to zero. This results in

$$\hat{\mathbf{r}} = (\mathrm{C}^T\mathrm{C} + \lambda \mathrm{I})^{-1}\mathrm{C}^T\mathbf{b} = \begin{bmatrix} l_A + \lambda & 0 \\ 0 & l_A + \lambda \end{bmatrix}^{-1} \begin{bmatrix} l_{A+B} \\ l_{A-B} \end{bmatrix} = \frac{1}{l_A + \lambda} \begin{bmatrix} l_{A+B} \\ l_{A-B} \end{bmatrix}, \tag{A.22}$$

where l_{A+B}, l_{A-B}, and l_A are defined in Equations A.8, A.9, and A.11, respectively. Substituting the obtained $\hat{\mathbf{r}}$ into the partial derivative of Equation A.21 and setting this to zero in turn gives

$$\hat{\mathbf{r}}^T\hat{\mathbf{r}} - 1 = \frac{l_{A+B}^2 + l_{A-B}^2}{(l_A + \lambda)^2} - 1 = 0. \tag{A.23}$$

Solving $l_A + \lambda$ yields in

$$l_A + \lambda = \sqrt{l_{A+B}^2 + l_{A-B}^2}. \tag{A.24}$$

Finally, by substituting $l_A + \lambda$ into the Equation A.22, we get

$$\hat{\mathbf{r}} = \frac{1}{\sqrt{l_{A+B}^2 + l_{A-B}^2}} \begin{bmatrix} l_{A+B} \\ l_{A-B} \end{bmatrix}. \tag{A.25}$$

Now that the optimal rotation has been acquired, the translation term can be calculated as

$$\begin{bmatrix} \hat{t}_x \\ \hat{t}_y \end{bmatrix} = \begin{bmatrix} \bar{x}_q \\ \bar{y}_q \end{bmatrix} - \begin{bmatrix} \bar{x}_p & -\bar{y}_p \\ \bar{y}_p & \bar{x}_p \end{bmatrix} \hat{\mathbf{r}}, \tag{A.26}$$

i.e., the translation is such that it translates the centroid of P into the same location with the centroid of Q. This concludes the proof for the rigid transformation.

To extend the result also for similarity transformation, the parameter vector is considered of form $\hat{\mathbf{r}} = s[\cos\hat{\theta}\ \sin\hat{\theta}]^T$. This gives a result that is similar to the one in Equation A.22. The exception is that the Lagrange multiplier λ is equal to zero as there are no constraints for $\hat{\mathbf{r}}$. The solution is achieved after solving s and θ, which concludes the proof for the similarity transformation. □

B. APPENDIX: LABORATORY DIARY

Time	Event	Notes
12:30	Session started	
12:35	Hardware setup	Camera cables, led light installation, laptop reboot
12:35	Installing modules on the jig	The infrared thermometer had to be retrieved from the office
13:00	Camera testing	Focus and magnification were adjusted
13:15	Calibration plate	The plate was missing but was finally found from one of the laboratory cupboards
13:45	Camera calibration	
13:55	Photographing of modules	
14:05	Running the correction software	MATLAB once crashed due to out of memory error in the initial processing of one of the images. Processing was successfully continued after a restart.
14:40	Print image alignment	
15:15	Printing	
15:20	Cancel	Print patterns of different modules seemed to be catenated with incorrect spacing
15:30	Module cleaning	
15:30	Bug fix	
15:50	Reprinting	
16:15	Photographing of results	
16:25	Cleaning up the place	
16:30	Session ended	

C. APPENDIX: EXPERIMENTAL RESULTS

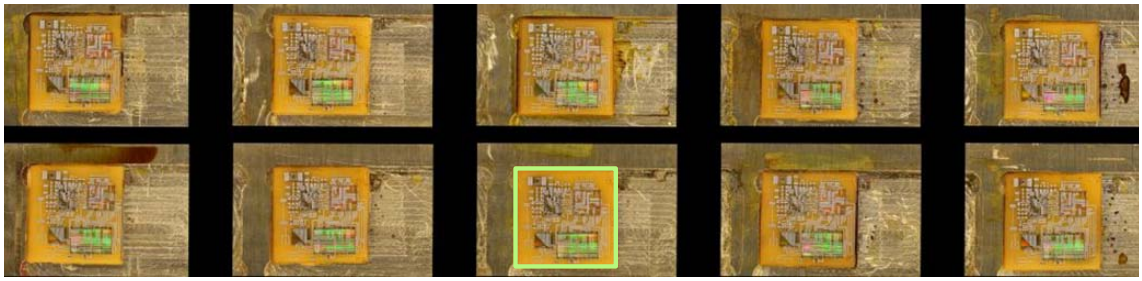
This appendix collects some of the results of the third experiment where wiring was simultaneously printed on ten modules. Table C.1 shows the results of the connection point detection step. In table C.2, there is statistics about the amount of error in the component locations. The printing result is shown in Figure C.1.

Table C.1: Connection point detection rate in the third experiment. (⋆) Module number six is the calibration module used to pick the connection point template and to train the classifier.

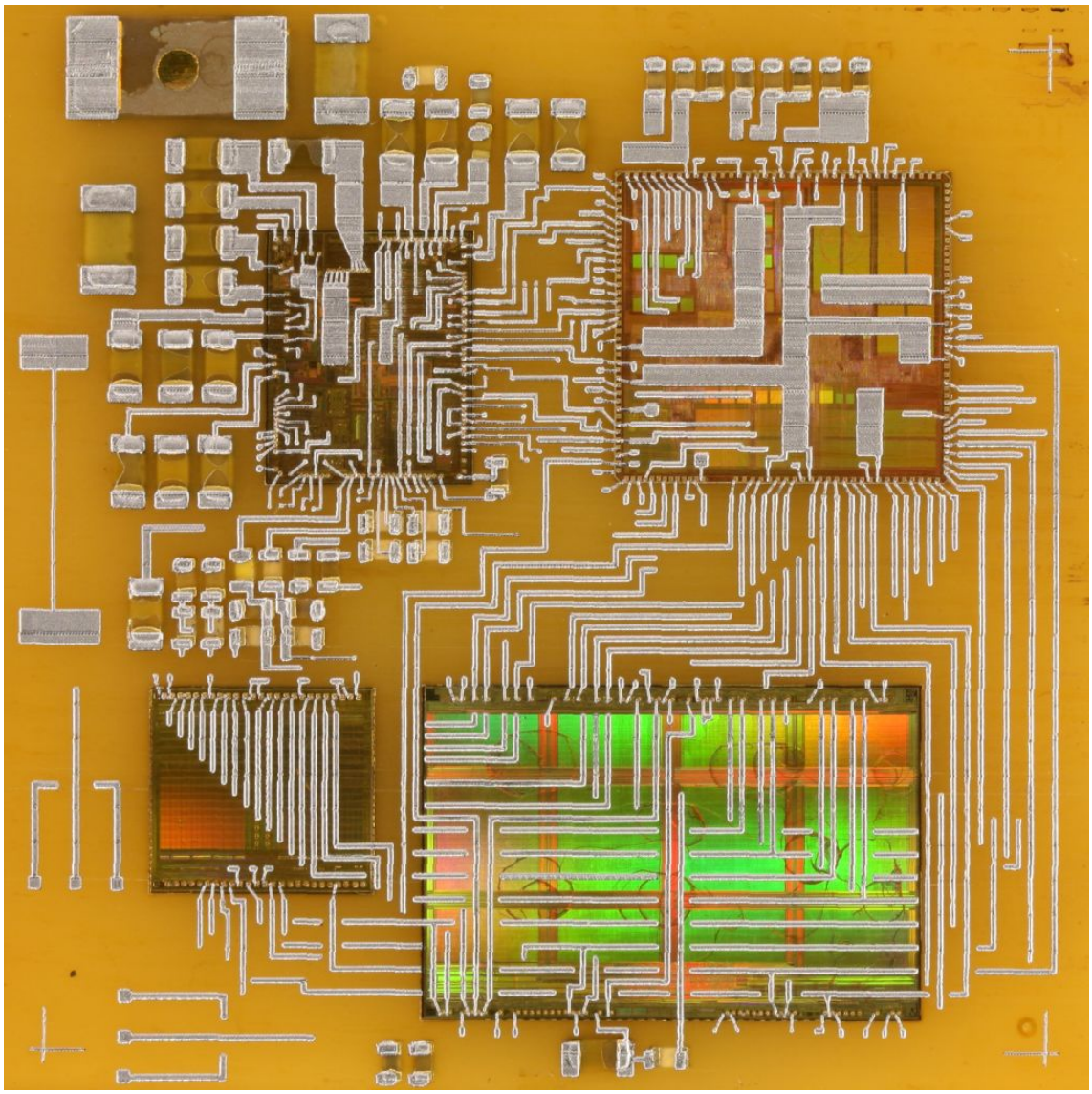
Module		True Pos.	True Neg.	False Pos.	False Neg.	Undetected
1	IC1	81	158	21	0	47
	IC2	122	82	2	26	7
	IC3	39	116	1	11	0
	IC4	60	52	1	14	2
2	IC1	77	210	25	4	47
	IC2	124	80	0	25	6
	IC3	33	118	7	17	0
	IC4	61	91	0	15	0
3	IC1	72	225	30	8	48
	IC2	123	92	1	16	16
	IC3	35	109	5	15	0
	IC4	60	97	1	10	6
4	IC1	85	194	17	0	43
	IC2	124	81	0	29	2
	IC3	33	117	7	15	2
	IC4	61	96	0	15	0
5	IC1	80	179	22	2	46
	IC2	121	93	3	27	7
	IC3	30	87	10	20	0
	IC4	61	71	0	15	0
6⋆	IC1	68	223	34	1	59
	IC2	124	52	0	27	4
	IC3	40	96	0	10	0
	IC4	61	76	0	15	0
7	IC1	88	197	14	5	35
	IC2	124	103	0	16	15
	IC3	32	120	8	16	2
	IC4	61	81	0	15	0
8	IC1	81	182	21	2	45
	IC2	124	65	0	30	1
	IC3	37	128	3	13	0
	IC4	61	91	0	15	0
9	IC1	74	176	28	3	51
	IC2	123	106	1	18	14
	IC3	30	110	10	20	0
	IC4	61	76	0	15	0
10	IC1	81	209	21	0	47
	IC2	124	114	0	29	2
	IC3	39	108	1	11	0
	IC4	60	90	1	15	1

Table C.2: Displacements of each IC and their connection points in the third experiment.

Module		Rotation (degrees)	Translation X (μm)	Translation Y (μm)	Avg. Error (μm)	Max. Error (μm)
1	IC1	0.05	2	7	7	9
	IC2	0.05	-2	-1	3	5
	IC3	0.08	5	-6	8	11
	IC4	0.11	0	-3	6	10
2	IC1	0.03	1	4	5	6
	IC2	0.07	-1	2	3	6
	IC3	-0.06	4	-1	5	6
	IC4	0.05	-1	-8	8	12
3	IC1	-0.02	2	4	5	6
	IC2	0.03	-3	4	5	6
	IC3	0.06	6	-6	9	11
	IC4	-0.05	-2	-9	9	12
4	IC1	0.02	1	7	7	8
	IC2	0.05	-0	2	3	4
	IC3	-0.07	-2	-8	8	10
	IC4	-0.03	0	-8	8	10
5	IC1	0.03	3	5	6	7
	IC2	0.04	-3	2	4	6
	IC3	-0.09	1	-11	11	14
	IC4	0.04	2	-7	8	10
6	IC1	0.06	-1	6	6	8
	IC2	0.00	0	1	1	1
	IC3	0.04	-2	-6	7	8
	IC4	-0.01	2	-3	3	4
7	IC1	0.06	-2	5	6	8
	IC2	0.05	2	-3	4	7
	IC3	0.00	-4	-1	4	4
	IC4	0.03	-0	-1	2	3
8	IC1	0.07	0	4	5	6
	IC2	0.05	-1	2	3	5
	IC3	0.12	5	-4	6	10
	IC4	-0.01	0	-6	6	6
9	IC1	0.02	1	6	6	7
	IC2	0.06	-1	3	4	7
	IC3	0.01	3	-7	8	8
	IC4	-0.03	0	-8	8	9
10	IC1	0.02	-1	8	9	9
	IC2	0.08	0	0	4	5
	IC3	0.07	3	-5	6	8
	IC4	0.02	-1	-7	7	8



(a)



(b)

Figure C.1: Printing result of the third experiment. Figure (a) shows the whole panel. The module inside the green square can be seen in detail in Figure (b).