TU WIEN Informatics

Automated Stitching for Scanning Electron Microscopy Images of Integrated Circuits

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Software Engineering and Internet Computing

eingereicht von

Daniel Burian, BSc
Matrikelnummer 00825451

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ.-Prof. Dipl.-Ing. Mag. Dr. techn. Edgar Weippl
Mitwirkung: Univ.Lektor Dipl.-Ing. Dr.techn. Georg Merzdovnik, MSc
Univ.Lektor Dipl.-Ing Christian Kudera, MSc

Wien, 27. Jänner 2022

Daniel Burian | Edgar Weippl

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

TU WIEN Informatics

Automated Stitching for Scanning Electron Microscopy Images of Integrated Circuits

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Software Engineering and Internet Computing

by

Daniel Burian, BSc
Registration Number 00825451

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.-Prof. Dipl.-Ing. Mag. Dr. techn. Edgar Weippl
Assistance: Univ.Lektor Dipl.-Ing. Dr.techn. Georg Merzdovnik, MSc
Univ.Lektor Dipl.-Ing Christian Kudera, MSc

Vienna, 27th January, 2022

Daniel Burian | Edgar Weippl

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Erklärung zur Verfassung der Arbeit

Daniel Burian, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 27. Jänner 2022

Daniel Burian

Acknowledgements

I would like to thank my supervisor Edgar Weippl, as well as my friends and colleagues Christian Kudera, Georg Merzdovnik, and Markus Müllner for their support and feedback.
Furthermore, I would like to express my sincere gratitude to Markus Kammerstetter and Trustworks GmbH for all their support, especially in the form of providing access to their scanning electron microscope, as well as existing image data.
My sincere thanks also go to Wolfgang Kastner and the Automation Systems Group at TU Wien, who hosted the Hardware Security Lab, where a substantial amount of research for this work was conducted.
I would also like to thank the Austrian Study Grant Authority for the opportunity to finish this work within the scope of a scholarship (Studienabschlussstipendium).
Finally, I would like to thank my wife Eszter and my son David for giving me time to work on my thesis as well as providing distraction when I needed to disentangle my thoughts.

Kurzfassung

Während etablierte Reverse-Engineering Toolchains für Software, Firmware und zu einem gewissen Grad auch für Hardware, in Form von Schnittstellen von oder zwischen Komponenten, existieren, ist die Hürde zum Einstieg in das Reverse Engineering von Mikrochips nachwievor hoch. Dies liegt vor allem an den Kosten für das benötigte Equipment, insbesondere an den Kosten eines dem Stand der Technik entsprechenden Rasterelektronenmikroskops (REM). Ein solches REM wird aufgrund der kleinen Fertigungsgrößen benötigt, da optische Mikroskope in vielen Fällen nicht die hierfür benötigte Vergrößerung bieten.

REM werden im Rahmen des Reverse-Engineering eingesetzt um Bilder der einzelnen Lagen eines Chips während des iterativen Delayering-Prozesses zu erstellen. Ziel dieses Prozesses ist es, mittels des erstellten Bildmaterials die eingesetzten Logik-Gatter und deren Verbindungen festzustellen. Darauf aufbauend kann die Funktionalität der implementierten Logik nachvollzogen werden. Weiters ist es in einigen Fällen möglich, den Inhalt von Read-Only Memory (ROM) im Chip anhand des Bildmaterials auszulesen.

REMs erzeugen üblicherweise Bilder mit einer Auflösung in der Größenordnung mehrerer Megapixel. Da eine Schicht eines Chips aber üblicherweise eine Gigapixel-Auflösung voraussetzt um ausreichend Details abbilden zu können, muss eine Vielzahl an Bildern pro Schicht angefertigt werden. Die so erzeugten Einzelbilder werden anschließend zu einem großen Gesamtbild zusammengefügt. Verglichen mit optischen Mikroskopen bringt dieser Vorgang bei REMs besondere Herausforderungen mit sich, insbesondere die Verzerrung der Bilder durch Ladungseffekte, sowie oftmals kontrastarmes und verrauschtes Bildmaterial.

Eine Möglichkeit, wie die hohe Einstiegshürde im Berech des Mikrochip-Reverse-Engineerings gesenkt werden kann, ist eine Reduktion der Anforderungen an die Bildqualität. Gelingt dies, können auch ältere, gebrauchte und dementsprechend günstigere REMs zur Erstellung des Bildmaterials genutzt werden.

Im Rahmen dieser Arbeit werden mehrere neue Algorithmen vorgestellt und evaluiert, mit der Zielsetzung verrauschtes, kontrastarmes Bildmaterial solcher älterer REM erfolgreich verarbeiten zu können. Zunächst wird der am besten geeignete bestehende Image-Registration Algorithmus festgestellt. Zur Evaluierung der zur Verfügung stehenden Kandidaten werden Einzelbilder mehrerer Chip-Schichten mit einer Gesamtgröße mehrerer Gigapixel erstellt. Für die am besten geeigneten Image-Registration Algorithmen

wird anschließend automatisierte Parametrierung anhand des eingesetzten Bildmaterials entwickelt und evaluiert.

Aufbauend auf den Ergebnissen des besten Image-Registration Algorithmus werden anschließend vier neue Global-Stitching Algorithmen vorgestellt, deren Aufbau für unterschiedliche qualitative Metriken optimiert ist. Diese Global-Stitching Algorithmen erstellen das Gesamtbild einer Schicht des untersuchten Chips, aufbauend auf den Ergebnissen des eingesetzten Image-Registration Algorithmus. Abschließend werden diese Algorithmen evaluiert indem die hierbei erstellten zusammengefügten Bilder sowohl miteinander, als auch mit aktuellen, dem Stand der Technik entsprechenden Image-Stitching Tools verglichen werden.

Abstract

While established toolchains are widely available for reverse engineering software, firmware, and even hardware on the printed circuit board level to some degree, the entry barrier to reverse engineering of integrated circuits (ICs) remains high due to the associated cost of equipment. One key driver of this high cost is the requirement of high-quality scanning electron microscopes (SEMs) for the analysis of ICs with small feature sizes. While optical microscopes are a cost-effective alternative for large feature sizes, modern ICs are manufactured with feature sizes that are too small for the limited magnification of optical microscopes.

In IC analysis, SEMs are used to create images of individual layers of integrated circuits during an iterative delayering process. The aim of this process is to image logic gate placement and interconnects, from which detailed information about the implemented logic functions can be recovered. In some cases, it is also possible to extract the contents of read-only memory (ROM) on the chip.

An SEM usually creates images in the range of megapixel resolutions, but analyzing an IC layer requires resolutions in the gigapixel range. To create such large images, many individual images must be taken and then fused into one large image. Compared to images created by optical microscopes, SEM images pose unique challenges: They are affected by distortion due to charging effects and often exhibit high levels of noise and low contrast.

One way of reducing the entry barrier to IC reverse engineering is to develop algorithms that can provide good results even in the case of suboptimal image quality, as can be produced by comparatively cheap, used SEMs.

This thesis introduces and evaluates several algorithms for the purpose of fusing noisy images with low contrast created by older SEMs. Based on an evaluation using gigapixel scale image sets, the most efficient and effective image registration algorithm for these image properties is determined. These algorithms determine offsets between individual overlapping images in the image set. For the two best algorithms, automated inference of optimal parameters is developed. Four global stitching algorithms are introduced, to create large fused images based on the results of image registration. These four algorithms optimize for different quality metrics in the generated fused image. Finally, the introduced algorithms are evaluated and compared to state-of-the-art image stitching software.

Contents

CHAPTER 1

Introduction

1.1 Motivation

Reverse engineering integrated circuits (ICs) is a sophisticated process, primarily used for detailed technical analysis, such as evaluation of IC security mechanisms, for the assessment of utilized intellectual property and detection of counterfeits [1]. Typical approaches to reverse engineering ICs start with the decapsulation of the IC followed by iterated delayering and imaging. The resulting images can then be analyzed to recover the netlist, which describes the components and interconnects of the IC. This process yields very detailed information about the internals of the IC. Unfortunately, the entry barrier to this kind of analysis is high due to the cost of required equipment.

While optical microscopes are comparatively inexpensive, current IC manufacturing processes create structures that are in most cases too small to be imaged this way [2]. Scanning electron microscopes (SEMs) can provide the required image resolution but are orders of magnitude more expensive [3]. Depending on the available financial budget, pre-owned and therefore cheaper SEMs may be a suitable compromise. However, cheaper pre-owned SEMs typically utilize older technology, limiting available beam current at the required spot size [4]. The result of these restrictions is that images produced by such SEMs are likely to contain significant noise and low contrast at the required levels of magnification.

One way to reduce the entry barrier to reverse engineering ICs is to improve image processing capabilities to allow for the use of noisy, low contrast images produced by older SEMs operating close to their magnification limits. Image stitching algorithms that still work with such low-quality images would be an important contribution towards tackling this challenge.

The problem of stitching many small overlapping images into one fused image is not new. One example is the discussion of efficient algorithms for motion analysis in Multireso-

lution Image Processing and Analysis, published in 1984 [5]. Technology has improved significantly since then, and better algorithms, particularly in the field of feature detection, have been published in the meantime [6, 7]. However, the noise and distortion in the previously mentioned low-quality SEM images pose problems for currently publicly available stitching software, especially when combined with areas of ICs that show few discernible features in the overlapping areas of adjacent images. One possible reason for this is that available software was developed for different fields of research, such as cell biology or neuroscience, each with its own requirements and challenges that differ from those of reverse engineering [8].

The improvement of available stitching algorithms and image stitching software, focusing on their suitability for IC reverse engineering, has the potential to reduce the entry barrier for research in this area. In the best case, the improvements will benefit other areas of research where microscopic image processing is utilized, such as Bioinformatics [9].

1.2 Problem Statement

This work aims to introduce and evaluate algorithms that tailor to a corner case of image stitching to improve upon currently available state-of-the-art tools. This corner case arises from the use of affordable pre-owned scanning electron microscopes to create images of integrated circuits for the purpose of reverse engineering IC components. Since pre-owned SEMs do not provide the visual clarity of current cutting-edge SEMs at higher magnifications, improved algorithms are required to deal with comparatively high levels of noise and low contrast in the created images. These challenges are aggravated by SEM-specific imaging properties such as distortion through charging effects. To achieve this goal, several problems must be addressed:

- **Image Registration:** A plethora of image registration algorithms are available, but not all are equally efficient considering the challenges posed by noise, low contrast, and distortion. It is therefore necessary to evaluate which image registration algorithms are best suited for this task.
- **Automated Determination of Parameters:** Ideally, the resulting tool should not require any information beside the input images and their ordering. Depending on the used image registration algorithm, parameters such as expected overlap or sensitivity for keypoint detection are required to achieve good results. Therefore, ways to automatically determine good or even optimal values for the required parameters can drastically improve usability.
- **Global Stitching:** Unlike stitching panorama photos or images created by optical microscopes, distortion in SEM images can vary significantly from image to image. This can pose significant problems when creating the fused image based on the individual offsets between all adjacent images. This is because changing the order in

which the images are added to the fused image will change the outcome significantly. Therefore, a good global stitching algorithm will minimize disagreement between the previously computed individual offsets and the final location of any given part of the fused image.

1.3 Methodology

In the first step, the current state of available image stitching software is assessed. Furthermore, a literature survey is conducted to determine suitable image registration algorithms for determining offsets between overlapping images.

In the second step, two large image test sets are created, which exhibit varying degrees of problems that are typical for IC Images created by SEMs. While two test sets may seem like a small number, each test set consists of over 400 individual images, with an overall size of 1.6 to 2 gigapixels per image set. Furthermore, these two test sets contain complete layers of the imaged ICs, thus assuring that areas that are easy to stitch are visible, as well as more challenging areas.

In the third step, the performance of multiple image registration algorithms is evaluated using the previously created test sets. These algorithms aim to detect the correct offset between two adjacent overlapping images. In this comprehensive evaluation, multiple parameters are tested for each algorithm, and the results are compared to manually determined offsets. For the best algorithms, automated detection of optimal parameters is investigated using the two test sets.

Based on the results of the third step, in the fourth step, multiple global stitching algorithms are developed that use the results of the previously determined best image registration algorithm to assemble the complete picture.

Prior to evaluation, a tool to execute the whole workflow for stitching an image set is developed. This tool incorporates the results of the previous steps: The most suitable image registration algorithm is run on the image set, using automatically determined parameters. The image registration results are then used as input to one or multiple global stitching algorithms, and the results are exported as fused image as well as in the form of metadata.

In the fifth step, suitable metrics are chosen to evaluate the quality of the stitched images. Subsequently, the algorithms are evaluated according to these metrics. To the extent this is possible using only the stitched images without detailed metadata, the fused images are also compared to the results produced by state-of-the-art image stitching software. For this fifth step, the two previously created test sets and an additional image set are used.

1.4 Structure of the Work

The further chapters of this work are structured as follows: Chapter 2 gives a short introduction to IC reverse engineering, scanning electron microscopy imaging, specifically focusing on distortions as observed in available test sets, and image stitching with a focus on image sets produced by microscopy automation tools.

Chapter 3 discusses open-source and proprietary state-of-the-art software products and their limitations when applied to noisy, distorted images.

Chapter 4 introduces several promising image registration algorithms, followed by their evaluation in Chapter 5. For the two best algorithms, automated discovery of the required parameters is discussed in Chapter 6.

Chapter 7 introduces algorithms to generate fused images from image sets based on an image registration algorithm's output. A detailed evaluation follows in Chapter 8, in which the performance of these algorithms is examined according to different metrics. Furthermore, in an extensive manual evaluation, the quality of generated images is compared to images stitched by state-of-the-art software.

Finally, Chapter 9 concludes this work and ends with an outlook on possible future work.

CHAPTER 2

Background

This chapter aims to provide a short overview of reverse engineering integrated circuits, the challenges of the imaging process when using scanning electron microscopes, and the general workflow of image stitching tools.

2.1 Integrated Circuit Reverse Engineering

The aim of reverse engineering an integrated circuit (IC) is to reconstruct the details of its inner workings in the form of a high-level netlist from the physical IC. The reasons for reverse engineering include detecting hardware trojans, analyzing intellectual property infringement, rebuilding obsolete parts, security analysis, and detecting counterfeit ICs that may influence reliability [10–12].

Reverse engineering ICs can be done either non-destructively or destructively. Non-destructive reverse engineering is possible using X-ray tomography but requires an expensive synchrotron light source [13]. The X-ray tomography data allows viewing individual layers of the IC with a resolution of up to 10nm [14].

Destructive reverse engineering often requires multiple ICs, as the process of delayering is typically somewhat error-prone. The workflow for destructive reverse engineering usually consists of the following steps:

1. **Decapsulation:** The IC package (shown in Figure 2.1) is removed to expose the die. Processes for decapsulation include wet and dry etching, milling, polishing, laser ablation, or using a focused ion beam (FIB) workstation [15]. After exposing the die, its surface is cleaned to prepare it for imaging.

2. **Imaging:** Using a scanning electron microscope (SEM), images of the exposed layer of the IC are taken. The microscope stage moves the die in a grid pattern to

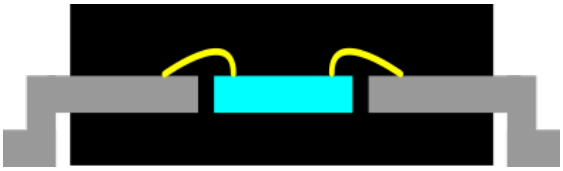


Figure 2.1: Simplified side view of an IC package. The die (blue) is connected to the lead frame (grey) via bonding wires (yellow). Decapsulation removes the package (black) to expose the die.

take overlapping images of the whole layer. The number of images is typically in the hundreds to thousands when imaging the whole layer. Optical microscopes may provide sufficient resolution in case of very large feature sizes (greater than $0.25\mu m$ according to Torrance et al. [2]). The imaging process stays the same in this case, although issues related to the limited depth of field in optical microscopes may require adaptations such as the use of z-stacking [15, 16].

3. **Delayering:** After imaging, the currently exposed layer is removed using a combination of wet and dry etching as well as polishing to expose the next layer underneath. Imaging and delayering steps are repeated for all layers [2, 15].

4. **Image Processing:** The images of each layer are combined into fused images. These fused images are then aligned to each other so that tracing connections and features across multiple layers becomes possible [11, 15, 16].

5. **Gate-Level Netlist Extraction and Verification:** Devices and their connections are identified using image recognition. Manual or automated verification is utilized to find errors in component identification, the image fusion process, or in the underlying images. Such verifications include checking for missing connections or disconnected traces. The result of this step is a gate-level netlist that identifies the logic gates and their interconnects [2, 12, 15, 16].

6. **High-Level Netlist Extraction:** Based on the gate-level netlist, a high-level netlist is generated. This process is an area of ongoing research and typically involves a combination of various automated approaches and manual analysis [17–21].

This process results in a high-level hierarchical netlist, which includes high-level components as well as their underlying logic in sufficient detail for simulation or analysis.

In the following chapters, this work will focus on one task within the image processing step, where individual images of the IC layers are stitched into fused images. The aim is to optimize this process for the unique challenges posed by noisy and distorted images produced by scanning electron microscopes operating at their capacity limits or by minor sample preparation errors.

2.2 Scanning Electron Microscopy

This section gives a short overview of the components and operating principles of scanning electron microscopes and relevant characteristics of the images created by SEMs for this work. A much more detailed description can be found in L. Reimer's book on scanning electron microscopy [22], which provided the basis for a large part of this chapter.

Scanning electron microscopes use an electron beam to scan over a given sample and measure the interactions of the electrons with the sample. State-of-the-art SEMs can resolve features down to sub-nanometer imaging resolutions. SEMs produce greyscale images because detectors of SEMs measure the interactions of the electron beam with the sample by quantifying the amount of energy deposited by electrons hitting the detector within a given timeframe.

While optical microscopes are sufficient for large feature sizes up to $0.25\mu m$, smaller feature sizes require the use of SEMs due to their higher magnification [2]. Another benefit of SEMs compared to optical microscopes is that the depth of field is considerably higher, making z-axis adjustments unnecessary when imaging large sections of flat samples such as integrated circuits if the sample is aligned correctly. Other less relevant differences include the characteristics of distortions as well as the fact that optical microscopes typically provide multiple color channels.

The subsequent sections outline the basic design of an SEM with a focus on facts relevant to imaging integrated circuits.

2.2.1 Basic SEM Components

Primary electrons that form the electron beam are generated in the electron source. This is done by heating the filament of a thermal emission source, such as a tungsten filament, or by utilizing a field emission cathode. The emitted electrons are then accelerated towards the anode in the column, using the differential in electrical charge between the negatively charged electron source and the positively charged anode. The electron beam is converged by the condenser lens and subsequently directed by the scan coils to generate the raster pattern required to create an image of the sample. The objective lens converges the beam to focus it on the sample.

The column and sample chamber need to be under vacuum to avoid atoms and molecules that do not belong to the sample interacting with the electrons of the electron beam. While low vacuum imaging is possible, high vacuum decreases the scattering of secondary and backscattered electrons, allowing for higher magnification.

2.2.2 Detectors

Two types of electron beam interactions with the sample are relevant for creating SEM images: Secondary electrons (SE) and backscattered electrons (BSE). Secondary electrons are low-energy electrons (<50eV) emitted when the electron beam's high-energy primary

electrons hit the sample [23]. These interactions occur near the surface of the sample. The secondary electrons are deflected towards the detector by a positively charged grid. One implementation of such a detector is using a scintillator that emits photons when hit by secondary electrons. A photomultiplier is then used to convert the incident photons into an electric signal.

On the other hand, backscattered electrons result from elastic interactions of the electron beam with the atomic nuclei of the sample. This interaction causes some of the beam's high-energy electrons to be reflected, with the direction in which the electrons are reflected depending on the angle at which the beam hits the sample. Heavier elements can deflect incident electrons better than lighter elements, making them appear brighter on an image generated using a BSE detector. This interaction is not limited to the surface of the sample but also occurs deeper inside the sample with the depth depending on acceleration voltage and composition of the sample [22].

Secondary electron detectors are poorly suited to detect backscattered electrons, as these high-energy electrons are not deflected by the positively charged grid used to attract low-energy electrons towards the SE detector. Since the path of the backscattered electrons depends on the angle at which the electron beam hits the sample, backscatter detectors are placed symmetrically above the sample, concentric with the electron beam. Additionally, the utilized detector can be segmented to attain topological contrast where the sample appears to be illuminated from the angle of the segment sensor. Since BSE are high-energy electrons (>50eV), detectors are typically semiconductor-based.

There are other types of detectors for SEMs, such as detectors for energy-dispersive x-ray spectroscopy (EDX). However, since these detectors are not relevant for imaging ICs, their technical details are omitted here. SE and BSE detectors are both suitable for imaging ICs. Which detector produces better images for ICs depends on the quality of sample preparation. While SE detectors often produce images with higher contrast, BSE detectors are more resilient regarding surface charge accumulation [24], which is discussed in detail in the following section.

2.2.3 Noise and Distortion

Fluctuations in the electron emission, which follow a Poisson distribution, cause shot noise. This type of noise is the dominant noise source in SEMs with thermionic electron guns [25]. For SEMs, sources for noise include primary emission, secondary emission, scintillator, photocathode, and photomultiplier [26, 27]. Additional sources of noise can arise from the internal noise of electronic components such as amplifiers at the end of the detection chain [27]. The overall pixel density distribution fits neither Poisson nor Gauss distributions correctly [28].

This noise can be reduced by decreasing scan speed or increasing beam energy. However, since slower scan speeds increase image acquisition time, tradeoffs between noise and image acquisition time are often necessary. Furthermore, increasing beam energy or decreasing scan speed can increase charging effects in non-conductive materials. These

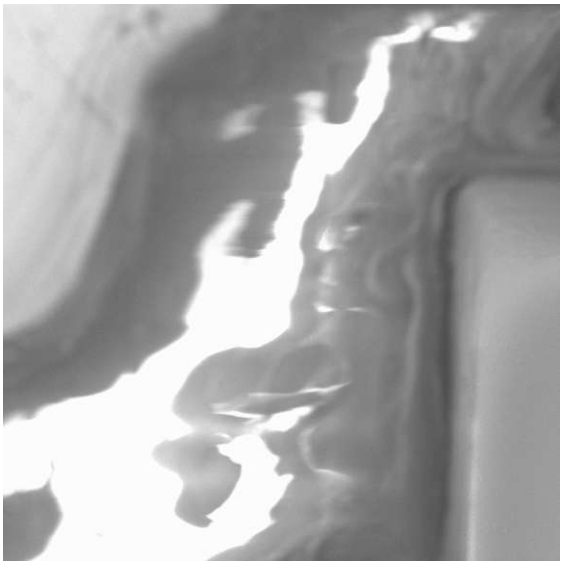


Figure 2.2: Charging effect causes increased brightness and distortion in areas with charge build-up.

effects are caused by an accumulation of charge on the sample's surface, which deflects primary and secondary electrons. It can cause distortion, abnormal contrast, bright lines, and other unusual phenomena in the produced images [29]. An example of such charging effects is shown in Figure 2.2. Charging effects occur in non-conducting material and can be minimized by coating the sample with conductive material [30]. In the case of semiconductor samples, the use of conductive coating makes microprobing impossible, and the layer can damage the sample indirectly [29]. Another downside of coating is the additional time required for the coating process and subsequent removal after imaging of the current layer is complete.

For the use case of IC reverse engineering, the distortion and noise of SEM images can pose significant challenges in the image processing and post-processing phases of the reverse engineering process. While these effects can be reduced with more expensive equipment and more time-consuming imaging processes, maximizing the utility of noisy, distorted images by creating more robust algorithms for image processing and post-processing can be far more cost-effective.

2.3 Image Stitching

The general workflow of stitching fused images consists of three steps: First, the relation between images is analyzed using image registration. Finding keypoints that are present in two or more images or comparing whole areas and assessing their similarity leads to the required information to fit the images to the camera or stage motion model. In the next step, using the data from step 1, images are mapped to the motion model, determining

the position of each individual image in the fused image. Finally, the fused image is created from the individual images. The individual images may be warped during this step to counteract distortion. Blending may also be utilized to smooth out differences between overlapping image sections.

When restricting image sets to SEM images of ICs, stitching is drastically simplified: Since SEMs provide a high depth of field and the surface of ICs is flat, it is reasonable to assume the IC area will always be in focus. Furthermore, since the planar surface of the IC should be aligned almost perfectly horizontally when creating image sets, the movement model for stitching can ignore the z-axis. Therefore, the stitching algorithm only needs to handle translation along the x- and y-axis. Relaxing the requirements further, images in the set are arranged in a grid pattern. This stems from the fact that image creation is automated due to its time-consuming nature, using microscopy automation tools such as µManager [31]. To create an image set with such an automation tool, the user selects the starting point, the number of rows and columns as well as the desired offset between images. Additionally to creating the images, these tools then produce metadata files that include each image's position on the image grid. These positions are not exact x,y-coordinates, but indices for the images along the x- and y-axis of the image grid, such as "3rd row, 2nd column". With this information, the stitcher has to find the correct offset between overlapping images using an image registration algorithm.

After computing the offset between adjacent images on the image grid, the main challenge is finding optimal locations of each individual image in the fused image. The usual workflow that the algorithms introduced in this work as well as state-of-the-art tools follow is to start at any one image by assigning it a location (e.g., 0x,0y) and then iteratively computing the location of adjacent images by summing up the location of the current image with the offset to the adjacent image [32, 33].

The progress of such an algorithm may be viewed as a graph within the image grid, as Figure 2.3 shows. The main difficulty when creating the fused image lies in unavoidable image registration errors, which may be caused by distortion, discretization, noise, or a lack of features in the overlapping area. These errors vary in intensity and may add up along paths through the graph/grid, as shown in Figure 2.4. Thus, the quality of fused images based on different graphs in the same grid may vary drastically.

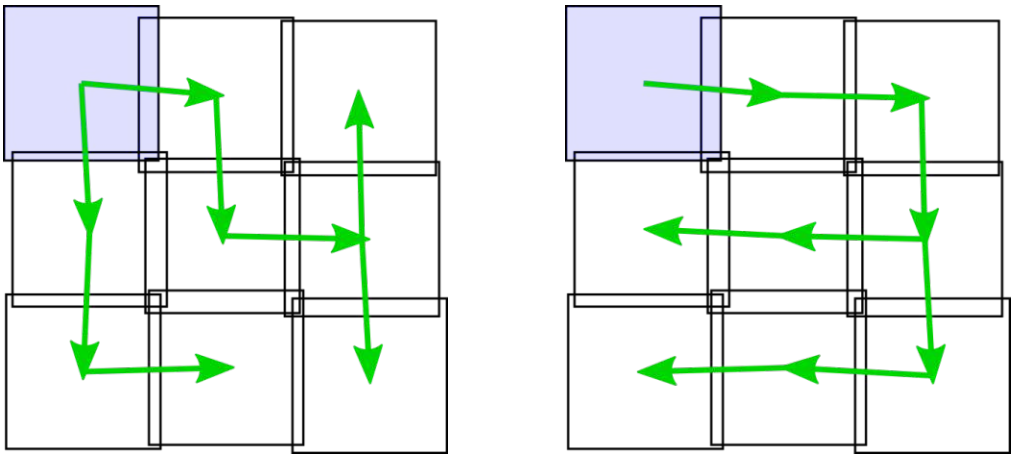
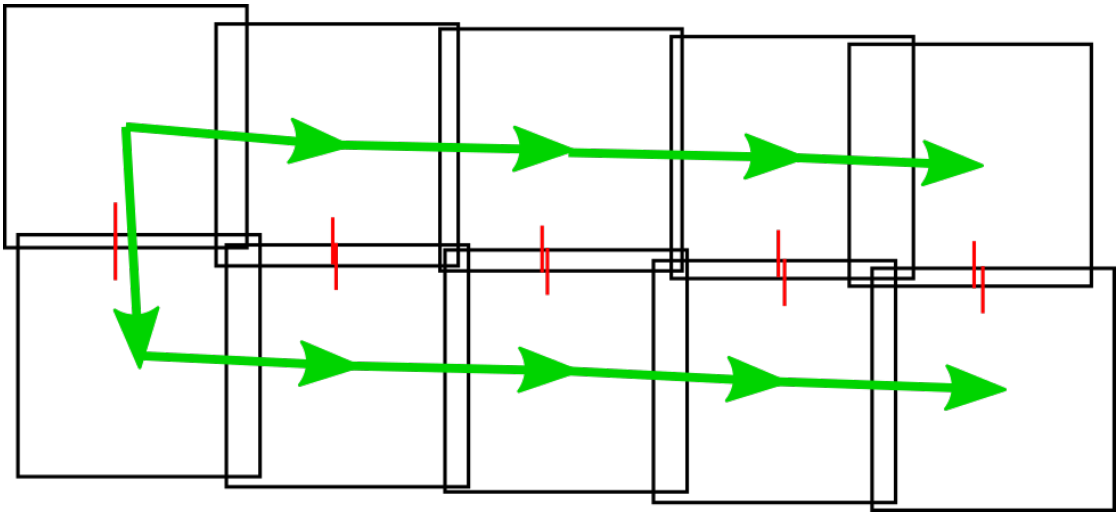
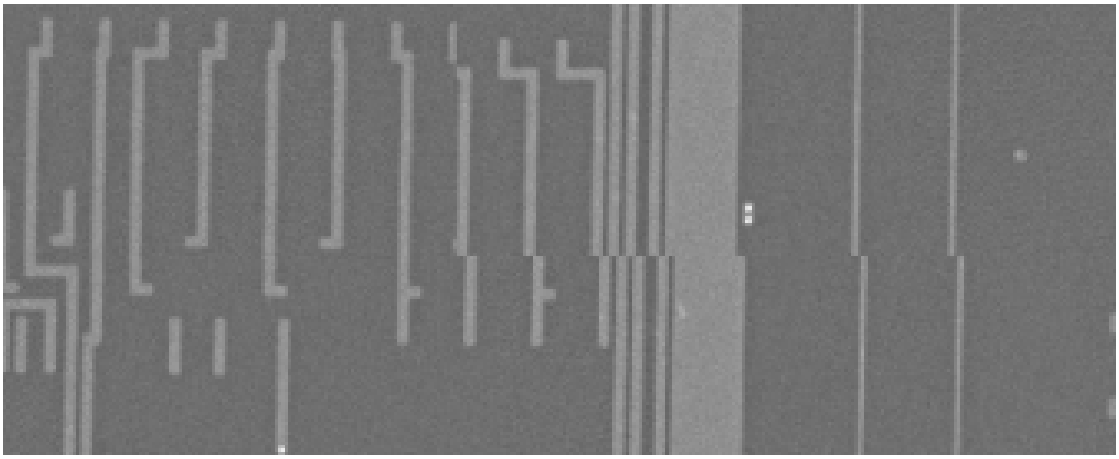


Figure 2.3: Examples of different stitching graphs in a 3x3 image grid. The start node is highlighted blue.



(a) Theoretical example of error accumulation: The stitching graph is shown with green arrows, while the accumulating error is shown as the offset between the red lines. The red lines should be horizontally aligned for a perfect fused image.



(b) Example of this type of error in one of the test sets.

Figure 2.4: Typical example for the accumulation of errors along stitching paths causing tearing in the fused image.

CHAPTER 3

State-of-the-Art Stitching Software

Two software categories are relevant for stitching scanning electron microscope images of integrated circuits: IC reverse engineering tools and image stitching software. However, software solutions in the first category may cover only a subset of steps in the reverse engineering process and may not include image stitching. Furthermore, these tools are often considered core intellectual property by the developing entity with neither free nor commercial licenses available to the public. Tools of the second category are plenty and readily available. State-of-the-art software in these two categories is discussed in the following sections.

3.1 IC Reverse Engineering Software

Multiple software solutions offer a broad spectrum of features ranging from automation of the imaging process to extracting high-level netlists [2, 11, 16]. However, several of these software tools are either no longer available or may have never been publicly available:

- Degate is a publicly available tool for reverse engineering ICs. Development of the original Degate software stopped in 2011, but recently a fork of this project was published on GitHub with the aim to modernize and eventually replace the original software [34, 35]. While Degate supports analysis of already stitched images, including tasks like gate recognition, grid definition, design rule checks, annotations, and netlist extraction, it does not include any stitching functionality at the time of writing.
- ChipJuice by Texplained reportedly contains multiple features for reverse engineering ICs such as layer stitching, via detection, track detection, and netlist generation

[36]. Licenses for ChipJuice, are supposedly available on request. Unfortunately, our license request remained unanswered. Hence, detailed evaluation was not possible, and it is unclear whether the product is actually available.

- GDS-X/GDS-eXtractor is documented in detail in an academic publication and reportedly supports layer reconstruction from multiple images, layer vectorization, error correction, and netlist extraction. However, it does not appear to be available for download or commercial licensing. At the time of writing, search results for this software only return the paper in which it is described [16].

- Similarly, ICWorks Extractor by Techinsights (formerly ChipWorks) is documented in several publications but is not publicly available [1, 2]. It reportedly supports features such as annotations, automated feature extraction, automated identification of on-chip structures, and netlist extraction.

- HAL is an open-source netlist reverse engineering tool [37]. It works directly on netlists and does not include any features to create netlists from images. This tool aims to aid the analyst in reverse engineering high-level functionality from a flat netlist.

3.2 Image Stitching Software

The problem of image stitching has been addressed many times in the previous decades, with the available software suitable to stitch SEM images of ICs roughly falling in one of two categories: Image stitchers for microscopy and panorama stitching software. The subsequent sections describe several publicly available image stitching applications that either claim to be suitable for stitching microscopy images or have been successfully used to stitch such images in prior research.

3.2.1 Image Stitchers for Microscopy

While the images used for this research clearly belong to this category, images from optical microscopes also fall into the same category. However, they differ in essential properties such as having color channels and possibly being out of focus due to the significantly reduced depth of field when compared to SEMs. On the other hand, SEM images are likely to have a worse signal-to-noise ratio (SNR), and the number of individual images to be fused into one large image may also be substantially greater due to higher magnification. Furthermore, distortion caused by charging effects only occurs in SEM images and poses additional challenges. Given these differences, stitching tools for optical microscopy may not deliver good results for SEM images. Since the available documentation does not discuss the ability of the tools to stitch SEM images, all tools listed in this chapter will be evaluated in Section 8.4. A notable commonality among the subsequently listed microscopy image stitchers is that they are all plugins for the open-source tool suite Fiji/ImageJ [38].

- **Grid/Collection Stitching:** This plugin was created by Preibisch et al. [33] and is included in the Fiji tool suite by default. It utilizes cross-correlation to match image pairs and a global optimization algorithm to compose the final image. Unlike many similar tools, this software is capable of stitching 2D and 3D images. This plugin requires knowledge of the overlap parameter, which specifies the expected overlap between adjacent tiles.

- **BigStitcher:** A Fiji plugin that succeeds the previously mentioned Grid/Collection Stitching plugin and has been published shortly after the algorithms introduced in this work were completed. Like the Grid/Collection Stitching plugin, it was developed by Preibisch et al. [39]. It is optimized for microscopy images of biological research and the published description, especially of its new global optimization algorithm, appears promising.

- **Microscopy Image Stitching Tool (MIST):** This Fiji plugin was published by Chalfoun et al. around the same time the tools and algorithms for this work were developed [32, 40, 41]. It follows similar ideas in principle: Images are matched pairwise, overlap and other parameters can be detected automatically. A model of the stage movement is inferred from the computed offsets, and subsequently, a minimum spanning tree is generated that follows the best stitching results in combination with the most likely tile position deduced from the stage movement model.

3.2.2 Panorama Stitching Software

Panorama stitching software is usually used to stitch digital camera images, which have different properties compared to microscope images, such as a high degree of lens distortion and requiring mapping using a 3-dimensional camera movement model. Nevertheless, these tools can provide useful results for sections of IC images with high SNR and clearly visible features in the overlapping image areas. The subsequently listed tools are panorama stitching tools that have been successfully used for research purposes in other fields [42].

- **Microsoft Image Composite Editor (MS-ICE):** This software was published by Microsoft and, while proprietary, is free to use. It is optimized for panoramas, but the settings support camera motion models appropriate for stitching microscopic images. No intermediate output is generated, and besides changing projection type and cropping, no manual correction is possible via the user interface. Manual correction of stitching errors may be possible in theory by changing the .spj file created by MS-ICE, but due to the transformation matrices in the file, stored data would have to be converted into human-readable format first. Wójcicka et al. successfully used this tool to stitch optical microscopy images of metallic minerals for the purpose of surface analysis [42].

- **PTGui Pro:** PTGui Pro is a closed source panorama stitching tool with a free fully-featured trial version available (version 11.16 at the time of writing). A guide on how to stitch microscopy images with PTGui was published by Dr. Georg von Arx at the Swiss Federal Institute for Forest, Snow, and Landscape Research WSL [43].
- **Teorex Photostitcher:** This proprietary software is available for multiple platforms. In addition to the commercially licensed version, a free trial version is available. Teorex advertises its photo stitching tool's ability to stitch microscope images, even though this is not the main focus of the application [44].

CHAPTER 4

Image Registration Algorithms

The first step to creating an automated image stitching tool is determining which image registration algorithm delivers the best results for noisy, distorted images of integrated circuits. Image registration in the context of stitching is used to find the offset between two adjacent, overlapping images. This processing step is necessary because the microscope stage is not moving in pixel-exact offsets between taking images. Furthermore, distortion also changes the offset and may even make it impossible to find an optimal offset.

The image registration algorithms considered for evaluation fall into two categories: The first category consists of feature matching algorithms that search for keypoints in each image and then evaluate which of those keypoints also appear on the other image to deduce how the images fit together. The other category consists of template matching algorithms, which are not limited to two-dimensional images but can be used for signal analysis in multiple dimensions. These template matching algorithms assign a value to every possible offset, which corresponds to the similarity of the overlapping areas of the images. This value is subsequently called the match score. The subsequent sections describe the selected feature matching and template matching algorithms in detail.

4.1 Feature Detection Algorithms

Many feature detection algorithms have been published that may be suitable to stitch scanning electron microscope images. To limit the scope of the subsequent evaluation, Scale-Invariant Feature Transform (SIFT) and Oriented FAST and Rotated BRIEF Algorithm (ORB) have been singled out as candidates for evaluation. The reason for this selection lies in the results of multiple publications, which show that for a wide range of use cases, SIFT often delivers the best accuracy while ORB tends to require less computational resources [45–49]. These two algorithms are described in detail in the following sections.

4.1.1 Scale-Invariant Feature Transform (SIFT)

SIFT was published in 1999 by David Lowe [50]. It improves upon previous feature detection approaches by providing features that are invariant to image scaling, translation, and rotation and partially invariant to illumination changes, affine, and 3D projection. The keypoint-computation follows three steps:

1. Keypoint candidates are determined by finding minima and maxima of a difference-of-Gaussian function in scale-space. The first step in this process is to convolve the input image with the Gaussian function using $\sigma = \sqrt{2}$ to produce image A. This step is then repeated, and the resulting image B is subtracted from image A to compute the first layer of the scale-space pyramid. The second layer of the pyramid is computed by resampling image B with 1.5x bilinear interpolation. Only keypoints that are detected on each layer of the pyramid will be considered in the following steps.

2. A canonical orientation is assigned to each keypoint, as determined by a histogram of local image gradient orientations.

3. A local image description is generated for each keypoint. This description characterizes the surrounding of a keypoint in a manner invariant to its location, scale, and orientation. It is composed of orientation planes that specify the image gradient magnitude of each orientation for every layer of the pyramid.

The resulting keypoints and keypoint descriptions must then be matched to find corresponding keypoints in the two images. To do so, Lowe introduced the best-bin-first indexing algorithm. However, OpenCV's Brute-Force K-Nearest-Neighbors matching algorithm was used for subsequent evaluations instead. It matches each descriptor of the first image with all descriptors of the second image and returns the k best matches, resulting in slightly better results at the cost of higher computational complexity.

To compute the offset between two overlapping images, the matched keypoint pairs are first grouped by their offsets. The largest group of agreeing keypoints is then chosen as the offset most likely to be correct. The number of keypoint pairs in the largest group and the difference in size to the second-largest group can be used to quantify confidence in the correctness of the computed offset. Figure 4.1 shows the matching keypoints discovered by the SIFT algorithm.

4.1.2 Oriented FAST and Rotated BRIEF Algorithm (ORB)

Features from Accelerated Segment Test (FAST) is a corner detection method used to find keypoints in an image [51]. On the other hand, Binary Robust Independent Elementary Features (BRIEF) is an algorithm used to describe the area surrounding such keypoints [52]. The ORB algorithm, published in 2011 by Ethan Rublee et al. [53], combines slightly modified versions of the FAST keypoint detector and BRIEF descriptors to

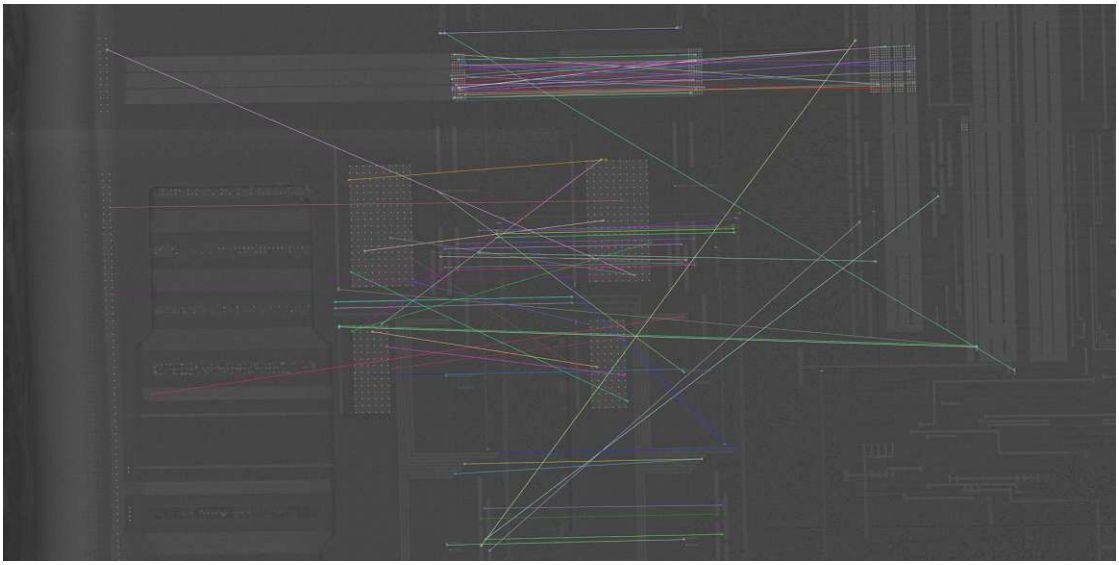


Figure 4.1: Matched SIFT-Keypoints in two adjacent images of test set2

provide accuracy similar to SIFT with computational requirements suitable for real-time applications. The algorithm is composed as follows:

- Detection using oFAST: Since FAST Keypoints do not possess an orientation component, it is extended to create oFAST. Due to its good performance, keypoints are detected using FAST with parameter K set to 9 (FAST-9). These keypoints are then extended with a Harris corner measure to order the keypoints, so the N best keypoints may be chosen. To mitigate the fact that FAST does not produce multi-scale features, the keypoints are computed for each level of a scale pyramid of the image.
- The orientation of each keypoint is determined by computing the intensity centroid, which assumes that a corner's intensity is offset from its center [54].
- Feature point description using rBRIEF: Since BRIEF is not rotation-invariant, it is "steered" according to the previously computed orientation in increments of 12 degrees to make it less sensitive to rotation. The resulting algorithm "steered BRIEF" was then used on a training set of keypoints to determine 256 binary-tests that are uncorrelated and have their means close to 0.5, resulting in the rBRIEF algorithm. This algorithm is used to convert the previously detected keypoints into binary feature vectors to describe the features.

The resulting keypoints and descriptors were matched and then grouped in the same way as for SIFT, using OpenCV's Brute-Force K-Nearest Neighbor matching algorithm and grouping the resulting matches by their x and y offsets to determine the most likely

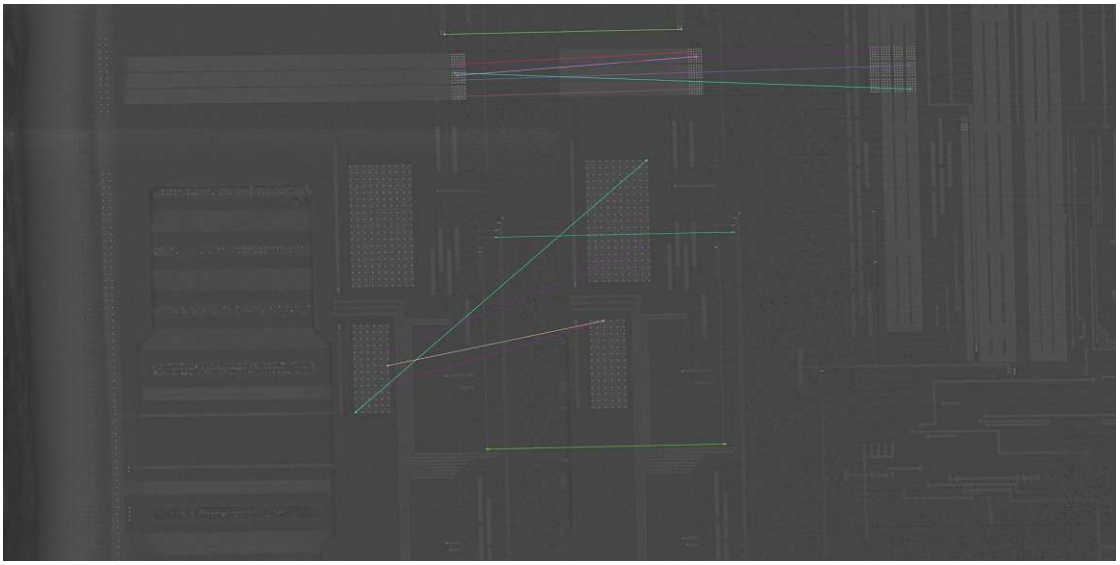


Figure 4.2: Matched ORB keypoints in two adjacent images of test set2

offset of two overlapping images. Figure 4.2 shows the matching keypoints discovered by the ORB algorithm.

4.2 Template Matching algorithms

The term template matching only partially describes the subsequently listed algorithms. The name of this category was chosen because most of the implementations used for evaluation are contained in the template matching module of OpenCV. However, the chosen algorithms are used in various situations ranging from other applications within the image processing domain, such as image registration, to less related domains such as signal processing in cryptanalysis or neurophysiology [55, 56]. The workflow for the subsequently listed algorithms is the same: For each possible offset, the template matching algorithm is given the overlapping areas of the images as input. It then returns a value called match score, which quantifies the similarity of these overlapping image sections. Typically the offset corresponding to the best match score is chosen automatically from the resulting match score matrix. Alternatively, the results can be visualized as a greyscale image for manual inspection. The following sections describe the template matching algorithms in detail.

4.2.1 Normed Sum of Squared Differences

The normed sum of squared differences, named SqDiff_Normed in OpenCV, is a comparatively simple measure of similarity for signals closely related to the residual sum of squares (RSS) in statistics. The underlying mathematical equation is shown in Equation (4.1), where $T(x, y)$ denotes the pixel value at coordinates (x, y) in the template and with

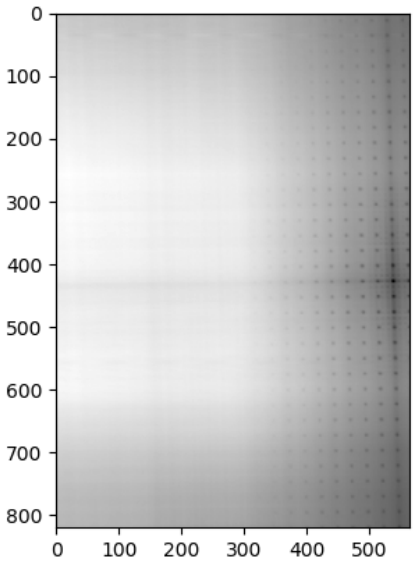


Figure 4.3: SqDiff_Normed output for the same two overlapping images shown in Section 4.1. The most probable offset according to this algorithm is at location (536, 427) with a match score of 0.0135331926867

$I(x, y)$ being the pixel value at coordinates (x, y) in the image. $R(x, y)$ is the resulting match score.

$$R(x,y) = \frac{\sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}} \tag{4.1}$$

In this equation, a low value for $R(x, y)$ indicates a good match at a given offset (x, y). To find the optimal overlap according to SqDiff within two overlapping images, $R(x, y)$ is computed for all possible offsets, and the minimum is chosen as the most probable offset. OpenCV also includes a non-normed version of this algorithm that consists only of the numerator in Equation (4.1). The most notable comparative advantage of normed SqDiff versus the non-normed version is that the value $R(x, y)$ of normed SqDiff depends only on the similarity between the compared image areas and not on the size of the compared area. Figure 4.3 shows the results for two overlapping images, with darker pixels corresponding to better matches.

4.2.2 Normed Cross-Correlation

Cross-correlation is probably most widely known for its use in the domain of signal processing, where it is used to determine the similarity of two signals at various time-lags.

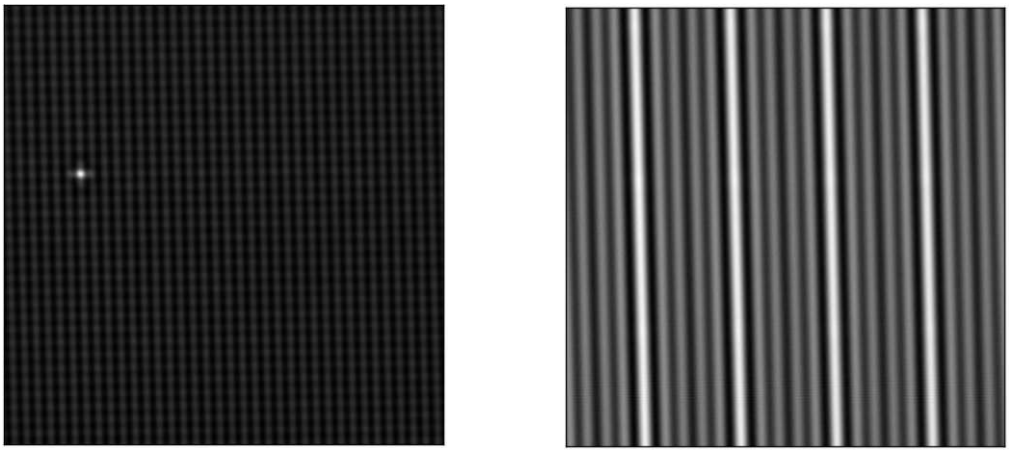


Figure 4.4: Result of normed cross-correlation of two images of test set 2 with a small number of repetitive features (left) vs a large number of repetitive features (right)

Mathematically it corresponds to the sliding dot product and to multiplication in the frequency domain. Normed cross-correlation is listed in Equation (4.2), where $T(x, y)$ denotes the pixel value at coordinates (x, y) in the template, with $I(x, y)$ being the pixel value at coordinates (x, y) in the image and $R(x, y)$ as the resulting match score. A larger value of $R(x, y)$ corresponds to a better match at the given offset (x, y).

$$R(x,y) = \frac{\sum_{x',y'} \left(T(x',y') \cdot I(x+x',y+y')\right)}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}} \tag{4.2}$$

Cross-correlation is quite resistant to noise. Peaks in the signals contribute to the sliding window integral most when they are perfectly aligned, while non-aligned peaks caused by noise are attenuated by the sliding dot product. On the other hand, distortions that change the alignment of the peaks of the two signals reduce the quality of the result, which may pose problems for stitching distorted SEM images.

As previously mentioned, cross-correlation is used in signal processing to find the location of a signal along the time axis. When using this algorithm for image registration or template matching, the result is a two-dimensional matrix, in which values indicate the similarity of the two images at the corresponding location. The matrices in Figure 4.4 show that it is harder to find the correct offset with cross-correlation when the signal is repetitive, as the peaks in the signal will match almost perfectly at several locations. In such repetitive images, even manual matching may not be suitable to determine the correct offset. However, due to the way cross-correlation matches the similarity of the whole image section as opposed to only matching individual features of the image, it is likely to produce a good result in cases where no one correct location can be determined. While the result, in this case, will often look as good as results from manual matching

when only considering the two overlapping images, an offset error introduced this way may propagate to neighboring images during global stitching.

A downside of cross-correlation when matching greyscale images such as SEM images is that the signals are represented by positive integer values. This means that negative peaks in the signal lack the ability to cancel out noise efficiently since their lowest possible value is not a negative value but 0. For this reason, better results were achieved with cross-covariance, where results are improved by subtracting the mean from each image before computing cross-correlation, as discussed in Section 4.2.3.

4.2.3 Normed Cross-Covariance

Normed cross-covariance is very similar to normed cross-correlation, with Equation (4.2) and Equation (4.3a) being essentially identical. The difference is that for each signal, the mean of the signal is subtracted before computing cross-correlation for a given sliding window, as Equation (4.3b) and Equation (4.3c) show, with w and h as width and height of the template. This improves noise attenuation in signals that would otherwise have a non-zero mean, since negative spikes will cancel out more effectively with positive spikes when the signals are not aligned perfectly.

$$R(x,y) = \frac{\sum_{x',y'} \left(T'(x',y') \cdot I'(x+x',y+y')\right)}{\sqrt{\sum_{x',y'} T'(x',y')^2 \cdot \sum_{x',y'} I'(x+x',y+y')^2}} \tag{4.3a}$$

$$T'(x',y') = T(x',y') - 1/(w \cdot h) \cdot \sum_{x'',y''} T(x'',y'') \tag{4.3b}$$

$$I'(x+x',y+y') = I(x+x',y+y') - 1/(w \cdot h) \cdot \sum_{x'',y''} I(x+x'',y+y'') \tag{4.3c}$$

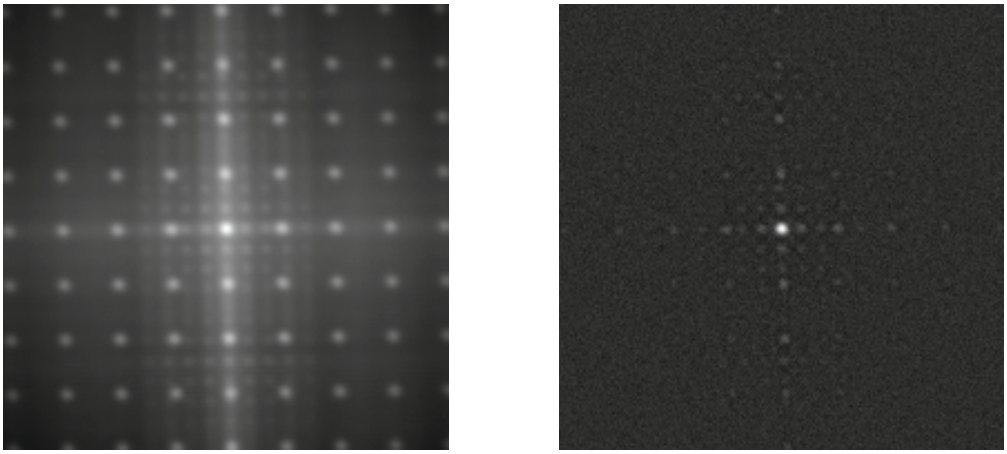
4.2.4 Phase Correlation

Phase correlation is very similar to cross-correlation when calculated via the frequency domain. Cross-correlation via the frequency domain is shown in Equation (4.4), while Equation (4.5) shows the definition of phase correlation, with $\circ$ as the entry-wise product, $\mathcal{F}$ as Fourier transform, $\mathcal{F}^{-1}$ as inverse Fourier transform and $*$ denoting the complex conjugate.

$$R = \mathcal{F}^{-1}\left(\mathcal{F}(I) \circ \mathcal{F}(T)^*\right) \tag{4.4}$$

$$R = \mathcal{F}^{-1}\left(\frac{\mathcal{F}(I) \circ \mathcal{F}(T)^*}{|\mathcal{F}(I) \circ \mathcal{F}(T)^*|}\right) \tag{4.5}$$

The main difference is the added element-wise normalization in the frequency domain, which causes clear spikes after inverse Fourier transformation where the similarity of



(a) Normalized cross-correlation (b) Phase correlation

Figure 4.5: Comparison of cross-correlation and phase correlation results for the same sections of images of test set 2.

the signals is maximized. Figure 4.5 shows a comparison of the results of normalized cross-correlation and phase correlation.

An additional benefit of phase correlation is that several computationally cheap algorithmic extensions are available to achieve subpixel accuracy due to the typical shape of the resulting spike at the best matching offset. OpenCV's implementation, which was used for evaluation, uses a 5x5 centroid at the peak for subpixel accuracy.

CHAPTER 5

Evaluation of Image Registration Algorithms

In this chapter, the previously introduced image registration algorithms are evaluated using two image sets. To establish a ground truth for the evaluation, the correct offsets for all horizontal pairs of overlapping image pairs were manually determined. Image pairs where no correct offset could be determined manually were excluded from the evaluation. The following sections describe the image sets and their preparation, the scope and methodology of the evaluation, and the evaluation results.

5.1 Image Sets

Two image sets from previous IC reverse engineering projects were chosen to compare the performance of the algorithms for realistic use cases. Test set 1 shows the top layer of a decapsulated chip. The 441 images of test set 1 are arranged in 21 columns and 21 rows, and the individual images show high contrast, low noise, and minimal distortion, making it comparatively simple to stitch. Test set 2 consists of 440 images in 20 columns and 22 rows. This image set is a typical worst-case scenario for stitching: The images show an IC with several layers removed. Part of the insulating layer between the next visible layer and the previously removed layer is still on the chip. This makes the vias clearly visible in the form of bright spots, but details of the next layer are only barely visible. The individual images are very noisy with low contrast and some distortion near the eastern and southern border of the chip. Figure 5.1 shows images of both test sets to illustrate the differences in contrast and noise.

The restriction of using just two test sets stems from the large amount of time required to determine correct offsets manually. For the sake of meaningful comparison, it was deemed more important to include all areas of the imaged ICs to account for common

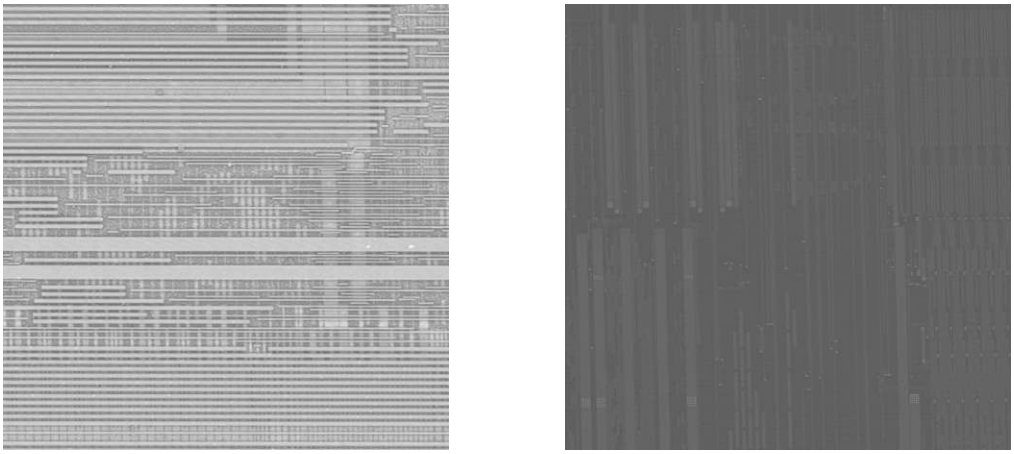


Figure 5.1: Sample images from both test sets

difficult-to-stitch areas such as memory regions than to include images from many different layers and projects, which vary in noise, contrast, and distortion. To compensate for this shortfall, the test sets were chosen such that test set 1 represents the level of detail and contrast which is typically required for the analysis of a single layer of an IC in the process of reverse engineering its logic. In contrast, test set 2 represents the worst-case scenario of using a somewhat dated SEM to take images between layers of the IC to determine the location of vias. The evaluation was limited to horizontal image pairs of the test sets due to the time-consuming nature of manual offset determination.

5.1.1 Removal of Unsuitable Images

For each image set, the correct pairwise offsets of horizontal pairs were determined manually. Due to reasons such as distortion, high noise, low contrast, or lack of significant features in the overlapping area, some image offsets could not be determined manually. The category of these images was named “indeterminate” and the affected image pairs were not taken into account during this evaluation. However, these images still matter for global stitching.

Test set 1 contains no such image pair, while 62 of 420 horizontal offsets in test set 2 were categorized as indeterminate. Figure 5.2 shows an image pair where the only features in the overlapping area are straight lines, allowing for arbitrary placement along those lines without discernible difference in outcome.

Some image pairs in these categories have subjectively correct offsets, but it was not possible to provide the algorithms with additional information concerning how to match images in these special cases. Examples of such a case were images along the southern border of the IC, where the upper 10% of the image area shows the IC border and the lower 90% show the blurry surface of the stage. In several cases, the translational difference of the stage surface between two images did not match the translational

Figure 5.2: Example of an overlapping image pair where manual matching failed due to the lack of significant features in the overlapping area

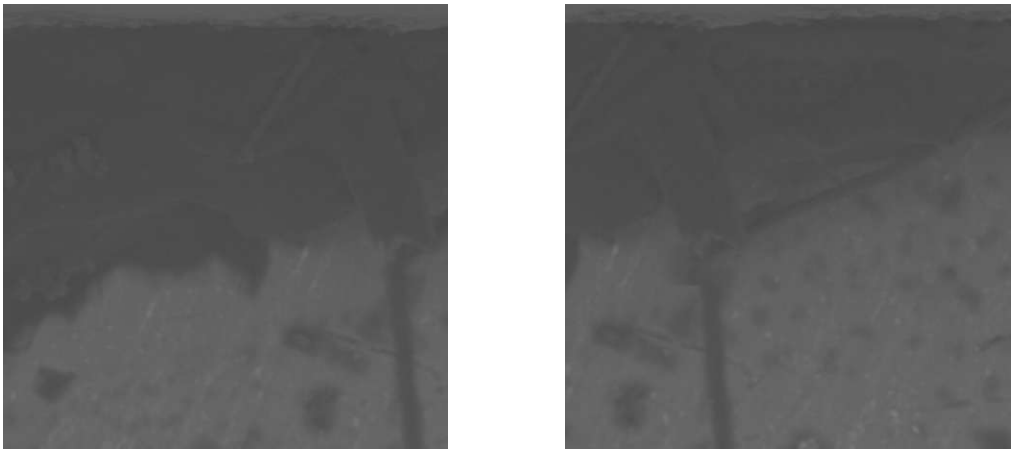


Figure 5.3: Example of an overlapping image pair where automated matching fails due to the area of interest (upper 10% of the images) having a different translational offset than the lower 90% that depict the stage surface of the microscope

difference of the IC surface, resulting in a parallax effect. Figure 5.3 shows such an image pair where an acceptable solution would mean ignoring the mismatch of the majority of the images to focus on the best match for the IC border in the top 10% of the image. However, since these images contain no significant features for the use case of reverse engineering, their accurate placement is not considered essential for evaluation.

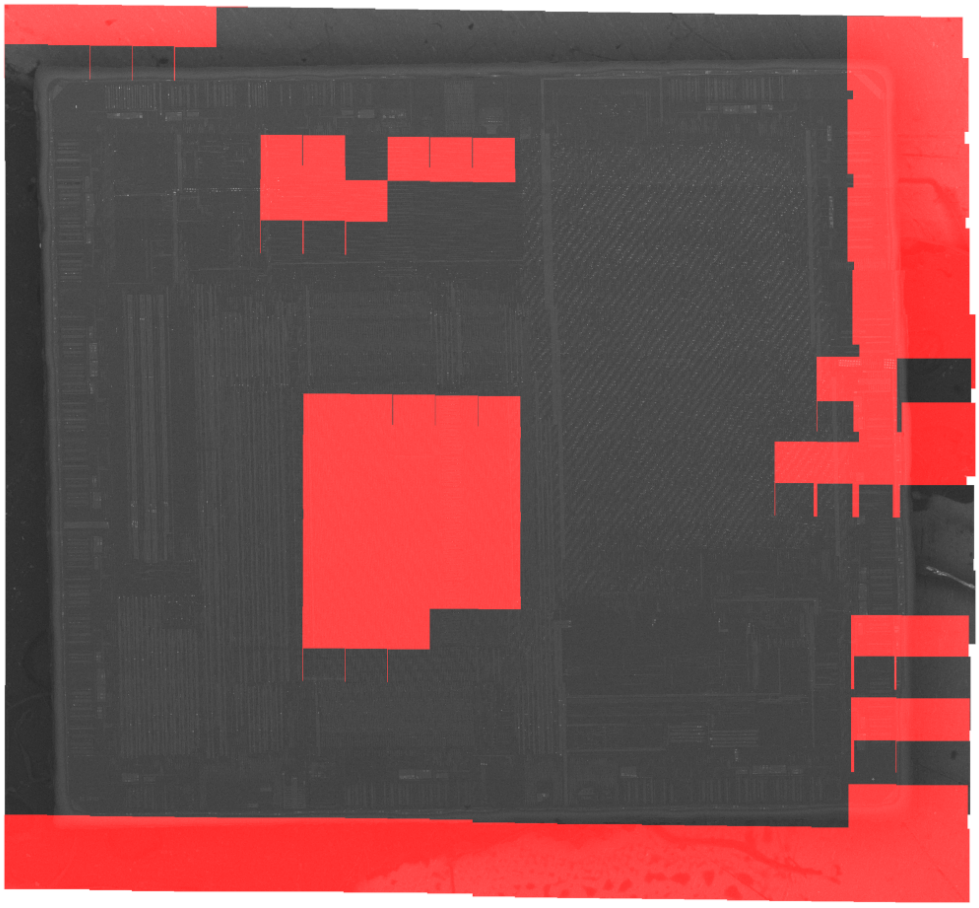


Figure 5.4: Fused test set 2 images with indeterminate images highlighted red.

Figure 5.4 shows all images in test set 2 whose offsets were deemed indeterminate. The areas can be separated into three categories:

- At the north-western corner, as well as along the southern border of the IC, the distance along the z-axis between the IC and the stage caused differing translational offsets for sample and background
- Two areas on the chip contained very repetitive structures consisting primarily of horizontal or vertical lines and no features to allow for exact stitching
- Images along the eastern border of the IC where charging effects or erratic stage movement caused distortions

5.2 Methodology

For each algorithm, image offsets were computed for all horizontal image pairs of the image sets. If the deviation of the result to the correct result was at most 1 pixel for each axis, the result was deemed correct. This margin of error was considered suitable to account for possible differences in rounding. For algorithms that required parameters such as the expected overlap, a subset of relevant parameters was selected manually, which were then tested using a wide range of values.

With the SIFT algorithm, a slightly different approach for evaluation was necessary: Its comparatively slow execution time meant that only 25 parameter configurations could be evaluated within the scope of this research. These parameter configurations were chosen in regular intervals around manually determined suitable parameters. The SIFT parameters were also evaluated using much smaller subsets of the original test sets. These subsets consisted of 100 randomly chosen image pairs for each image set. Evaluation of this algorithm still took significantly longer than for all other algorithms combined. While a performance difference between the two feature detection algorithms SIFT and ORB was to be expected, it is unclear why SIFT was so much slower than ORB. One possible reason may be that, like ORB, it was not using random sample consensus (RANSAC), but OpenCV's brute force matcher, which meant higher numbers of keypoints resulted in much longer execution times. While further optimization would likely lead to significantly better performance, the results were not promising enough to justify the required engineering effort.

Each algorithm was evaluated in a virtual machine (VM) running Ubuntu Linux 18.04. The VM host was equipped with an Intel Core i7-7820HQ, but without a dedicated graphics card. The VM was assigned 8 GB of RAM. The host machine was otherwise idle for the duration of the evaluation. Algorithm implementations of the open-source OpenCV library were used. Better results may be achieved using implementations of the tested algorithms that utilize GPUs to accelerate computation. To speed up computation, execution was parallelized to utilize all 8 CPU cores. Only one algorithm and test set were evaluated at any given time.

5.2.1 Evaluated Parameters

Since the OpenCV-based implementations of all template matching algorithms except phase correlation used the same function signature, they required the same two parameters: expected overlap and minimal overlap. These two parameters determine the size of the image segments that are compared using template matching, such as the right border of the left image and the left border of the right image. If the difference between these parameters is large, the sections to be compared are larger, leading to slower execution times. On the other hand, phase correlation was part of a different OpenCV module and only required one parameter specifying the expected overlap. Like before, the expected overlap determines the size of the compared image sections.

	Test Set 1			Test Set 2		
Algorithm	**Score**	**t_{min}**	**t_{max}**	**Score**	**t_{min}**	**t_{max}**
pcorr	100.00% (420)	0.10	0.18	99.44% (356)	0.17	0.21
ccoeff	100.00% (420)	0.19	0.20	98.60% (353)	0.38	0.38
sift	100.00% (100)	37.49	306.15	95.00% (95)	56.31	57.07
sqdiff	100.00% (420)	0.18	0.20	90.78% (325)	0.51	0.73
ccorr	100.00% (420)	0.19	0.20	90.50% (324)	0.48	0.60
orb	83.57% (351)	0.31	0.39	76.82% (275)	0.19	0.19

Table 5.1: Best scores for each tested algorithm. Score is the number of correctly determined offsets. Average execution time was computed for each parameter set. t_{min} and t_{max} are minimum and maximum average execution time of parameter sets that produced optimal results, measured in seconds. Unlike the other algorithms, SIFT's maximum score is 100 due to the reduced image set size.

For SIFT, the parameters contrastThreshold and edgeThreshold were evaluated across a range of values. These parameters are expected to substantially impact the number of detected keypoints and, therefore, the accuracy of the result.

OpenCV's ORB implementation exposes many parameters, some of which depend on each other. For evaluation, the independent parameters WTA_K and fastThreshold were tested across a range of values. The remaining parameters were set to their default values.

5.3 Evaluation Results

The two clear winners of this evaluation are phase correlation and cross-covariance, with phase correlation yielding the best results while also performing faster than the other tested algorithms. Result quality of SIFT comes somewhat close, but the tested OpenCV implementation was orders of magnitude slower. All algorithms except ORB yielded 100% correct results for the first test set. Results for test set 2 varied between 99.44% and 76.82%. Table 5.1 shows the results of the best parameters for each algorithm, where the score is the number of correctly determined offsets. Due to the large number of tests run, the corresponding detailed results for each tested parameter combination are listed in Appendix A. The extent of RAM usage was not explicitly tested. However, SIFT was the only algorithm for which the evaluation could not be run in parallel to fully utilize each CPU core, as the machine stopped execution in this case after running out of memory.

CHAPTER 6

Automated Parameter Determination

The previously discussed algorithms mainly rely on a parameter that defines the expected overlap of the two input images. As the algorithm evaluation showed, the quality of results depends on the accuracy of this parameter. This chapter aims to identify solutions for the automated determination of optimal parameters.

All previously discussed algorithms provide feedback about the degree of confidence that can be assigned to the computed offset. For feature detection algorithms, the number of identified keypoints and how well the keypoint matches agree with the movement model provide such information. Template matching algorithms assign numerical values that represent the computed similarity of the overlapping image areas. These values are not necessarily comparable between different image pairs but may be suitable to evaluate multiple parameters for the same two overlapping images.

This chapter explores how the match score of the two best algorithms determined by the evaluation in Chapter 5 may be used to deduce the overlap parameters. This previous evaluation also provides a list of optimal parameters for each algorithm and test set within the tested parameter range, which will be used as a reference to measure the success rate for parameter determination algorithms.

Test Set 1			
Phase correlation		Cross-covariance	
Parameter	AP1-Score	Parameter	AP1-Score
0.20	**420 (100%)**	**0.20**	**361 (86%)**
		0.25	59 (14%)

Table 6.1: Results of AutoPar1 for test set 1, best parameters as manually determined in previous evaluation are bold. Parameters with score 0 have been removed. AP1-Score represents the number of times a parameter led to the highest match score, as computed by AutoPar1

Test Set 2			
Phase correlation		Cross-covariance	
Parameter	AP1-Score	Parameter	AP1-Score
0.35	**375 (89.3%)**	0.30	171 (40.7%)
0.40	22 (5.2%)	0.40	139 (33.1%)
0.30	7 (1.7%)	**0.35**	**84 (20.0%)**
0.20	6 (1.4%)	0.45	22 (5.2%)
0.25	4 (1.0%)	0.50	2 (0.5%)
0.45	4 (1.0%)	0.25	1 (0.2%)
0.50	2 (0.5%)	0.20	1 (0.2%)

Table 6.2: Results of AutoPar1 for test set 2, best parameters as manually determined in previous evaluation are bold. AP1-Score represents the number of times a parameter led to the highest match score, as computed by AutoPar1

6.1 Parameter Determination Using Match Score Sum

Both cross-covariance and phase correlation return a numerical value indicating the similarity of the overlapping image area, the match score. The first approach to find the optimal parameter for a given test set using this match score, called AutoPar1, essentially consists of the following two phases:

1. For each image pair, compute all offsets and their match scores for each parameter in the selected parameter range
2. To rate each parameter, sum up the number of times it yielded the best match score among all parameters.

6.2 Parameter Determination Using Match Score Mean

As the results in Table 6.1 and Table 6.2 show, this approach worked well for phase correlation but not for cross-covariance. The optimal parameter for cross-covariance, an

Test Set 1			
Phase correlation		Cross-covariance	
Parameter	AP2-Score	Parameter	AP2-Score
0.20	**0.85596**	**0.20**	**0.86889**
0.25	**0.67217**	0.25	0.86029
0.30	**0.54878**	0.30	0.79741
0.35	**0.47161**	0.35	0.75464
0.40	0.40655	0.40	0.72152
0.45	0.36134	0.45	0.60229
0.50	0.32566	0.50	0.51346
0.55	0.29588	0.55	0.49290
0.60	0.27329	0.60	0.47530

Table 6.3: Results of AutoPar2 for test set 1, best parameters as manually determined in previous evaluation are bold. AP2-Score is the mean match score computed by AutoPar2.

Test Set 2			
Phase correlation		Cross-covariance	
Parameter	AP2-Score	Parameter	AP2-Score
0.35	**0.46983**	**0.35**	**0.40090**
0.40	**0.41096**	0.40	0.40065
0.30	0.36323	0.30	0.39312
0.45	0.36127	0.45	0.39062
0.50	0.32317	0.50	0.37328
0.55	0.29222	0.55	0.35873
0.60	0.26786	0.60	0.34635
0.25	0.24623	0.25	0.19359
0.20	0.09788	0.20	0.17078

Table 6.4: Results of AutoPar2 for Testset2, best parameters as manually determined in previous evaluation are bold. AP2-Score is the mean match score computed by AutoPar2.

expected overlap of 0.35, only ranked third. Better results were achieved by modifying phase 2 in AutoPar1 to compute the mean match score for each parameter across all image pairs of the image set and then choose the parameter with the highest match score mean. This modified algorithm was named AutoPar2.

Table 6.3 and Table 6.4 show that AutoPar2 returned correct results for both algorithms when computed across all image pairs of a given test set. Since computation across whole image sets is computationally expensive, a random sample of the image set may be used to compute optimal parameters. Figure 6.1 shows the reliability of this approach for multiple sample sizes. For test set 1, this algorithm works well for both cross-covariance and phase correlation, requiring only a sample size of 1.2% of the test set to yield the

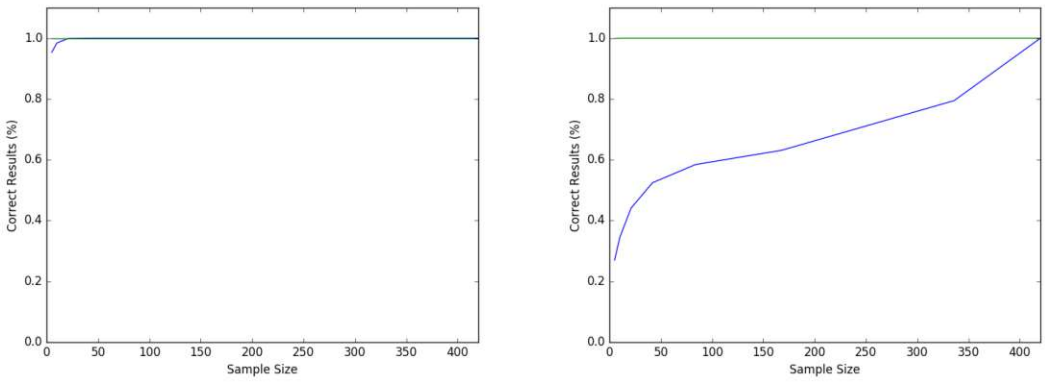


Figure 6.1: Results of AutoPar2 for test set 1 (left) and test set 2 (right) with 100000 random sample sets per sample size for algorithms phase correlation (green) and cross-covariance (blue).

correct parameter over 95% of the time for both algorithms. Test set 2, however, shows significantly different results. While for phase correlation, the algorithm determines the correct parameters over 99.8% of the time with sample sizes as small as 1.2% of the image set, finding the correct parameter for cross-covariance requires a much larger sample size. Even with a sample size of 80%, the algorithm still only provides correct results 79% of the time.

6.3 Parameter Determination for Cross-Covariance Using Phase Correlation

From these results, it follows that it should be possible to find optimal parameters for cross-covariance with a smaller random sample by first finding the optimal parameter for phase correlation and in a second step to use the computed (and likely to be correct) offsets to find cross-covariance parameters that yield the same offsets. One could argue that since the parameters for both algorithms specify the expected overlap, they should have the same value, but since algorithm implementations may differ in details such as padding, it is not obvious that they must match exactly. Similarly, it is not trivial to derive the optimal parameter from the computed offset values, as these offset values follow a specific distribution according to the stage movement model. To avoid these issues, AutoPar3 selects parameters by matching the computed offsets with the offsets determined by phase correlation.

As Figure 6.2 shows, AutoPar3 leads to significantly better results than AutoPar2 for test set 2 but to worse results for test set 1. The reason for this counterintuitive result for test set 1 is that the algorithm has many “good” parameters that lead to the same

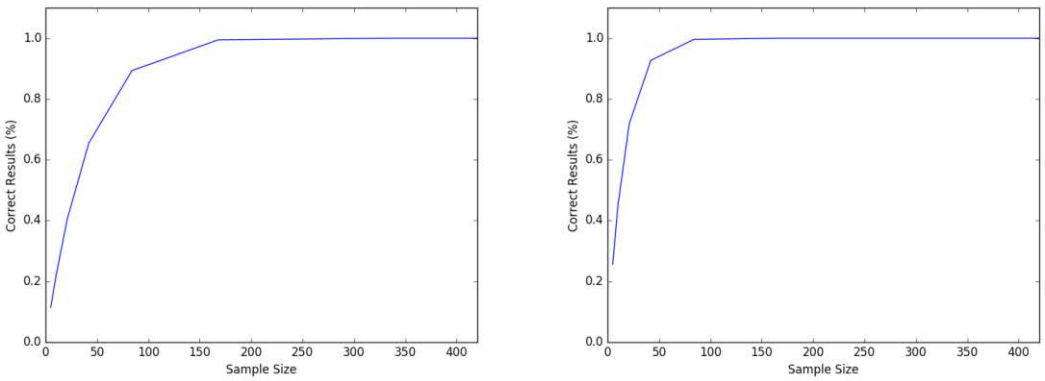


Figure 6.2: Results of AutoPar3 for test set 1 (left) and test set 2 (right) with 100000 random sample sets per sample size for cross-covariance.

correct offset for most of the image pairs, with the optimal parameter only making a difference in a small subset of the total image set. In the case of test set 1, the subset of image pairs where the optimal parameters yield better results than merely adequate parameters is sufficiently small that this approach is not practical compared to previous algorithms. For test set 2, however, the improvements in finding the correct parameters are significantly better, with a sample size of 10% (42 image pairs) already providing the correct result 92.7% of the time.

6.4 Conclusion for Parameter Determination

With the given test sets, it is possible to derive optimal parameters for phase correlation and cross-covariance automatically. Finding the optimal parameter for phase correlation is significantly simpler and can aid in finding the optimal parameters for other algorithms. It may be possible to improve automated parameter detection further, e.g. by using the fact that similar results of different algorithms may be more likely to be correct. This optimization is left to future work, as the results for phase correlation are sufficiently good for the purpose of this work.

CHAPTER 7

Global Stitching Algorithms

After computing the individual offsets between overlapping image pairs of the image set, the next step towards creating a fused image is to find the best combination of these offsets to compute each individual image's location in the fused image. To ease further discussion of the details of global stitching, the following terminology will be used going forward: The individual images composing the fused image will be called tiles. Offsets describe the computed offset between two neighboring tiles, while location specifies the absolute position of a tile in the fused image.

In an ideal setting, all offsets lead to the same final fused image. However, in real image sets, computing the tile locations along different paths will lead to different results. Among the reasons for this behavior are:

- Discretization: Between taking two images, the microscope stage usually will not move in exact pixel widths, leading to discretization errors that may sum up, e.g., when computing locations along a row of tiles.
- Distortion: There are several sources for image distortion in SEMs, as outlined in Section 2.2.3. This distortion causes problems when trying to determine correct offsets and when stitching the fused image. It can only be reverted by warping the images, which poses significant challenges as the exact transformation to undo the distortion is unknown.
- Incorrect offset: The image registration algorithm may not always find the correct offset. This mainly occurs in areas with only a small number of detectable features or areas with highly repetitive features such as horizontal or vertical lines.

Typically, a global stitching algorithm will start at one tile, assign it the location (0,0), and from there on compute the locations of the remaining tiles iteratively. The paths it

can follow to reach other tiles (and thus determine their location) may be viewed as a graph, with each tile as node and one edge to each immediate neighbor. If the result of the image registration algorithm also contains the level of confidence (match score), this number can be used as weight of the corresponding edge in the graph.

Once all locations of tiles have been computed, the next step is to add the minimum values to all locations so that all image coordinates are ≥ 0. Subsequently, the fused image can be created by allocating a large enough canvas and drawing each tile at the set location onto the canvas. The subsequent sections describe approaches for determining the tile locations required to create the fused image.

7.1 Version 1: Maximum Score

The first algorithm creates a minimum spanning tree by starting with one tile and then iteratively picking the best tile to add next, as defined by the highest phase correlation score.

1. Start at the tile closest to the center, determined by its grid index. Set its location to (0,0).
2. Create a list of all images with no location set, that have at least one neighbor with a set location.
3. Choose the tile from this list connected by the offset with the highest match score.
4. Compute this tile's location by combining the offset with the location of the neighbor image.
5. Repeat steps 2 to 4 until all locations are set.

This algorithm has at least two flaws. The first is that previously discussed discretization and distortion errors cause tearing along sections where tiles were added sequentially along the same axis. For example, if the best matches lead the algorithm to go right five tiles, go down one tile, and go left five tiles, then offset errors can accumulate horizontally in either row. Figure 7.1 shows an example of such horizontal tearing.

The second flaw is that in some cases, the best scoring offset may still be wrong. This algorithm relies on the highest scoring offset being correct. Figure 7.2 shows an example of such an error in a fused image. An attempt to improve this is described in the next section.

7.2 Version 2: Offset Statistics

The previously discussed second flaw of the maximum score algorithm is that the best scoring offset of a tile may be wrong in some cases. One way to improve this may

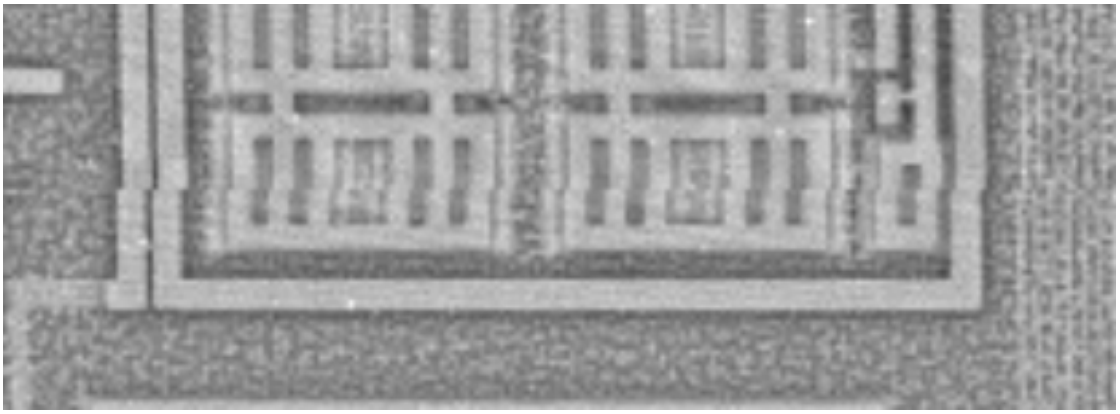


Figure 7.1: Offset error accumulation leading to visible tearing in fused image.

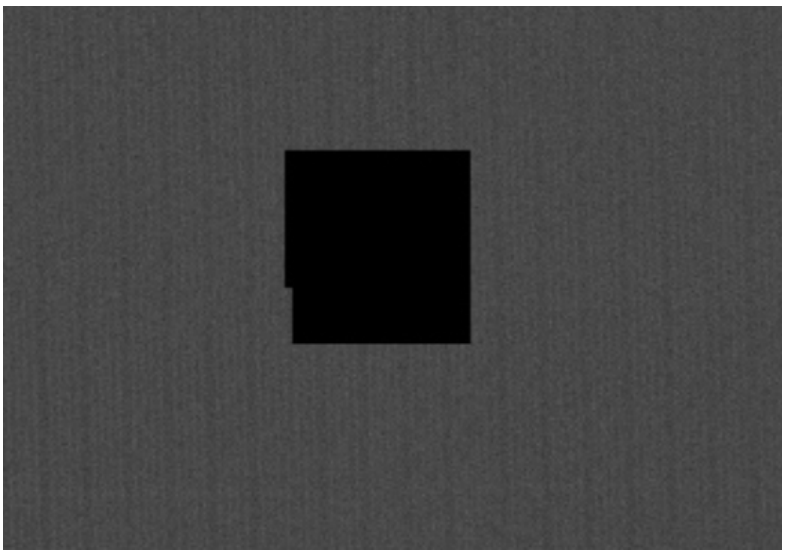


Figure 7.2: Tile misplacement due to a wrong offset having the highest score.

be to augment the match score with additional statistical information. In theory, the microscope stage should move the exact same distance between all image pairs for a given direction. Of course, in practice the underlying mechanisms of stage movement do not provide infinite accuracy, so that we would expect a normal distribution around the set distance, ideally with very low deviation. This also means that the greater the difference between computed offset and mean offset is, the less likely it is to be correct - assuming the stage is not damaged.

To test this hypothesis, histograms over all computed x and y offsets for all vertical and horizontal image pairs were computed. Figure 7.3 shows how the offsets (as computed using phase correlation) are distributed. Unlike test set 1, test set 2 has a wide range of outliers and higher deviation. If we assume that the stage mechanism of the utilized

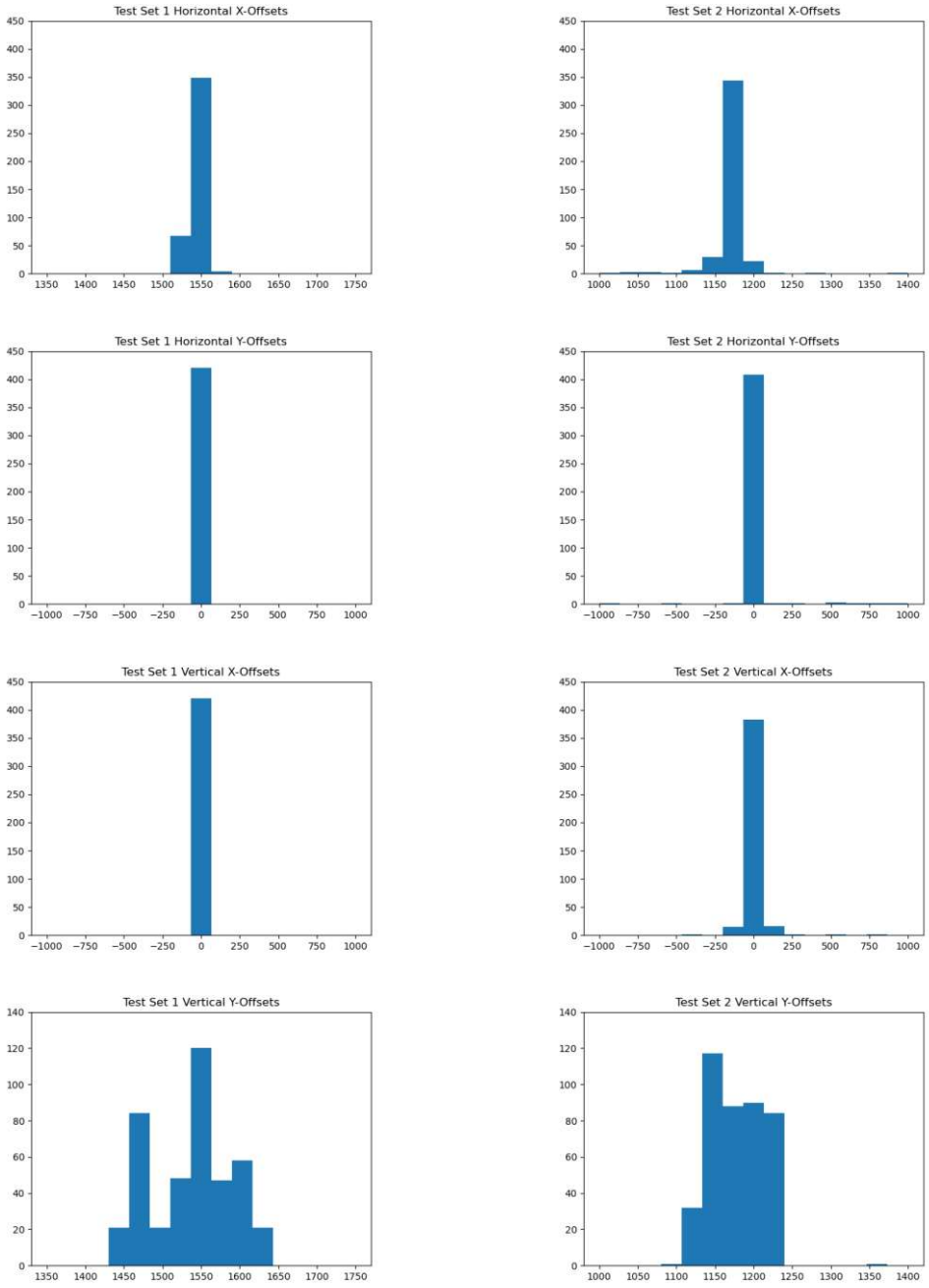


Figure 7.3: Histogram with 15 bins for computed offsets in test set 1 and 2.

microscope did not change significantly between creating the two test sets, it follows that many computed offsets in test set 2 must be wrong. Furthermore, this outlines the potential of statistical methods to improve the global stitching algorithm.

To combine the match score with the statistical likelihood of an offset being correct, the new score was computed by multiplying the match score with the probabilities of the x and y offset. This means that for a statistically unlikely offset to be accepted as the best choice by the algorithm, the match score must be proportionately higher than that of the other available offsets, with higher statistical probabilities of x and y being correct. The new algorithm thus consists of the following steps:

1. Start at the image closest to the center of the fused image, set its location to (0,0).
2. Create a list of all images with no location set, that have at least one neighbor with a set location.
3. Choose the image from this list whose offset has the highest score, defined as $MScore = PCScore * P(x) * P(y)$, where $PCScore$ is the phase correlation score, and $P(x)$ and $P(y)$ are the probabilities of the respective offsets.
4. Compute the location of this image by combining the offset with the location of the neighbor image.
5. Repeat steps 2 to 4 until all locations have been computed.

Of course, there are many ways to modify this simple approach and, more importantly, tailor it to the stage movement model of a specific microscope. However, the goal of this work is to find an algorithm that will work well on many different SEMs. Improvements for a specific SEM, such as using automated aggregation of stage movement data across multiple image sets to create better statistical mechanisms, are left for future work.

7.3 Version 3: Match Multiple Neighbor Tiles

This improvement upon Version 1 aims to reduce the intensity of tearing caused by accumulating offset errors without modifying the original images by warping. To test its efficacy over Version 1 individually, Version 3 does not include the statistical modifications introduced in Version 2.

The general idea for improving the algorithm regarding tearing is that if a tile has two or more neighbors with set locations, and the offsets to those tiles agree within a certain margin of error, then this is most likely the correct location or at least close enough. To determine what can be considered “close enough”, there are two possible options: Option 1 is that the difference to the optimal offset must be visually indistinguishable. Since offsets are floating-point numbers that are rounded to the next integer for image fusion, this would mean a difference of at most 1 pixel (px).

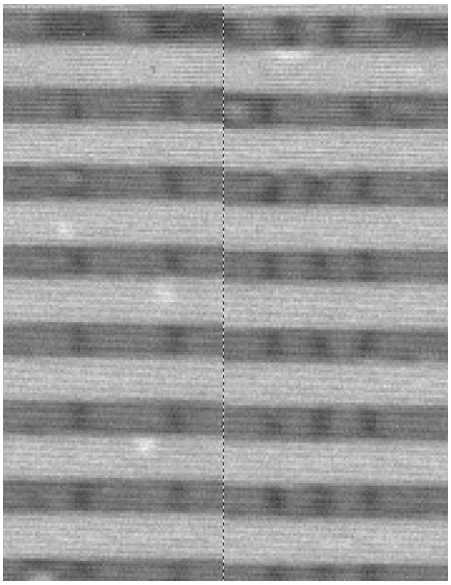


Figure 7.4: Demonstration of a 2-pixel offset deviation in test set 1.

Option 2 is that it must be sufficient for the purpose of reverse engineering, meaning that the smallest visual features required for reverse engineering must be recognizable with this offset. The exact definition in terms of the maximum number of pixels of deviation from the optimal offset depends on the image quality. In the microscope images used for this work, a maximum deviation of 2 pixels was chosen, as this still allowed correct tracing of connections. An example illustrating such a 2px shift is shown in Figure 7.4. In the implementation of this algorithm, the maximum tolerable deviation can be modified to enable adjustment depending on image quality and reverse engineering requirements.

This algorithm consists of the following steps:

1. Start at the image closest to the center of the fused image, set its location to (0,0).

2. Create a list of all images with no location set, that have at least one neighbor with a set location.

3. For each tile from this list, the score is determined as the sum of match scores of all agreeing offsets to neighbor tiles with locations. The tile with the highest overall score is chosen, and its location is computed and set.

4. Repeat steps 2 and 3 until all locations are determined.

7.4 Version 4: Match Multiple Neighbor Tiles With Added Offset Statistics

Version 4 combines offset statistics of Version 2 with the multi-neighbor offset combination of Version 3. The implementation is similar to Version 3, with the offset scores being replaced by the combined score introduced in Section 7.2, defined as $MScore = PCScore * P(x) * P(y)$, where $PCScore$ is the phase correlation match score, and $P(x)$ and $P(y)$ are the probabilities associated with the respective offsets.

CHAPTER 8

Global Stitching Evaluation

Evaluation of stitching results is a challenging task. Complicating factors are the size of the image sets, the large number of possible solutions, and most of all, the uncertainty of whether an optimal (or even reasonably good) solution exists, given the challenges posed by distortion. To evaluate the results of algorithms introduced in Chapter 7, several metrics were developed that indicate different qualities of the resulting fused image. These metrics rely on the results of the utilized image registration algorithm and thus can not be applied to fused images produced by other programs. Additionally, manual inspection and comparison were utilized to compare the results to state-of-the-art tools. To improve legibility, the algorithms Version 1 through 4 are henceforth abbreviated as V1-V4.

8.1 Image Sets

The image sets for the evaluation of global stitching algorithms consist of the two previously introduced test sets and one additional image set. While image sets from a range of different SEMs would be optimal for evaluation, only one SEM could be used to create the image sets, and no suitable public image sets were available. Thus, all three image sets used for evaluation were produced by the same SEM. It can thus be argued, that the algorithms introduced in this work are optimized for the specific flaws of this SEM. However, while designing and improving the global stitching algorithms, special care was taken to implement the algorithms as independent of the utilized hardware as possible.

Of the utilized image sets, the first two were already introduced in Chapter 5 to evaluate image registration algorithms. Additionally, a third test set was created. The image quality of this third test set is between the two previously introduced test sets, albeit with higher variability in stage movement accuracy. While using more image sets would improve the significance of the results, the expensive and time-consuming nature of the

sample preparation and image acquisition processes prevented the creation of additional sets.

8.2 Evaluation Methodology

Several evaluation metrics have been utilized to reflect different requirements depending on the intended use case of the final fused image. The automated evaluation was conducted using two metrics, exact offset count and absolute offset deviation. Since these metrics depend on metadata such as the underlying image registration results, using these metrics was limited to the algorithms developed in this work. Additionally, a comprehensive manual evaluation was conducted to compare the results of these algorithms to state-of-the-art tools. During the manual evaluation, errors encountered during the visual inspection were assigned to one of four categories, depending on the severity of the error. The subsequent sections describe the metrics for automated evaluation and the manual evaluation in detail.

8.2.1 Exact Offset Count

This metric computes the number of exact offsets divided by the total number of offsets. An exact offset, in this case, means that an image's offset to a given neighbor in the fused image is equal to the offset computed by the utilized image registration algorithm. Since many computed offsets will be inaccurate in most image sets, this score is not expected to reach 100%. Furthermore, even if all offsets computed by the image registration algorithm would be correct, distortion would make it impossible to fuse the complete image perfectly in all but the best quality image sets. As such, this metric is an indicator of how many of the offsets in the fused image are expected to be accurate within a predefined tolerance (1 or 2 pixels in this evaluation). This metric strictly divides the offsets into the categories "correct" and "incorrect" and labels minor deviations as wrong, which may not reflect subjective judgment when manually inspecting a fused image.

8.2.2 Absolute Offset Deviation

This metric computes the total sum of pixels that offsets in the fused image deviate from their offset as computed by the image registration algorithm. As with the previously introduced metric, this metric assumes the image registration algorithm is always correct. Since this will never be the case in real-world scenarios, the computed score on its own is irrelevant. Only differences between algorithms' scores should be considered. Compared to exact offset count, the results of this metric are closer to an impression gained by manual inspection, as minor deviations influence the score less than large ones.

8.2.3 Manual Evaluation

A thorough manual evaluation was used to compare the results of algorithms introduced in this work to each other and to state-of-the-art tools. The main reason for choosing this

approach was that the previously discussed metrics to quantify and compare results of algorithms V1 to V4 relied on the results of the image registration algorithm as well as on being able to identify where individual tiles were placed in the fused image. Especially in the case of MS-ICE, this was not considered feasible as the source code was not available and technical details of the project file were not available at the time. Because of the time-consuming nature of this manual evaluation, two state-of-the-art tools were chosen for comparison. Details of this preselection are discussed in Section 8.4.

To create the fused images, V1 to V4 and MIST were configured to apply no warping or blending of images. This reduced the overall image quality but simplified spotting errors. These aspects could not be configured for MS-ICE, giving MS-ICE a significant advantage in the scope of this evaluation.

Manual evaluation posed several challenges. Most importantly, manually inspecting multiple gigapixels worth of noisy image data leads to some errors remaining undetected. Furthermore, the exact length of a tearing error was often hard to determine, as was the correct placement of a given tile. Determining the maximum amount of tearing to a pixel-exact value was not feasible, as the misplacement offset varied along most tearing errors. To be able to quantify the results, errors in the fused image were assigned to one of the following classifications:

1. Irrelevant errors: Errors outside the IC area, usually caused by stage movement induced parallax effects between IC area and stage surface background.

2. Small errors: Tearing where the offset between two tiles is small enough that correct placement can be easily determined and possibly fixed using minor warping of the affected images. One example would be horizontal tearing of vertical lanes, where the offset is so small, that the offset to the correct placement of affected traces is smaller than the offset to the next parallel trace. Examples of small errors are shown in Figure 8.1.

3. Medium errors: Tearing or tile misplacement, where correct placement is not immediately obvious, for example, when the offset is large enough for parallel traces to appear closer to a wrong offset than the correct offset when considering only a single trace. In the absence of smaller features or tearing of individual traces or features, subjective estimation of the amount of displacement was used to assign the correct category in line with category assignments in other areas of the same image set. Two such errors are shown in Figure 8.2.

4. Large errors: Tearing or tile misplacement, where correct placement is hard or impossible to determine, or where misplacement obscures significant features of the IC, such as vias. Two large errors are shown in Figure 8.3.

Cases where it was not clear whether an error had been made by the algorithm were not counted. Errors that fit several of the categories were assigned the worst matching rating.

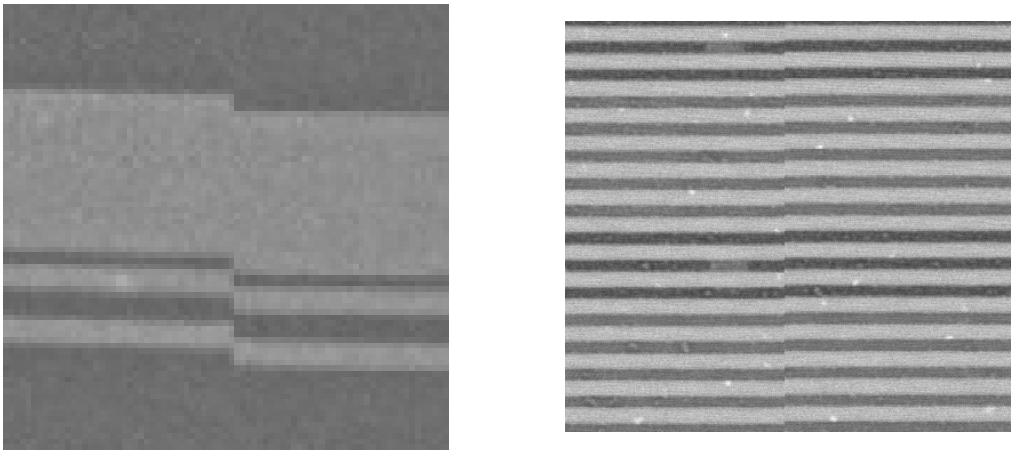


Figure 8.1: Two examples of small errors. In both cases, the traces are still partially connected and close to their correct position.

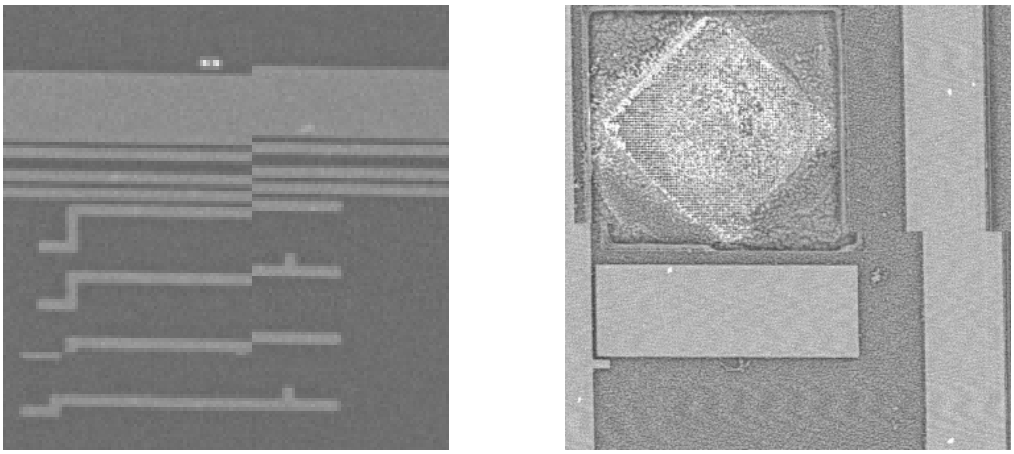


Figure 8.2: Examples for medium errors: In the left image, the correct offset for a given trace is as close as the offset required to connect the trace to the adjacent, parallel trace. On the right, the offset to the correct solution is as large as other medium errors in the same image set.

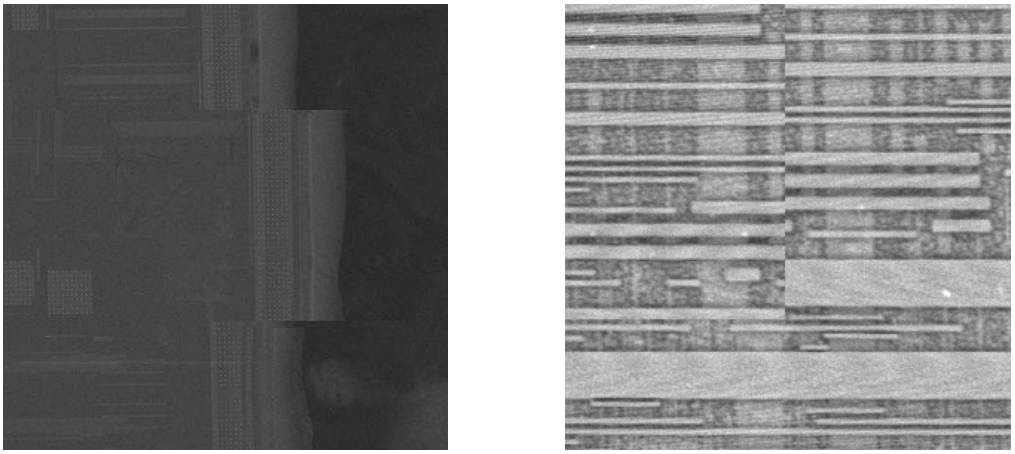


Figure 8.3: Examples for two large errors: in the left image, the tile was significantly misplaced. In the right image, misalignment of the tile hides significant features.

To assure equal scrutiny in the assessment of all tested algorithms and stitching tools, the fused images were split into areas of 4400x4400 pixels and composed into images showing the same area for each algorithm, as depicted in Figure 8.4. Efforts were made to count each tearing error or misplacement exactly once where possible, except for tearing that crossed the boundaries of the currently inspected area, as tracing errors across area borders would have made manual evaluation prohibitively time-consuming. Investigation of a subset of areas showed that this type of double-counting affected the evaluated algorithms proportionally to the number of errors in a given region. While the absolute number of errors may not be perfectly exact, the relative differences between algorithms are still expected to be sufficiently accurate.

Furthermore, while the distinction between small and medium errors was easy to determine in most cases, the decision of when to place an error in the medium or large category was subjective in some borderline cases. Special care was taken to apply the same classification rules to all algorithms.

8.3 Evaluation Results for Exact Offset Count and Absolute Offset Deviation

The following sections discuss the evaluation results for exact offset count and absolute offset deviation for each of the three test sets. Differences between the algorithm's scores show that statistics affect absolute offset deviation significantly, while the multi-edge approach of V3 and V4 improves exact offset count scores with a tolerance of two pixels. V1 delivered the best results for exact offset count with a tolerance of only one pixel in 2 of 3 test sets while also achieving the second-best score in the third test set.

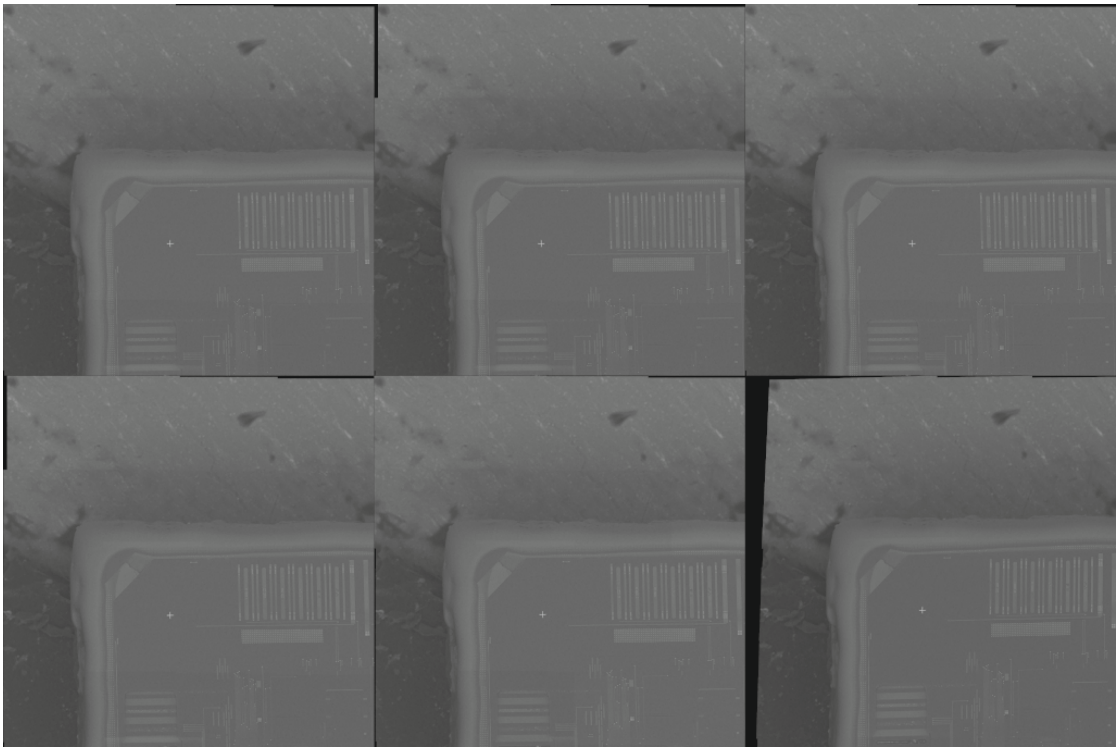


Figure 8.4: Example of a composed image for comparison and evaluation. The lower right section shows how MS-ICE warped tiles, unlike the other evaluated tools.

8.3.1 Test Set 1

The results for test set 1 are shown in Figure 8.5. Images of test set 1 show high contrast, low noise, and minimal distortion, perfect conditions for the image registration algorithm. Therefore, one would expect that additional sources of positioning information, such as that stemming from statistical analysis over all computed offsets, to add little additional value. This is precisely what the results for this test set show. Algorithm version 3, which matches multiple neighbors without utilizing additional offset statistics, yields the lowest absolute offset deviation and the highest exact offset count with a tolerance of 2 pixels. V1 beats V3 when allowing for no offset deviation (tolerance 1 pixel) while reducing scores for the other metrics.

8.3.2 Test Set 2

Unlike test set 1, the typical characteristics of test set 2 are low contrast and a high amount of noise. Also, several images exhibit significant distortion. When comparing the exact offset count metrics with test set 1, the algorithms yield very similar results, as shown in Figure 8.6. The multi-neighbor stitching approaches outperform algorithms V1 and V2, with t1 values similar or better and t2 values distinctively better. Regarding

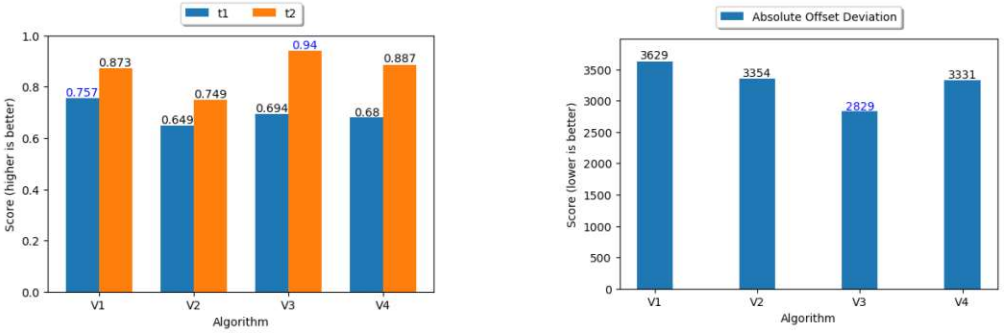


Figure 8.5: Evaluation results for test set 1, best results highlighted blue. The exact offset count was computed with a tolerance of one pixel (t1) and two pixels (t2).

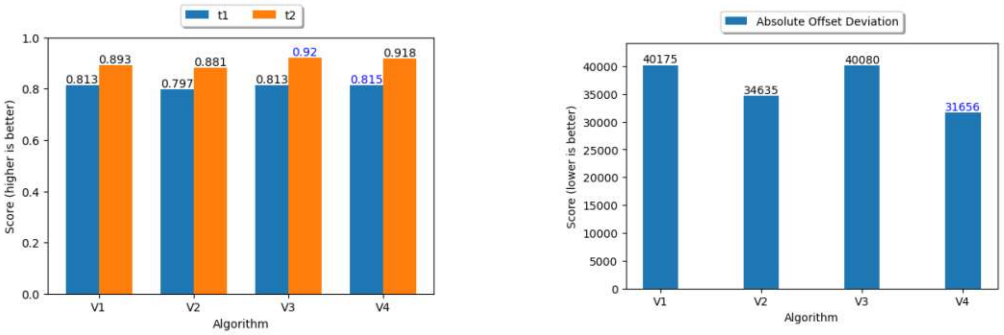


Figure 8.6: Evaluation results for test set 2, best results highlighted blue. The exact offset count was computed with a tolerance of one pixel (t1) and two pixels (t2).

absolute offset deviation, the algorithms V2 and V4, which use statistics to improve their results, clearly outperform V1 and V3, which only rely on the match score computed by the image registration algorithm.

8.3.3 Test Set 3

As shown in Figure 8.7, test set 3 has a different winner for each metric, with V1 yielding the best results for exact offsets with a maximum tolerance of 1 pixel, V3 scoring best with a tolerance of 2 pixels, and V4 minimizing total offset deviation better than the other algorithms. This demonstrates the trade-off between using exact offset matches as computed by the image registration algorithm and minimizing offset error accumulation to reduce the intensity of tearing.

8.4 Preselection for Manual Evaluation

Due to the time-consuming nature of manual evaluation, comparison to state-of-the-art software was limited to two of the tools introduced in Chapter 3: MIST and MS-ICE.

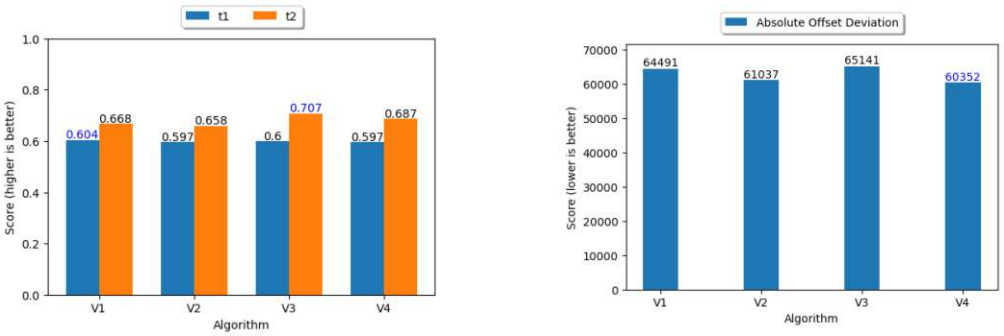


Figure 8.7: Evaluation results for test set 3, best results highlighted blue. The exact offset count was computed with a tolerance of one pixel (t1) and two pixels (t2).

These tools were selected for their superior performance compared to the other candidates when run on the test sets. The subsequent sections describe all candidates and briefly outline details of their performance.

8.4.1 Fiji/ImageJ Plugins for Microscopy Images

Fiji is an open-source project based on ImageJ. At the time of writing, it is one of the few open-source tools capable of handling hundreds of individual images, even if the resulting stitched image does not fit into RAM.

Three relevant plugins are optimized for microscopy images: The Grid/Collection stitching algorithm, which is included by default, its successor BigStitcher and the Microscopy Image Stitching Tool (MIST) plugin published by the National Institute of Science and Technology (NIST) [32, 33, 39–41].

8.4.2 Grid/Collection Stitching Plugin

The Grid/Collection stitching plugin created by Preibisch et al. [33] utilizes cross-correlation to match image pairs and subsequently a global optimization algorithm to compose the final image. Unlike many other tools, this software is capable of stitching 2D and 3D images. For IC images with low noise and high contrast, the resulting fused images produced by Fiji are adequate in many cases, but as can be seen in Figure 8.8, even with high-quality images, missing tiles are not unusual. For IC images with low contrast and low signal-to-noise ratio (SNR), the results are significantly worse, as shown in Figure 8.9. In these cases, the algorithm often decides not to place tiles in the fused image or to draw them on top of other tiles, leading to large black areas. It is possible to manually place the missing tiles by editing the tile configuration file generated by Fiji, but this would be exceedingly time-consuming in cases with many misplaced tiles. Additionally, this program requires knowledge of the overlap parameter, the expected percentage of overlap between adjacent tiles.

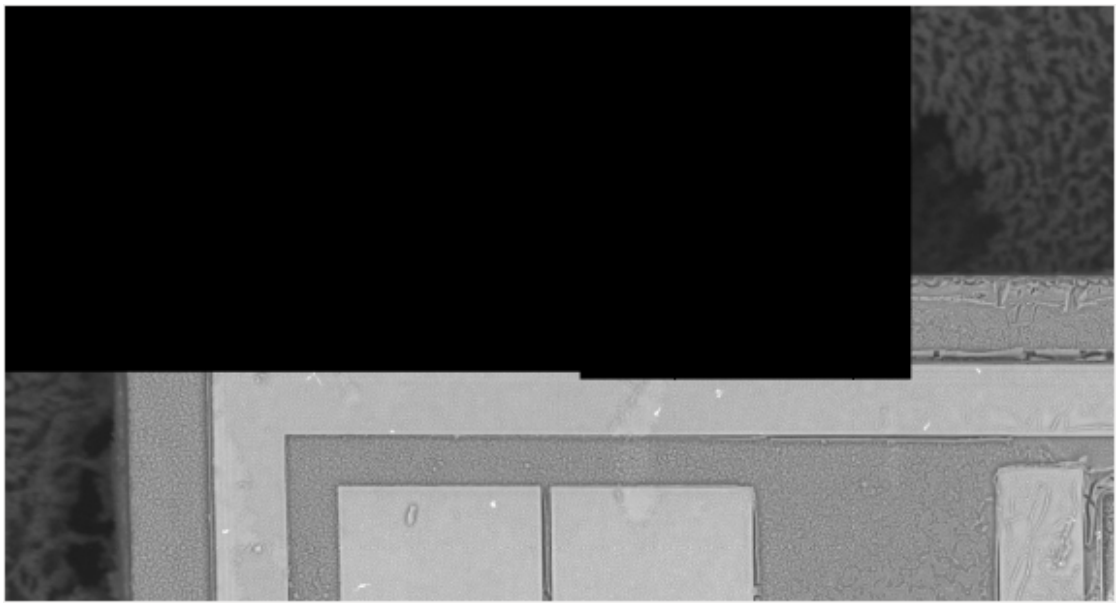


Figure 8.8: Missing tiles in test set 1 stitched with Grid/Collection stitching Fiji-plugin

8.4.3 Microscopy Image Stitching Tool (MIST)

This Fiji plugin was published by Chalfoun et al. around the same time the tools and algorithms for this work were developed [32, 40, 41]. It follows similar ideas in principle: Images are matched pairwise. Overlap, and other parameters are set manually or detected automatically. A model of the stage movement is then inferred from the computed offsets. Subsequently, a spanning tree is generated that follows the highest-rated stitching results in combination with the best tile position deduced from the stage movement model. This approach can lead to misalignments, as shown in Figure 8.10.

While MIST writes several intermediate files before stitching the result, manual correction of the computed results is by no means trivial, as these files have to be edited with a text editor without any additional software support, for example in finding correct offsets.

8.4.4 BigStitcher

A recently announced Fiji plugin that succeeds the previously discussed Grid/Collection Stitching plugin has been published by Preibisch et al. around the time of writing of this document [39]. While this Plugin is optimized for microscopy images of biological research, the published description, especially of its new global optimization algorithm, appears promising. At the time of writing, the available version is a beta release, and it was not possible to generate a fused image from the test sets due to runtime errors.

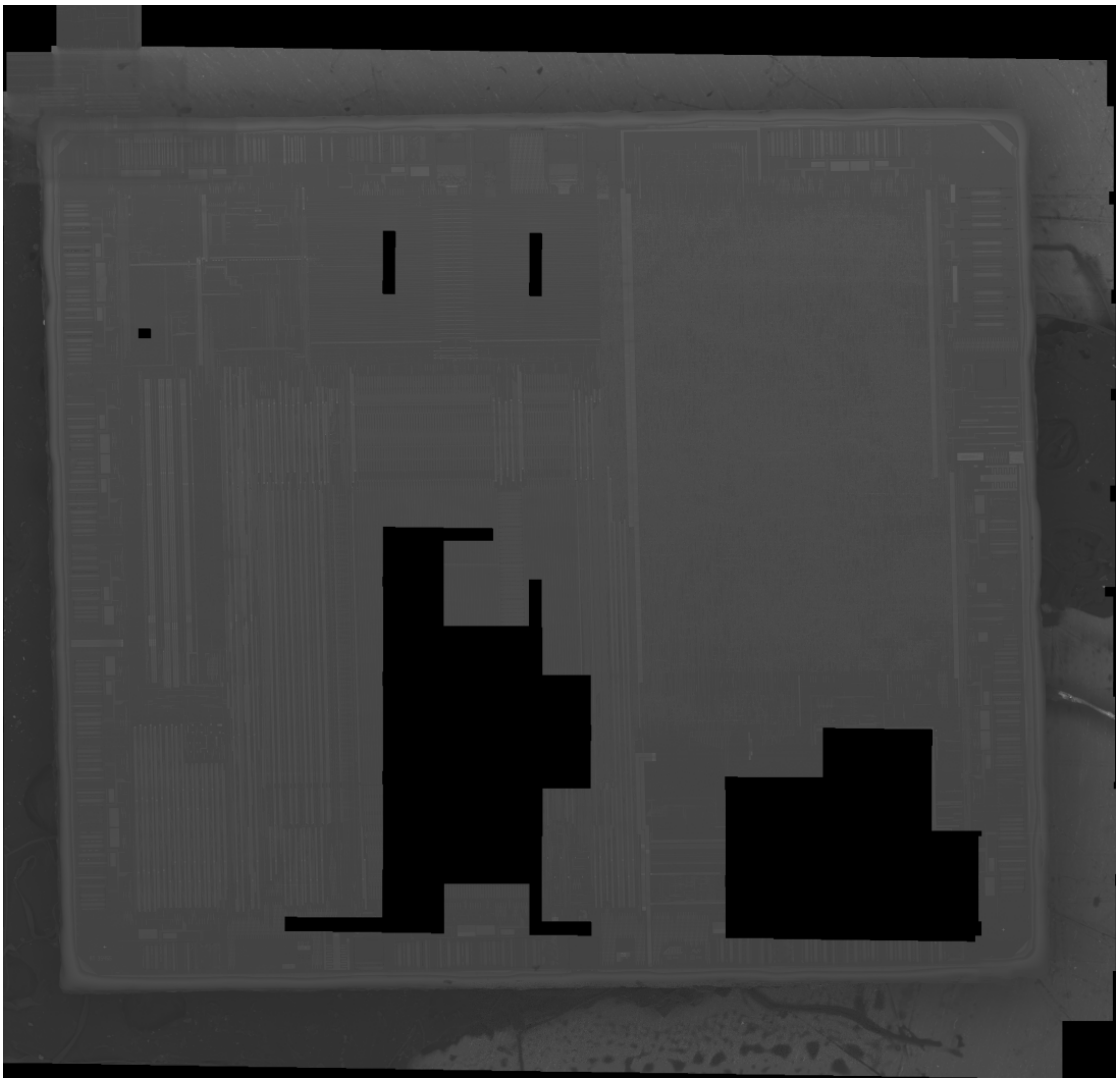
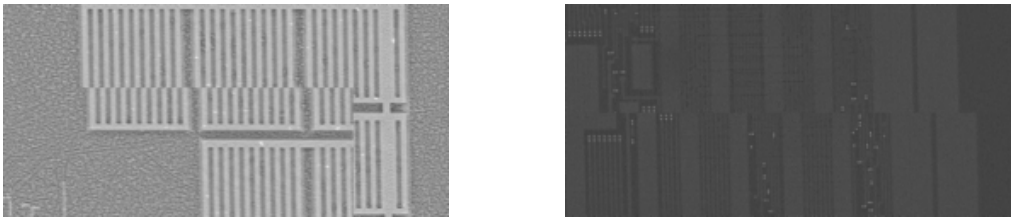


Figure 8.9: Missing tiles in test set 2 stitched with Grid/Collection stitching Fiji-plugin



(a) Test set 1 (b) Test set 2

Figure 8.10: Examples of misalignments in both test sets when stitched with MIST

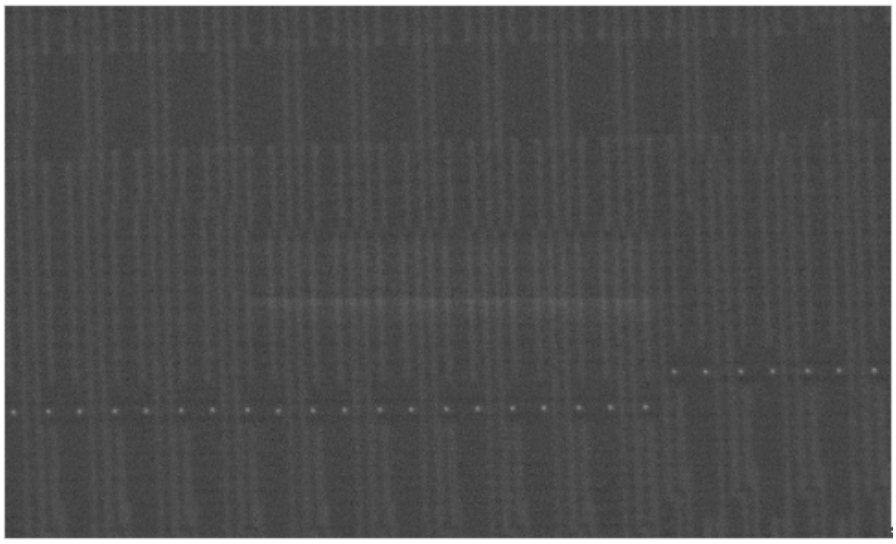


Figure 8.11: Misalignment in the fused image produced by MS-ICE for test set 2

8.4.5 Microsoft Image Composite Editor (MS-ICE)

This software is published by Microsoft and is free to use. It is optimized for panoramas, but the settings support adjustment appropriate for stitching microscopic images. Unlike the Fiji plugins, no intermediate output is generated, and besides changing projection type and cropping, no manual correction is possible.

This software uses warping and blending which results in a visually pleasing fused image, at the cost of hard-to-spot errors. One example of such an error is shown in Figure 8.11. In some cases, warping may become so severe that traces can not be aligned to a straight design grid anymore, thus complicating subsequent steps of the reverse engineering process. Another significant drawback of MS-ICE is that in more challenging IC image sets, the software may just quit with an error message, where other tools would at least create a best-effort fused image with only a few areas unsuitable for reverse engineering.

Manual correction of stitching errors may be possible in theory by changing the .spj file created by MS-ICE, but due to the transformation matrices in the file, stored data would have to be converted into human-readable format first.

8.4.6 PTGui Pro

PTGui Pro is a closed source panorama stitching tool with a free fully-featured trial version available (version 11.16 at the time of writing). A guide on how to stitch microscopy images with PTGui is available online [43]. However, the test sets used in this work failed to stitch automatically, with the amount of manual intervention necessary to assist PTGui in fusing the complete image being prohibitively high. The automated stitching results are shown in Figure 8.12.

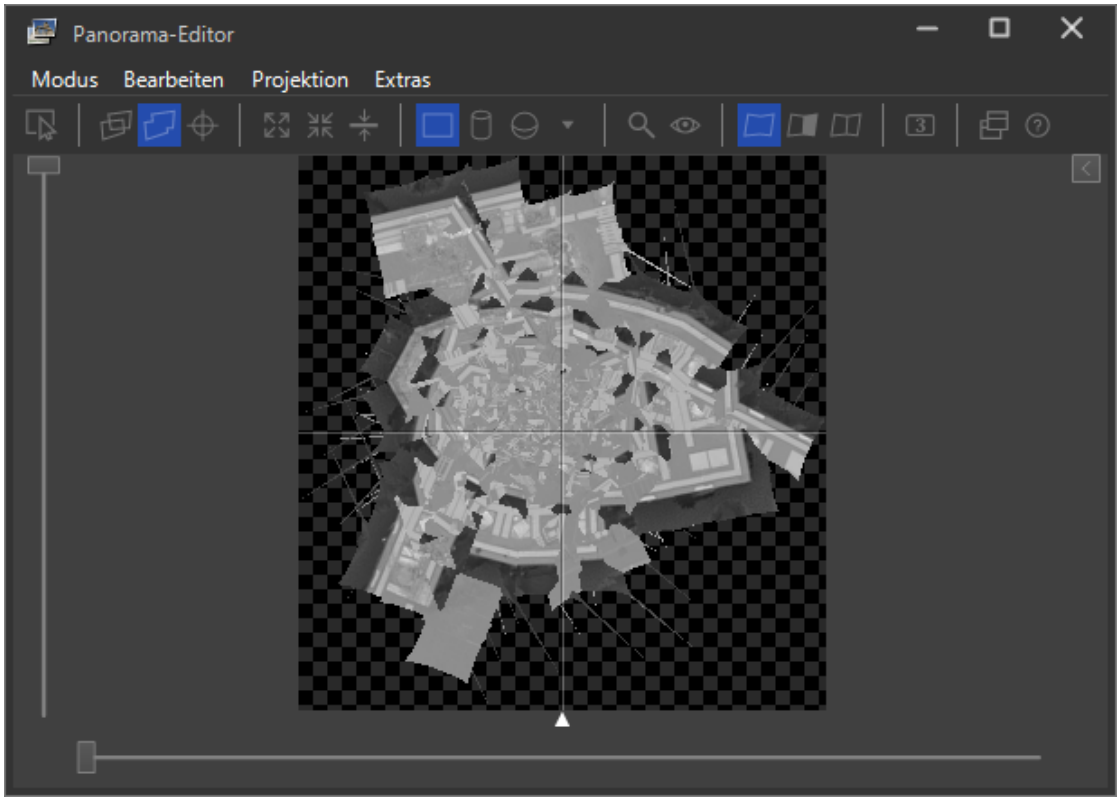


Figure 8.12: Stitching result of PTGui on test set 1

8.4.7 Teorex Photostitcher

Teorex advertises its photo stitching tool's ability to stitch microscopy images, even though this is not the main focus of the application [44]. This could not be verified, as Photostitcher version 2.0 crashed without error messages when attempting to stitch the test sets.

8.5 Manual Evaluation Results

The subsequent sections discuss the manual evaluation results for each of the three test sets. Overall the results show that the algorithms introduced in this work outperform MIST on all three test sets and MS-ICE on the two test sets with worse contrast and higher noise levels. MS-ICE performed best on the high-contrast test set 1, with the caveat that, unlike its competitors, MS-ICE used blending and warping.

8.5.1 Manual Evaluation of Test Set 1

The results for each tested tool/algorithm are shown in Figure 8.13. Judging from the previous evaluations, V1 and V3 were expected to attain better scores than V2 and V4.

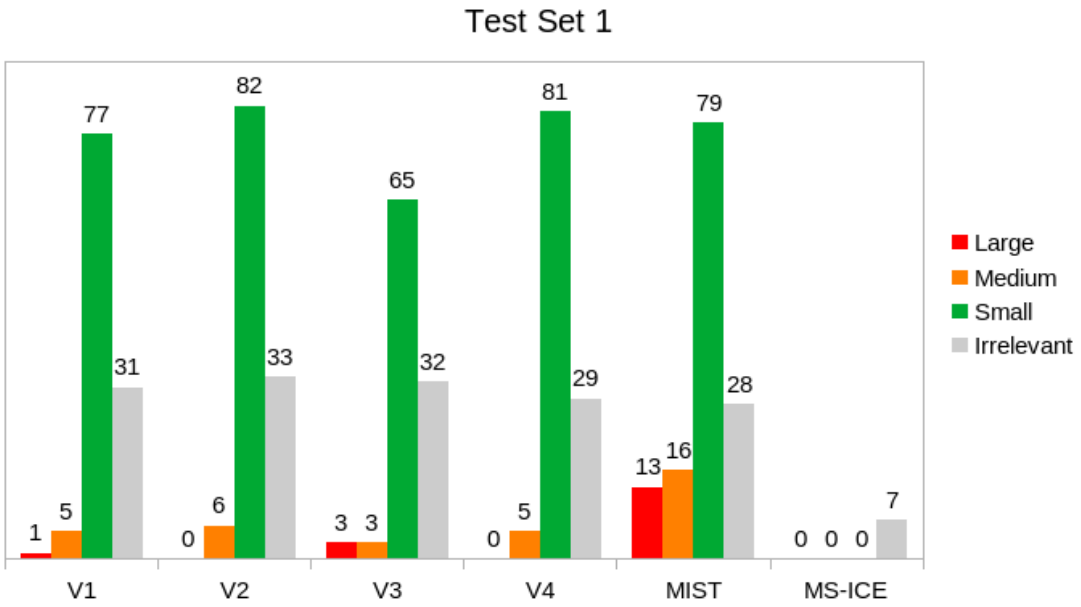


Figure 8.13: Manual evaluation results for test set 1.

This is only partially reflected in this metric, with both of these algorithms having a lower number of small errors as well as a lower total number of errors. In turn, both introduced more large errors. V1 to V4 all outperform MIST, which by this metric has significantly more medium and large errors, with roughly the same number of small and irrelevant errors. All of these algorithms were outperformed by Microsoft Image Composite Editor, the fused image of which did not exhibit any detectable errors in areas showing the IC surface. However, it is important to note that, unlike the other algorithms, MS-ICE was allowed to warp and blend images, which is bound to have a significant effect, especially on the number of small and medium errors.

8.5.2 Manual Evaluation of Test Set 2

For test set 2, V1 and V3 show a lower number of medium and large errors when compared with V2 and V4. MIST again performed significantly worse than algorithms V1 to V4, with a higher number of medium and large errors. Interestingly, MS-ICE also scored worse than algorithms V1-V4 when considering the sum of small, medium and large errors, albeit by a much lower margin. MS-ICE performed much better in the category of irrelevant errors, which is most likely due to blending and warping. The previously listed Figure 8.4 shows how much MS-ICE warped the tiles and overall image compared to its competitors.

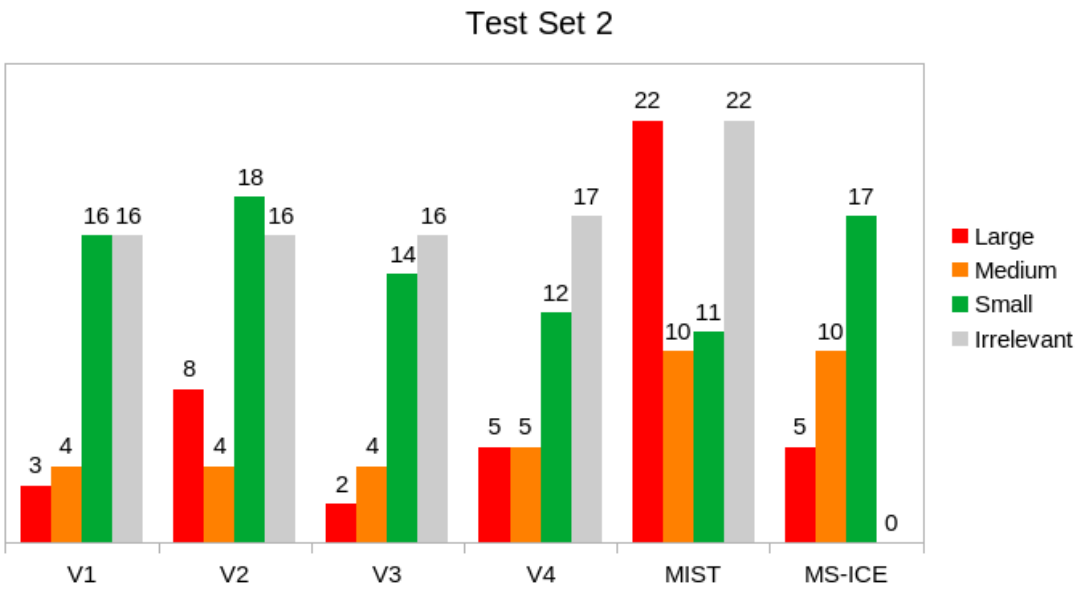


Figure 8.14: Manual evaluation results for test set 2.

8.5.3 Manual Evaluation of Test Set 3

All tested tools and algorithms struggled with some unique challenges posed by this image set. MS-ICE failed to provide a usable result, as shown in Figure 8.15. Since the remaining five candidates struggled with two particular rows of images, these rows were removed from the evaluation. As Figure 8.16 shows, V1 and V3 are slightly ahead of V2 and V4. MIST yielded comparable results; it scored slightly worse with the highest number of large and small errors but the lowest number of medium and irrelevant errors. When considering the total number of small, medium and large errors, V1-V4 all outperformed MIST.

8.6 Discussion

The algorithms introduced in this work improve upon the two evaluated state-of-the-art tools in two of the three tested image sets. Compared to the state-of-the-art open-source tool MIST, these algorithms achieve better results for all three test sets. The most significant improvement stems from modifying the minimum spanning tree approach used in V1 to use all available information on neighboring tiles when determining the location of a given tile in the algorithms V3 and V4. V2 never scored best between these four algorithms. The differences between the four algorithms are outlined by the two different metrics used in the first part of this evaluation, showing the trade-off between absolute accuracy (all or nothing) versus minimizing the accumulation of errors, essentially by splitting large offset errors into multiple smaller offset errors.

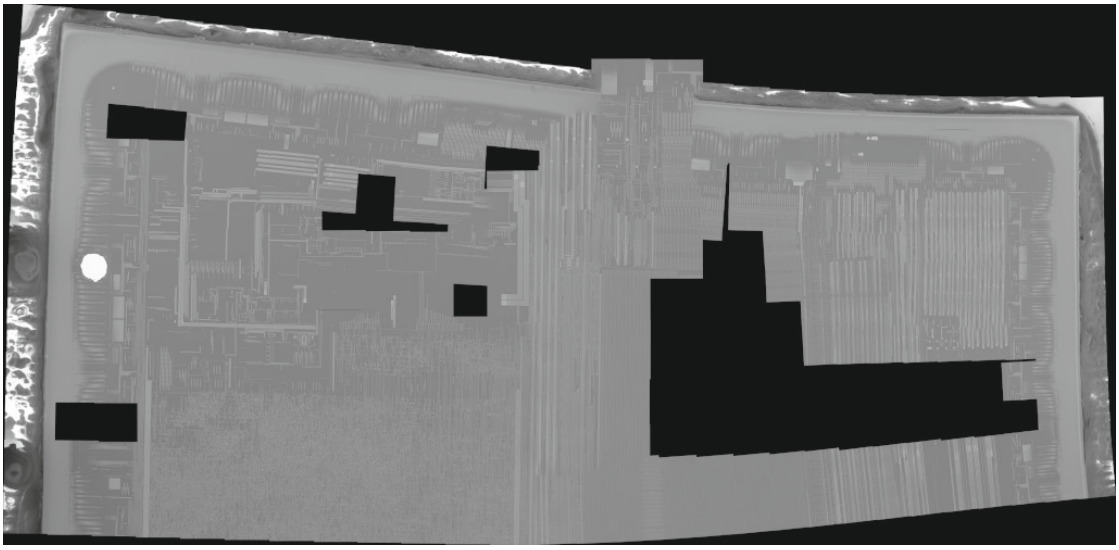


Figure 8.15: Fused image composed by MS-ICE.

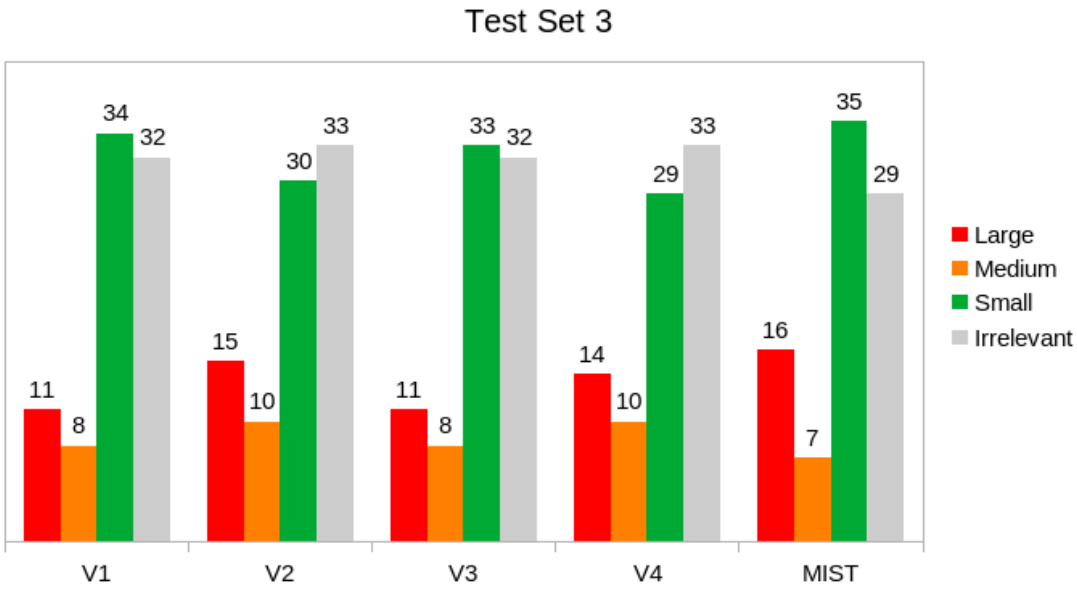


Figure 8.16: Manual evaluation results for test set 3.

CHAPTER 9

Conclusion and Future Directions of Work

Reverse engineering integrated circuits (ICs) requires sufficiently detailed imaging data, with modern IC manufacturing processes producing feature sizes too small for optical microscopes. The cost of state-of-the-art scanning electron microscopes (SEMs) can be prohibitive for the purpose of reverse engineering ICs, with used SEMs being more readily available and affordable at the cost of lower image quality. To lower the entry bar to this type of research, better image processing algorithms are required to allow the use of such lower-quality images.

This work introduced four algorithms to compose individual scanning electron microscope images into a fused image, with the explicit purpose of providing a basis for reverse engineering integrated circuits. These algorithms were shown to work particularly well for images with high noise and low contrast. Compared to two state-of-the-art tools, they outperformed the proprietary tool Microsoft Image Composite Editor in 2 of the 3 test sets and the open-source tool MIST in all 3 test sets. While the number of test sets was limited, these results are promising and show that extending the minimum spanning tree approach by combining multiple edges in certain situations can improve results significantly. This approach decreased the number of large errors, in some cases at the cost of introducing multiple smaller errors.

The results of this work may be extended further by improving upon the utilized offset statistics. However, more image sets from different microscopes are required to avoid overfitting such algorithms to one particular microscope. Furthermore, subsequent image sets of one SEM could be used to automatically adjust these statistics to the unique properties of the microscope's stage movement for improved results of subsequent image fusion. Finally, given enough image data and computing resources, machine learning could improve image registration and stitching error correction.

APPENDIX A

Detailed Evaluation Results for Image Registration Algorithms

The following tables show the evaluation data across all evaluated parameters. The SIFT evaluation in Table A.1 differs from the other evaluations in that the test sets were reduced to 100 random image pairs to reduce the overall required computation time. Bold font indicates the best results.

Table A.1: Sift Evaluation Summary

		Test Set 1		Test Set 2	
Contrast Threshold	**Edge Threshold**	**Correct**	**Avg Exec Time**	**Correct**	**Avg Exec Time**
0.01	8	**100**	306.1453	94	53.7046
0.01	9	**100**	320.5279	94	55.4632
0.01	10	**100**	339.2553	**95**	56.3139
0.01	11	**100**	344.3994	**95**	56.7571
0.01	12	**100**	349.2112	**95**	57.0676
0.02	8	**100**	275.1676	85	6.4590
0.02	9	**100**	285.0985	85	6.5390
0.02	10	**100**	291.8855	85	6.5690
0.02	11	**100**	306.9043	85	6.6304
0.02	12	**100**	316.3945	85	6.6561
0.03	8	**100**	188.9968	79	3.3813
0.03	9	**100**	193.9263	79	3.3922
0.03	10	**100**	199.0301	79	3.4168
0.03	11	**100**	203.3335	79	3.4157
0.03	12	**100**	206.7627	79	3.4210

0.04	8	**100**	93.4027	78	2.5574
0.04	9	**100**	96.4854	78	2.5605
0.04	10	**100**	99.5494	78	2.5618
0.04	11	**100**	102.0732	78	2.5651
0.04	12	**100**	104.6575	78	2.5676
0.05	8	**100**	32.4341	75	2.3470
0.05	9	**100**	34.0302	75	2.3547
0.05	10	**100**	35.1621	75	2.3663
0.05	11	**100**	36.4704	75	2.3454
0.05	12	**100**	37.4885	75	2.3549

Table A.2: ORB Evaluation Summary

		Test Set 1		Test Set 2	
WTA_K	**Fast Thresh.**	**Correct**	**Avg Exec Time**	**Correct**	**Avg Exec Time**
2	5	349	0.5175	268	0.2433
2	6	350	0.4740	270	0.2138
2	7	349	0.4262	270	0.1933
2	8	**351**	0.3898	271	0.1804
2	9	350	0.3599	267	0.1727
2	10	349	0.3354	271	0.1700
2	11	347	0.3175	**275**	0.1859
2	12	**351**	0.3089	270	0.1845
2	13	349	0.2863	271	0.1631
2	14	350	0.2695	267	0.1877
2	15	348	0.3245	270	0.1402
2	16	349	0.2930	271	0.1403
2	17	348	0.2572	270	0.1331
2	18	349	0.2462	270	0.1303
2	19	346	0.2354	270	0.1289
2	20	350	0.2204	273	0.1270
2	21	350	0.2197	270	0.1282
2	22	349	0.2144	268	0.1335
2	23	348	0.2084	268	0.1269
2	24	348	0.2086	271	0.1238
2	25	348	0.1991	267	0.1236
3	5	338	0.5617	264	0.2703
3	6	337	0.4656	268	0.2469
3	7	339	0.4404	263	0.2149
3	8	341	0.4265	268	0.1980
3	9	337	0.3527	266	0.1837

3	10	339	0.3199	267	0.1721
3	11	336	0.2990	263	0.1593
3	12	341	0.2782	265	0.1579
3	13	341	0.2635	263	0.1522
3	14	335	0.2554	262	0.1482
3	15	341	0.2461	263	0.1368
3	16	336	0.2210	263	0.1360
3	17	338	0.2116	264	0.1328
3	18	336	0.2039	264	0.1313
3	19	338	0.1925	265	0.1292
3	20	339	0.1899	262	0.1276
3	21	337	0.1894	263	0.1268
3	22	339	0.1785	260	0.1261
3	23	337	0.1767	261	0.1244
3	24	339	0.1673	263	0.1228
3	25	340	0.1640	263	0.1222
4	5	335	0.4850	265	0.2588
4	6	334	0.4529	265	0.2235
4	7	334	0.4111	267	0.1927
4	8	334	0.3787	264	0.1780
4	9	335	0.3652	265	0.1664
4	10	332	0.3394	266	0.1575
4	11	333	0.3109	264	0.1505
4	12	332	0.2905	263	0.1452
4	13	336	0.2669	263	0.1413
4	14	335	0.2498	265	0.1418
4	15	334	0.2410	262	0.1348
4	16	335	0.2298	263	0.1359
4	17	334	0.2132	264	0.1320
4	18	334	0.2077	264	0.1305
4	19	334	0.1949	266	0.1278
4	20	333	0.1864	266	0.1271
4	21	332	0.1812	268	0.1269
4	22	333	0.1835	265	0.1269
4	23	335	0.1829	269	0.1285
4	24	332	0.1663	266	0.1287
4	25	333	0.1655	265	0.1250

Table A.3: Phase Correlation Evaluation Summary

	Test Set 1		Test Set 2	
Overlap	**Correct**	**Avg Exec Time**	**Correct**	**Avg Exec Time**
0.1	0	0.0507	0	0.0472
0.15	419	0.0942	0	0.0881
0.2	**420**	0.0950	0	0.0915
0.25	**420**	0.1619	318	0.1573
0.3	**420**	0.1496	346	0.1452
0.35	**420**	0.1786	**356**	0.1697
0.4	419	0.2235	**356**	0.2147
0.45	0	0.3603	352	0.3376
0.5	0	0.2974	351	0.2984
0.55	0	0.3289	353	0.3242
0.6	0	0.3655	351	0.3559

Table A.4: Cross-Correlation Evaluation Summary

		Test Set 1		Test Set 2	
Expected Overlap	**Minimal Overlap**	**Correct**	**Avg Exec Time**	**Correct**	**Avg Exec Time**
0.1	0.1	0	0.0796	0	0.0754
0.15	0.1	0	0.1355	0	0.1239
0.2	0.1	**420**	0.1971	0	0.1865
0.25	0.1	415	0.2839	0	0.2547
0.3	0.1	339	0.3430	226	0.3201
0.35	0.1	314	0.4017	242	0.3720
0.4	0.1	306	0.4590	241	0.4327
0.45	0.1	75	0.5445	253	0.5078
0.5	0.1	0	0.6167	266	0.5751
0.55	0.1	0	0.6724	276	0.6247
0.6	0.1	0	0.7360	287	0.6878
0.65	0.1	0	0.7773	300	0.7406
0.15	0.15	0	0.1225	0	0.1205
0.2	0.15	**420**	0.1851	0	0.1782
0.25	0.15	415	0.2577	0	0.2414
0.3	0.15	339	0.3071	165	0.2899
0.35	0.15	314	0.4050	254	0.3779
0.4	0.15	306	0.4329	251	0.4098
0.45	0.15	0	0.5111	264	0.4817
0.5	0.15	0	0.5628	271	0.5330
0.55	0.15	0	0.6685	284	0.6302

0.6	0.15	0	0.6893	296	0.6543
0.65	0.15	0	0.7932	307	0.7586
0.2	0.2	**420**	0.1970	0	0.1922
0.25	0.2	415	0.2583	0	0.2461
0.3	0.2	339	0.4054	8	0.3821
0.35	0.2	314	0.3793	263	0.3499
0.4	0.2	279	0.4498	265	0.4229
0.45	0.2	0	0.5088	271	0.4782
0.5	0.2	0	0.7214	277	0.6819
0.55	0.2	0	0.6321	289	0.6013
0.6	0.2	0	0.7260	301	0.6789
0.65	0.2	0	0.7291	313	0.6856
0.25	0.25	415	0.2460	0	0.2363
0.3	0.25	339	0.3087	2	0.2903
0.35	0.25	314	0.3441	268	0.3246
0.4	0.25	49	0.4092	273	0.3720
0.45	0.25	0	0.5001	275	0.4612
0.5	0.25	0	0.5432	283	0.5112
0.55	0.25	0	0.6146	298	0.5730
0.6	0.25	0	0.6797	308	0.6326
0.65	0.25	0	0.7342	323	0.6871
0.3	0.3	339	0.3078	0	0.2938
0.35	0.3	314	0.3620	272	0.3445
0.4	0.3	0	0.4060	278	0.3815
0.45	0.3	0	0.4559	282	0.4394
0.5	0.3	0	0.5411	289	0.5035
0.55	0.3	0	0.5608	302	0.5227
0.6	0.3	0	0.6716	314	0.6280
0.65	0.3	0	0.6893	312	0.6586
0.35	0.35	314	0.3374	279	0.3194
0.4	0.35	0	0.4085	283	0.3832
0.45	0.35	0	0.4379	288	0.4122
0.5	0.35	0	0.5098	297	0.4827
0.55	0.35	0	0.7151	309	0.6815
0.6	0.35	0	0.6316	**324**	0.6000
0.65	0.35	0	0.7217	18	0.6840
0.4	0.4	0	0.3934	288	0.3778
0.45	0.4	0	0.4529	296	0.4225
0.5	0.4	0	0.5069	303	0.4763
0.55	0.4	0	0.7150	315	0.6790
0.6	0.4	0	0.6126	321	0.5648

0.65	0.4	0	0.6658	9	0.6348
0.45	0.45	0	0.4113	298	0.3820
0.5	0.45	0	0.4620	305	0.4261
0.55	0.45	0	0.5418	323	0.4923
0.6	0.45	0	0.5635	24	0.5270
0.65	0.45	0	0.6668	5	0.6036
0.5	0.5	0	0.4275	315	0.4079
0.55	0.5	0	0.5124	**324**	0.4770
0.6	0.5	0	0.7150	10	0.6791
0.65	0.5	0	0.6038	3	0.5698
0.55	0.55	0	0.5036	211	0.4774
0.6	0.55	0	0.5395	5	0.5101
0.65	0.55	0	0.5957	0	0.5596
0.6	0.6	0	0.5003	4	0.4722
0.65	0.6	0	0.7013	0	0.6724
0.65	0.65	0	0.5425	0	0.5111

Table A.5: Cross-Covariance Evaluation Summary

		Test Set 1		Test Set 2	
Expected Overlap	**Minimal Overlap**	**Correct**	**Avg Exec Time**	**Correct**	**Avg Exec Time**
0.1	0.1	0	0.0828	0	0.0810
0.15	0.1	0	0.1344	0	0.1363
0.2	0.1	**420**	0.2017	0	0.2009
0.25	0.1	413	0.2853	0	0.2752
0.3	0.1	329	0.3412	327	0.3289
0.35	0.1	307	0.3965	352	0.3867
0.4	0.1	299	0.4629	351	0.4498
0.45	0.1	73	0.5437	343	0.5360
0.5	0.1	0	0.6120	331	0.5941
0.55	0.1	0	0.6670	323	0.6720
0.6	0.1	0	0.7352	316	0.7226
0.65	0.1	0	0.7881	315	0.7772
0.15	0.15	0	0.1268	0	0.1354
0.2	0.15	**420**	0.1869	0	0.1941
0.25	0.15	413	0.2511	0	0.2615
0.3	0.15	329	0.3094	230	0.2936
0.35	0.15	307	0.4022	**353**	0.3789
0.4	0.15	299	0.4377	351	0.4069
0.45	0.15	0	0.5118	343	0.4749

0.5	0.15	0	0.5655	331	0.5240
0.55	0.15	0	0.6741	323	0.6366
0.6	0.15	0	0.6985	316	0.6561
0.65	0.15	0	0.8008	315	0.7473
0.2	0.2	**420**	0.1988	0	0.1881
0.25	0.2	413	0.2513	0	0.2417
0.3	0.2	329	0.4046	8	0.3812
0.35	0.2	307	0.3757	350	0.3456
0.4	0.2	272	0.4496	351	0.4147
0.45	0.2	0	0.5099	343	0.4795
0.5	0.2	0	0.7177	331	0.6780
0.55	0.2	0	0.6374	323	0.6113
0.6	0.2	0	0.7184	316	0.6854
0.65	0.2	0	0.7373	315	0.6929
0.25	0.25	413	0.2523	0	0.2393
0.3	0.25	329	0.3042	1	0.2904
0.35	0.25	307	0.3511	348	0.3247
0.4	0.25	45	0.4062	351	0.3729
0.45	0.25	0	0.5054	343	0.4670
0.5	0.25	0	0.5490	331	0.5101
0.55	0.25	0	0.6139	323	0.5606
0.6	0.25	0	0.6773	316	0.6295
0.65	0.25	0	0.7408	315	0.6848
0.3	0.3	329	0.3006	0	0.2878
0.35	0.3	307	0.3686	347	0.3426
0.4	0.3	0	0.4125	351	0.3813
0.45	0.3	0	0.4681	343	0.4379
0.5	0.3	0	0.5445	331	0.5093
0.55	0.3	0	0.5717	323	0.5306
0.6	0.3	0	0.6790	316	0.6296
0.65	0.3	0	0.7121	303	0.6656
0.35	0.35	307	0.3450	347	0.3207
0.4	0.35	0	0.4068	351	0.3840
0.45	0.35	0	0.4359	343	0.4097
0.5	0.35	0	0.5169	331	0.4797
0.55	0.35	0	0.7187	323	0.6741
0.6	0.35	0	0.6468	317	0.6061
0.65	0.35	0	0.7226	17	0.6755
0.4	0.4	0	0.4027	351	0.3765
0.45	0.4	0	0.4553	343	0.4186
0.5	0.4	0	0.5135	331	0.4789

0.55	0.4	0	0.7134	323	0.6765
0.6	0.4	0	0.6205	314	0.5708
0.65	0.4	0	0.6748	8	0.6315
0.45	0.45	0	0.4103	343	0.3860
0.5	0.45	0	0.4679	331	0.4361
0.55	0.45	0	0.5433	323	0.5095
0.6	0.45	0	0.5646	25	0.5318
0.65	0.45	0	0.6757	4	0.6391
0.5	0.5	0	0.4360	331	0.4077
0.55	0.5	0	0.5158	322	0.4786
0.6	0.5	0	0.7113	9	0.6810
0.65	0.5	0	0.6108	2	0.5694
0.55	0.55	0	0.5107	210	0.4768
0.6	0.55	0	0.5481	4	0.5139
0.65	0.55	0	0.6070	0	0.5638
0.6	0.6	0	0.5111	3	0.4825
0.65	0.6	0	0.7151	0	0.6815
0.65	0.65	0	0.5494	0	0.5145

Table A.6: Sum of Squared Differences Evaluation Summary

		Test Set 1		Test Set 2	
Expected Overlap	**Minimal Overlap**	**Correct**	**Avg Exec Time**	**Correct**	**Avg Exec Time**
0.1	0.1	0	0.0792	0	0.0774
0.15	0.1	0	0.1353	0	0.1334
0.2	0.1	**420**	0.1977	0	0.1993
0.25	0.1	415	0.2795	0	0.2697
0.3	0.1	339	0.3394	230	0.3386
0.35	0.1	312	0.3908	247	0.3985
0.4	0.1	306	0.4592	243	0.4601
0.45	0.1	75	0.5442	258	0.5438
0.5	0.1	0	0.6064	267	0.6004
0.55	0.1	0	0.6813	278	0.6679
0.6	0.1	0	0.7289	289	0.7359
0.65	0.1	0	0.7691	302	0.7788
0.15	0.15	0	0.1266	0	0.1237
0.2	0.15	**420**	0.1836	0	0.1862
0.25	0.15	415	0.2515	0	0.2593
0.3	0.15	339	0.3097	169	0.3101
0.35	0.15	312	0.4002	259	0.3951

0.4	0.15	306	0.4301	252	0.4282
0.45	0.15	0	0.5137	268	0.5129
0.5	0.15	0	0.5646	274	0.5643
0.55	0.15	0	0.6725	286	0.6692
0.6	0.15	0	0.7018	299	0.7041
0.65	0.15	0	0.7971	313	0.7988
0.2	0.2	**420**	0.1929	0	0.1957
0.25	0.2	415	0.2560	0	0.2604
0.3	0.2	339	0.3981	8	0.4000
0.35	0.2	312	0.3647	265	0.3729
0.4	0.2	279	0.4458	267	0.4537
0.45	0.2	0	0.5087	273	0.5025
0.5	0.2	0	0.7090	281	0.7077
0.55	0.2	0	0.6355	290	0.6407
0.6	0.2	0	0.7198	302	0.7228
0.65	0.2	0	0.7315	316	0.7342
0.25	0.25	415	0.2438	0	0.2509
0.3	0.25	339	0.3092	2	0.3033
0.35	0.25	312	0.3429	270	0.3480
0.4	0.25	49	0.4014	275	0.4028
0.45	0.25	0	0.4957	278	0.4960
0.5	0.25	0	0.5437	287	0.5425
0.55	0.25	0	0.6075	300	0.6123
0.6	0.25	0	0.6668	314	0.6769
0.65	0.25	0	0.7294	**325**	0.7314
0.3	0.3	339	0.3021	0	0.3042
0.35	0.3	312	0.3631	273	0.3606
0.4	0.3	0	0.4043	280	0.4088
0.45	0.3	0	0.4619	284	0.4600
0.5	0.3	0	0.5397	292	0.5479
0.55	0.3	0	0.5613	303	0.5618
0.6	0.3	0	0.6651	316	0.6720
0.65	0.3	0	0.6920	314	0.7002
0.35	0.35	312	0.3345	280	0.3370
0.4	0.35	0	0.4021	287	0.4023
0.45	0.35	0	0.4338	290	0.4369
0.5	0.35	0	0.5039	300	0.5088
0.55	0.35	0	0.7142	313	0.7080
0.6	0.35	0	0.6328	**325**	0.6386
0.65	0.35	0	0.7149	18	0.7268
0.4	0.4	0	0.4022	290	0.4013

0.45	0.4	0	0.4474	297	0.4483
0.5	0.4	0	0.5019	303	0.5113
0.55	0.4	0	0.7131	316	0.7177
0.6	0.4	0	0.6135	322	0.6099
0.65	0.4	0	0.6733	9	0.6759
0.45	0.45	0	0.4086	300	0.4079
0.5	0.45	0	0.4579	306	0.4664
0.55	0.45	0	0.5436	324	0.5430
0.6	0.45	0	0.5630	24	0.5609
0.65	0.45	0	0.6717	5	0.6717
0.5	0.5	0	0.4364	316	0.4344
0.55	0.5	0	0.5071	**325**	0.5079
0.6	0.5	0	0.7080	10	0.7142
0.65	0.5	0	0.6079	3	0.6162
0.55	0.55	0	0.5039	212	0.5136
0.6	0.55	0	0.5434	5	0.5452
0.65	0.55	0	0.5949	0	0.5966
0.6	0.6	0	0.5086	4	0.5159
0.65	0.6	0	0.7170	0	0.7236
0.65	0.65	0	0.5430	0	0.5448

List of Figures

List of Tables

Bibliography

[1] Randy Torrance and Dick James. "The State-of-the-Art in Semiconductor Reverse Engineering". In: *Proceedings of the 48th Design Automation Conference*. DAC '11. San Diego, California: Association for Computing Machinery, 2011, 333–338. ISBN: 9781450306362. DOI: 10.1145/2024724.2024805. URL: https://doi.org/10.1145/2024724.2024805.

[2] Randy Torrance and Dick James. "The State-of-the-Art in IC Reverse Engineering". In: *Cryptographic Hardware and Embedded Systems - CHES 2009*. Ed. by Christophe Clavier and Kris Gaj. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 363–381. ISBN: 978-3-642-04138-9.

[3] TSS Microscopy. *Scanning Electron Microscopes, Browse our currently available inventory of pre-owned, ReManufactured SEM microscopes for sale. Refine results by searching for brand and price. If you don't see what you need, please contact us, as we may have it in-house or know where we can find it.* https://tssmicroscopy.com/instruments/categories/scanning-electron-microscopes/. Aug. 2021. (Visited on 08/27/2021).

[4] A. V. Crewe et al. "Electron Gun Using a Field Emission Source". In: *Review of Scientific Instruments* 39.4 (1968), pp. 576–583. DOI: 10.1063/1.1683435. eprint: https://doi.org/10.1063/1.1683435. URL: https://doi.org/10.1063/1.1683435.

[5] P. J. Burt. "The Pyramid as a Structure for Efficient Computation". In: *Multiresolution Image Processing and Analysis*. Ed. by Azriel Rosenfeld. Berlin, Heidelberg: Springer Berlin Heidelberg, 1984, pp. 6–35. ISBN: 978-3-642-51590-3. DOI: 10.1007/978-3-642-51590-3_2. URL: https://doi.org/10.1007/978-3-642-51590-3_2.

[6] D.G. Lowe. "Object recognition from local scale-invariant features". In: *Proceedings of the Seventh IEEE International Conference on Computer Vision*. Vol. 2. 1999, 1150–1157 vol.2. DOI: 10.1109/ICCV.1999.790410.

[7] Ethan Rublee et al. "ORB: An efficient alternative to SIFT or SURF". In: *2011 International Conference on Computer Vision*. 2011, pp. 2564–2571. DOI: 10.1109/ICCV.2011.6126544.

[8] Michael D Abràmoff, Paulo J Magalhães, and Sunanda J Ram. “Image processing with ImageJ”. In: *Biophotonics international* 11.7 (2004), pp. 36–42.

[9] Hanchuan Peng. “Bioimage informatics: a new area of engineering biology”. In: *Bioinformatics* 24.17 (July 2008), pp. 1827–1836. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btn346. eprint: https://academic.oup.com/bioinformatics/article-pdf/24/17/1827/16882564/btn346.pdf. URL: https://doi.org/10.1093/bioinformatics/btn346.

[10] Swarup Bhunia et al. “Hardware Trojan Attacks: Threat Analysis and Countermeasures”. In: *Proceedings of the IEEE* 102.8 (2014), pp. 1229–1247. DOI: 10.1109/JPROC.2014.2334493.

[11] Bernhard Lippmann et al. “Integrated Flow for Reverse Engineering of Nanoscale Technologies”. In: *Proceedings of the 24th Asia and South Pacific Design Automation Conference*. ASPDAC ’19. Tokyo, Japan: Association for Computing Machinery, 2019, 82–89. ISBN: 9781450360074. DOI: 10.1145/3287624.3288738. URL: https://doi.org/10.1145/3287624.3288738.

[12] Marc Fyrbiak et al. “Hardware reverse engineering: Overview and open challenges”. In: *2017 IEEE 2nd International Verification and Security Workshop (IVSW)*. 2017, pp. 88–94. DOI: 10.1109/IVSW.2017.8031550.

[13] Michael Bajura et al. “Imaging Integrated Circuits with X-ray Microscopy”. In: Apr. 2011.

[14] Junjing Deng et al. “Nanoscale x-ray imaging of circuit features without wafer etching”. In: *Phys. Rev. B* 95 (10 2017), p. 104111. DOI: 10.1103/PhysRevB.95.104111. URL: https://link.aps.org/doi/10.1103/PhysRevB.95.104111.

[15] Shahed E. Quadir et al. “A Survey on Chip to System Reverse Engineering”. In: *J. Emerg. Technol. Comput. Syst.* 13.1 (Apr. 2016). ISSN: 1550-4832. DOI: 10.1145/2755563. URL: https://doi.org/10.1145/2755563.

[16] Raul Quijada et al. “Large-Area Automated Layout Extraction Methodology for Full-IC Reverse Engineering”. In: *Journal of Hardware and Systems Security* 2.4 (2018), pp. 322–332. ISSN: 2509-3436. DOI: 10.1007/s41635-018-0051-4. URL: https://doi.org/10.1007/s41635-018-0051-4.

[17] Sebastian Wallat et al. “Highway to HAL: Open-Sourcing the First Extendable Gate-Level Netlist Reverse Engineering Framework”. In: *Proceedings of the 16th ACM International Conference on Computing Frontiers*. CF ’19. Alghero, Italy: Association for Computing Machinery, 2019, 392–397. ISBN: 9781450366854. DOI: 10.1145/3310273.3323419. URL: https://doi.org/10.1145/3310273.3323419.

[18] Pramod Subramanyan et al. “Reverse Engineering Digital Circuits Using Structural and Functional Analyses”. In: *IEEE Transactions on Emerging Topics in Computing* 2.1 (2014), pp. 63–80. DOI: 10.1109/TETC.2013.2294918.

[19] Michael Werner et al. “Reverse Engineering of Cryptographic Cores by Structural Interpretation Through Graph Analysis”. In: *2018 IEEE 3rd International Verification and Security Workshop (IVSW)*. 2018, pp. 13–18. DOI: 10.1109/IVSW.2018.8494896.

[20] Yiqiong Shi et al. “Extracting functional modules from flattened gate-level netlist”. In: *2012 International Symposium on Communications and Information Technologies (ISCIT)*. 2012, pp. 538–543. DOI: 10.1109/ISCIT.2012.6380958.

[21] J.L. White et al. “Efficient algorithms for subcircuit enumeration and classification for the module identification problem”. In: *Proceedings 2001 IEEE International Conference on Computer Design: VLSI in Computers and Processors. ICCD 2001.* 2001, pp. 519–522. DOI: 10.1109/ICCD.2001.955082.

[22] L Reimer. “Scanning Electron Microscopy: Physics of Image Formation and Microanalysis, Second Edition”. In: *Measurement Science and Technology* 11.12 (2000), pp. 1826–1826. DOI: 10.1088/0957-0233/11/12/703. URL: https://doi.org/10.1088/0957-0233/11/12/703.

[23] H Seiler. “Secondary electron emission in the scanning electron microscope”. In: *Journal of Applied Physics* 54.11 (1983), R1–R18. DOI: 10.1063/1.332840. eprint: https://doi.org/10.1063/1.332840. URL: https://doi.org/10.1063/1.332840.

[24] Michael T. Postek and András E. Vladár. “Does Your SEM Really Tell the Truth?-How Would You Know? Part 4: Charging and its Mitigation”. eng. In: *Proceedings of SPIE–the International Society for Optical Engineering* 9636 (2015). 28663665[pmid], 963605 (October 21, 2015); doi:10.1117/12.2195344 Text Size: A A A. ISSN: 0277-786X. DOI: 10.1117/12.2195344. URL: https://pubmed.ncbi.nlm.nih.gov/28663665.

[25] Goldstein J.I. et al. “Electron-Beam-Specimen Interactions”. In: *Scanning Electron Microscopy and X-Ray Microanalysis*. Springer, Boston, MA, 1981, pp. 53–122. ISBN: 978-1-4613-3273-2. DOI: https://doi.org/10.1007/978-1-4613-3273-2_3.

[26] F. Timischl, M. Date, and S. Nemoto. “A statistical model of signal–noise in scanning electron microscopy”. In: *Scanning* 34.3 (2012), pp. 137–144. DOI: https://doi.org/10.1002/sca.20282. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/sca.20282. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/sca.20282.

[27] J. HEJNA. “Noise coefficients of backscattered electron detectors for low voltage scanning electron microscopy”. In: *Journal of Microscopy* 252.1 (2013), pp. 35–48. DOI: https://doi.org/10.1111/jmi.12066. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/jmi.12066. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/jmi.12066.

[28] Naresh Marturi, Sounkalo Dembélé, and Nadine Piat. "Scanning electron microscope image signal-to-noise ratio monitoring for micro-nanomanipulation." In: *The Journal of Scanning Microscopies.* (2014), pp. 1–11. DOI: 10.1002/sca.21137. URL: https://hal.archives-ouvertes.fr/hal-01051309.

[29] W. Z. Wan Ismail et al. "Reducing charging effects in scanning electron microscope images by Rayleigh contrast stretching method (RCS)". In: *Scanning* 33.4 (2011), pp. 233–251. DOI: https://doi.org/10.1002/sca.20237. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/sca.20237. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/sca.20237.

[30] Kazuhiko Suzuki and Eisaku Oho. "Special raster scanning for reduction of charging effects in scanning electron microscopy". In: *Scanning* 36.3 (2014), pp. 327–333. DOI: https://doi.org/10.1002/sca.21112. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/sca.21112. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/sca.21112.

[31] Mark Tsuchida. *μManager.* https://micro-manager.org/. (Visited on 08/27/2021).

[32] Joe Chalfoun et al. "MIST: Accurate and Scalable Microscopy Image Stitching Tool with Stage Modeling and Error Minimization". In: *Scientific Reports* 7.1 (2017), p. 4988. ISSN: 2045-2322. DOI: 10.1038/s41598-017-04567-y. URL: https://doi.org/10.1038/s41598-017-04567-y.

[33] Stephan Preibisch, Stephan Saalfeld, and Pavel Tomancak. "Globally optimal stitching of tiled 3D microscopic image acquisitions". In: *Bioinformatics* 25.11 (Apr. 2009), pp. 1463–1465. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btp184. eprint: https://academic.oup.com/bioinformatics/article-pdf/25/11/1463/950295/btp184.pdf. URL: https://doi.org/10.1093/bioinformatics/btp184.

[34] Martin Schobert. *Degate.* https://degate.org/status.html. (Visited on 07/02/2021).

[35] Martin Schobert Dorian Bachelot. *Degate.* https://github.com/DegateCommunity/Degate/. (Visited on 07/02/2021).

[36] Texplained. *Texplained-ChipJuice-Presentation-English-2019.* https://www.texplained.com/download/chipjuice-presentation. (Visited on 07/02/2021).

[37] Embedded Security Group. *HAL - The Hardware Analyzer.* https://github.com/emsec/hal. 2019.

[38] Johannes Schindelin et al. "Fiji: an open-source platform for biological-image analysis". In: *Nature Methods* 9.7 (2012), pp. 676–682. ISSN: 1548-7105. DOI: 10.1038/nmeth.2019. URL: https://doi.org/10.1038/nmeth.2019.

[39] David Hörl et al. “BigStitcher: reconstructing high-resolution image datasets of cleared and expanded samples”. In: *Nature Methods* 16.9 (2019), pp. 870–874. ISSN: 1548-7105. DOI: 10.1038/s41592-019-0501-0. URL: https://doi.org/10.1038/s41592-019-0501-0.

[40] Joe Chalfoun. “A power stitching tool”. In: *SPIE Newsroom* (2014).

[41] Timothy Blattner et al. “A hybrid CPU-GPU system for stitching large scale optical microscopy images”. In: *2014 43rd International Conference on Parallel Processing*. IEEE. 2014, pp. 1–9.

[42] Anna Wójcicka and Zygmunt Wróbel. “The Panoramic Visualization of Metallic Materials in Macro- and Microstructure of Surface Analysis Using Microsoft Image Composite Editor (ICE)”. In: *Information Technologies in Biomedicine*. Ed. by Ewa Piętka and Jacek Kawa. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 358–368. ISBN: 978-3-642-31196-3.

[43] Georg von Arx. *Stitching distortion-free mosaic images for QWA using PTGui*. https://www.wsl.ch/fileadmin/user_upload/WSL/Services_Produkte/Software_Apps/Roxas/ptgui_quickguide.pdf. Feb. 2017. (Visited on 05/07/2021).

[44] Teorex. *How to Seamlessly Stitch Microscope Images Together*. https://www.photostitcher.com/how-to-stitch-microscope-images.html. (Visited on 05/07/2021).

[45] Ahmad P. Tafti et al. “A comparative study on the application of SIFT, SURF, BRIEF and ORB for 3D surface reconstruction of electron microscopy images”. In: *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization* 6.1 (2018), pp. 17–30. DOI: 10.1080/21681163.2016.1152201. eprint: https://doi.org/10.1080/21681163.2016.1152201. URL: https://doi.org/10.1080/21681163.2016.1152201.

[46] Hsiang-Jen Chien et al. “When to use what feature? SIFT, SURF, ORB, or A-KAZE features for monocular visual odometry”. In: *2016 International Conference on Image and Vision Computing New Zealand (IVCNZ)*. 2016, pp. 1–6. DOI: 10.1109/IVCNZ.2016.7804434.

[47] Ebrahim Karami, Siva Prasad, and Mohamed Shehata. *Image Matching Using SIFT, SURF, BRIEF and ORB: Performance Comparison for Distorted Images*. 2017. arXiv: 1710.02726 [cs.CV].

[48] Nabeel Khan, Brendan McCane, and Steven Mills. “Better than SIFT?” In: *Machine Vision and Applications* 26.6 (2015), pp. 819–836. ISSN: 1432-1769. DOI: 10.1007/s00138-015-0689-7. URL: https://doi.org/10.1007/s00138-015-0689-7.

[49] Shimiao Li. “A review of feature detection and match algorithms for localization and mapping”. In: *IOP Conference Series: Materials Science and Engineering* 231 (2017), p. 012003. DOI: 10.1088/1757-899x/231/1/012003. URL: https://doi.org/10.1088/1757-899x/231/1/012003.

[50] D. G. Lowe. “Object recognition from local scale-invariant features”. In: *Proceedings of the Seventh IEEE International Conference on Computer Vision*. Vol. 2. 1999, 1150–1157 vol.2. DOI: 10.1109/ICCV.1999.790410.

[51] Edward Rosten and Tom Drummond. “Machine Learning for High-Speed Corner Detection”. In: *Computer Vision – ECCV 2006*. Ed. by Aleš Leonardis, Horst Bischof, and Axel Pinz. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 430–443. ISBN: 978-3-540-33833-8.

[52] Michael Calonder et al. “BRIEF: Binary Robust Independent Elementary Features”. In: *Computer Vision – ECCV 2010*. Ed. by Kostas Daniilidis, Petros Maragos, and Nikos Paragios. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 778–792. ISBN: 978-3-642-15561-1.

[53] Ethan Rublee et al. “ORB: An efficient alternative to SIFT or SURF”. In: *Proceedings of the 2011 International Conference on Computer Vision*. IEEE Computer Society. 2011, pp. 2564–2571.

[54] Paul L. Rosin. “Measuring Corner Properties”. In: *Computer Vision and Image Understanding* 73.2 (1999), pp. 291 –307. ISSN: 1077-3142. DOI: https://doi.org/10.1006/cviu.1998.0719. URL: http://www.sciencedirect.com/science/article/pii/S1077314298907196.

[55] C. Ding and X. Tang. “The Cross-Correlation of Binary Sequences With Optimal Autocorrelation”. In: *IEEE Transactions on Information Theory* 56.4 (2010), pp. 1694–1701. DOI: 10.1109/TIT.2010.2040883.

[56] Margaret J. Mayston, Linda M. Harrison, and John A. Stephens. “A neurophysiological study of mirror movements in adults and children”. In: *Annals of Neurology* 45.5 (1999), pp. 583–594. DOI: 10.1002/1531-8249(199905)45:5<583::AID-ANA6>3.0.CO;2-W. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/1531-8249%28199905%2945%3A5%3C583%3A%3AAID-ANA6%3E3.0.CO%3B2-W. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/1531-8249%28199905%2945%3A5%3C583%3A%3AAID-ANA6%3E3.0.CO%3B2-W.