MASTERARBEIT | MASTER'S THESIS

Titel | Title

Pattern Recognition in Quasicrystalline Structures

verfasst von | submitted by

Tano Kim Kender BSc

angestrebter akademischer Grad | in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien | Vienna, 2024

Studienkennzahl lt. Studienblatt | Degree programme code as it appears on the student record sheet:	UA 066 910
Studienrichtung lt. Studienblatt | Degree programme as it appears on the student record sheet:	Masterstudium Computational Science
Betreut von | Supervisor:	Univ.-Prof. Dr.techn. Cesare Franchini

Acknowledgements

I would like to thank Marco Corrias for all his guidance and support during the project. I also want to thank Cesare Franchini for his encouragement and patience. Lastly, thank you to Stefan Förster, Weizhen Wang and Goran Ungar for providing images for testing.

Abstract

The discovery of quasicrystals in 1984 redefined the essence of crystalline materials. Due to their lack of translational periodicity, the atom arrangements in quasicrystals cannot be modelled by lattices, but rather by quasiperiodic tilings. This makes the analysis of atomic resolution images of quasicrystal surfaces an arduous task, as no currently available lattice recognition software can be applied. This project introduces a method to detect those tilings using image feature recognition coupled with machine learning, tailored towards quasiperiodicity with 8-, 10- and 12-fold rotational symmetry. Atom positions are identified using clustering of feature descriptors. Subsequent nearest neighbour analysis and border following on the inter-atom connections deliver the tiling. Using test images of different quasicrystals produced by various techniques, a generalised pattern recognition software was developed, able to consistently and correctly identify the atomic arrangements. With no requirement for prior expertise to use, this offers a significant boost to the speed of the surface analysis process for any polygonal quasicrystal image.

Kurzfassung

Die Entdeckung von Quasikristallen in 1984 führte zu einer neuen Definition kristalliner Festkörper. Dadurch, dass sie keine translatorische Periodizität aufweisen, kann die Atomverteilung in Quasikristallen nicht durch Gitter modelliert werden. Stattdessen wird es durch ein quasiperiodische Parkettierung charakterisiert. Das erschwert die Untersuchung der Oberflächen in Quasikristallen, da aktuelle Kristallgittererkennungssoftware nicht dafür geeignet sind. Diese Projekt befasst sich mit der Erstellung einer Parkettierungserkennung auf Basis von Bildmerkmalen und maschinellem Lernen, speziell zugeschnitten auf quasiperiodische Muster mit 8-, 10- und 12-facher Drehsymmetrie. Die Positionen der Atome werden anhand von Ballungsanalyse der Deskriptoren der Merkmale bestimmt. Darauf folgende "Nearest-Neighbour"-auswertung und die dadurch entstehenden Verbindungen zwischen den Atomen liefert mithilfe von Randerkennung die Parkettierung. Die Entwicklung wird mit Kontrollbildern verschiedener Quasikristalle von verschidenenen Mikroskopiermethoden unterstützt, wodurch eine allgemein beständige und präzise Mustererkennungssoftware für Quasikristalle zustande kommt. Diese ohne Vorkenntnis anwendbare Software und die damit verbundene Vorgehensweise beschleunigt somit den Untersuchungsprozess für alle Oberflächen-quasikristalle.

Contents

List of Figures

1. Introduction

Since their discovery 40 years ago, an accomplishment awarded with a Nobel Prize in 2011, quasicrystals (QC) have been researched in depth to the point of potential commercial application. As structure and symmetry is an important aspect of crystals, separating them from all other solids, it would logically follow that such is the case for QCs as well. While efforts are underway to automate lattice recognition in crystals, the same cannot be said for QC. Although there are only a few well-resolved images of QC surfaces, there is nothing available to facilitate in recognizing atom arrangements within. Getting the QC tiling from an image takes time and manual labour.

This project aims to build upon existing approaches for crystal lattice recognition and expand them to quasiperiodicity. The QC pattern recognition software will aid in surface symmetry analysis, automating a tedious process and thus saving time and resources. It will be used to expand the AiSurf package[1] to quasiperiodicity.

1.1. Quasicrystals

1.1.1. Symmetry in Crystals

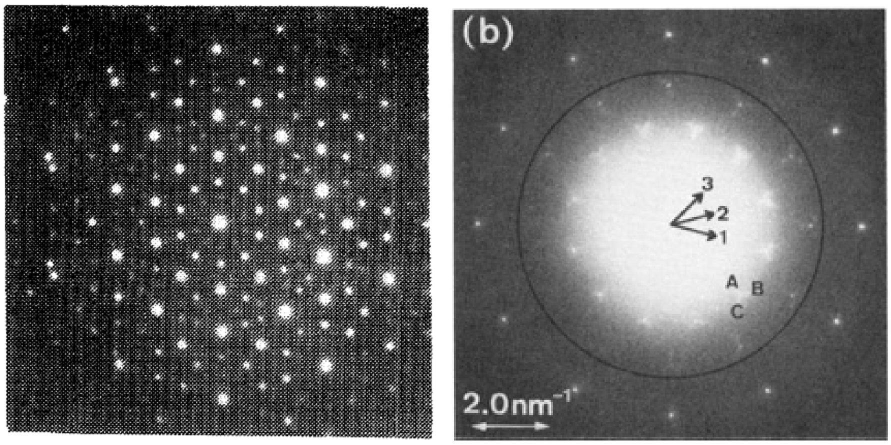
Solid states of matter used to be divided into two categories, amorphous and crystalline. The latter used to be defined by the periodic ordering of its components. Crystalline structures can be broken down into unit cells stacked together without gaps. The unit cell contains the basis, which is a formation of atoms that recurs across the crystal at set points. These points are called lattice points and their relative positions to each other are determined by lattice vectors, which are also used to describe the unit cell. Moving the unit cells along these vectors does not change the crystal structure, and is therefore periodic with translational symmetry.

Furthermore, crystals possess rotational symmetry. An object with n-fold rotational symmetry can face in n spatial directions along a plane and remain invariant. Specifically, only 1-, 2-, 3-, 4- and 6-fold rotational symmetry can be encountered in crystals. This is because unit cells with rotational symmetry of 5 or > 6 cannot be ordered such no gaps remain.

The rotational symmetry can be investigated by diffraction imaging, using X-rays or electron beams. When passing through a crystal they get diffracted, causing constructive or destructive interference resulting in a pattern that can be used to describe the crystal structure. This pattern type can only be found in the presence of symmetry, and as such is unique to crystalline matter.

1.1.2. Symmetry in Quasicrystals

In 1984, a paper was published on the findings of an Al-Mn alloy with diffraction showing 10-fold rotational symmetry with no translational periodicity along any axis, the diffraction pattern of which can be seen in figure 1.1a [4]. As such, it was not amorphous but also



(a) (b)

Figure 1.1.: **(a)** One of the first electron diffraction patterns of a QC published in Shechtman et al. (1984). Rings of 10 bright dots each can be seen, indicating "forbidden" rotational symmetry. **(b)**The first dodecagonal QC diffraction pattern seen in small NiCr particles, published in Ishimasa et al. (1985).

not a crystal by definition[1]. Therefore, this new material was named "Quasicrystal", short for "quasiperiodic crystal". There are two types of QC known today, icosahedral QC with no translational periodicity along any axis and polygonal quasicrystals with one periodic axis and quasiperodicity in the perpendicular plane. The latter can possess 8-, 10- or even 12-fold rotational symmetry. It was only later in the same year (1984) that the first dodecagonal QC was found, with it's diffraction pattern showing 12 distinct spots, seen in figure 1.1b.

The rotational symmetry seen in QC diffractions can be understood as the projection of a diffraction resulting from a lattice in a higher dimension. If one would take an 4-dimensional lattice and project it at an irrational angle onto an $\frac{n}{2}$-dimensional space, the result would a quasiperiodic pattern in $\frac{n}{2}$D. For example, a 12-fold rotationally symmetric pattern in 2D can be characterized by 4 linearly independent vectors in the reciprocal space, which are the projections of the unit vectors of a reciprocal lattice in 4D [4].

Tilings

Symmetry in diffraction indicates the presence of a pattern in the distribution of atoms. Instead of a lattice, the atoms are arranged as an aperiodic tiling. These tilings can consist of 2 or more primitive tiles arranged so that an aperiodic gap-less coverage is achieved. Fourier transforming the vertices in these tilings shows symmetry in the reciprocal space in accordance with the diffraction patterns from QC.

[1] The definition was later changed such that any materials showing diffraction patterns are categorised as crystalline solids.[5]

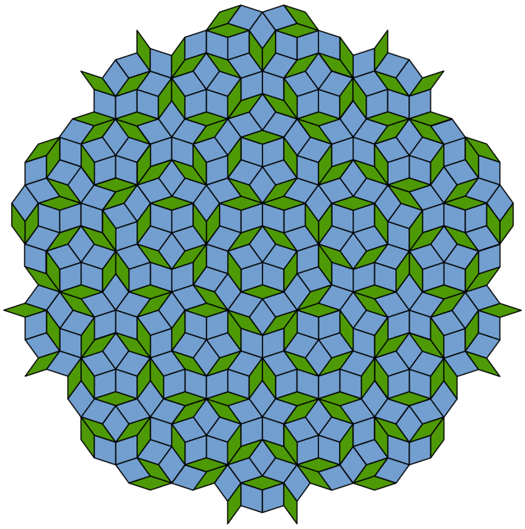


Figure 1.2.: Example of a Penrose tiling, composed of thick and thin rhombic tiles. *Source: Inductiveload, Public Domain, via Wikimedia Commons.*

A well-known example of such a tiling are the Penrose tilings. The original basis of these tilings were pentagons, pentacles, half-pentacles and rhombs [6]. Their placement was decided by matching rules indicating which tiles can touch each other at which edges [6]. These placement rules are set so that quasiperiodicity is upheld and the plane can be tiled without gaps [7].

Furthermore, the vertices of the tiling can be used to impose a super-tiling with the same shapes in a size ratio of $1 : \tau$ where $\tau = \frac{1+\sqrt{5}}{2}$ is the golden ratio. This golden ratio can be found again in the rhombic Penrose tiling with a basis of two distinct rhombs, one thick and one thin as seen in figure 1.2. The ratio of the area of the thin rhomb to the area of the thick rhomb is $1 : \tau$. Even the tiling frequency is $1 : \tau$ [7].

1.1.3. Physical Properties

The possible elemental combinations of thermodynamically stable QC are shown in figure 1.3. Although mainly metallic atoms are present, QC show properties not typical of such components. Interestingly, polygonal QC only show these non-metallic properties along the quasiperiodic plane while icosahedral ones exhibit them in all directions [8]. Stable QC with high structural quality tend to be similar to semiconductors in terms of electrical conductivity. In those QC, the conductivity decreases with decreasing temperature [9]. It has been postulated that conductivity could even reach zero for very low temperatures.

The thermal conductivity is comparatively low and is proportional to the temperature. But this linear conductivity relation means that QC are closer to ceramic insulators than metals at very high temperatures [9].

QC are hard and brittle and do not allow any deformation. This hardness means

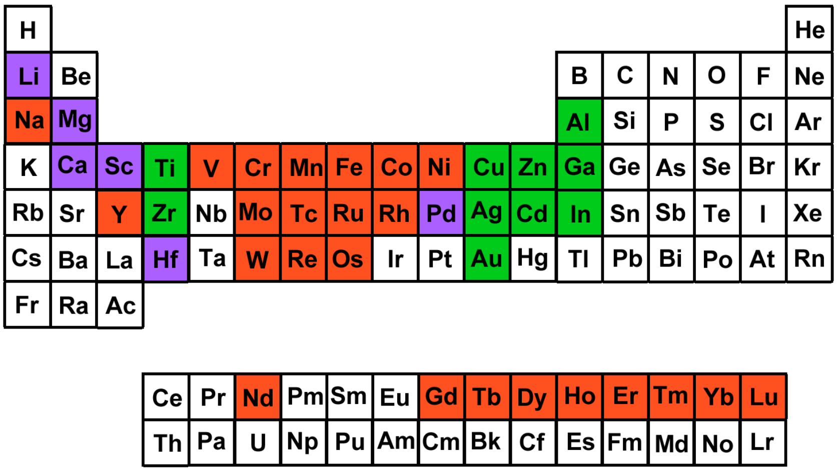

H He
Li Be B C N O F Ne
Na Mg Al Si P S Cl Ar
K Ca Sc Ti V Cr Mn Fe Co Ni Cu Zn Ga Ge As Se Br Kr
Rb Sr Y Zr Nb Mo Tc Ru Rh Pd Ag Cd In Sn Sb Te I Xe
Cs Ba La Hf Ta W Re Os Ir Pt Au Hg Tl Pb Bi Po At Rn
Fr Ra Ac
Ce Pr Nd Pm Sm Eu Gd Tb Dy Ho Er Tm Yb Lu
Th Pa U Np Pu Am Cm Bk Cf Es Fm Md No Lr

Figure 1.3.: Periodic table outtake featuring all the elements contributing in forming QC with thermodynamic stability according to Barber (2019). The green elements are the main components, followed by purple and then orange in order of contribution.

that the surfaces are scratch-resistant with very low friction. Additionally, they have non-stick properties and are resistant to corrosion. With all these properties, QCs are good candidates for industrial application, for example as coating for cookware or equipment for handling concrete [9].

1.2. Pattern Recognition

There exist already multiple approaches of applying computer vision to crystal surface images in order to get atom positions. Together with machine learning, a field that is rapidly growing, this gives rise to automated pattern recognition with which new images can be analysed in a short amount of time. Methods used include cross-correlation with a probe, training a neural network model or just simple feature recognition [10; 11; 12; 13]. The first two methods require preparation and knowledge of the observed material, while the last method can be applied directly in the field for new observations [1]. The feature recognition method uses blob detection (specifically the scale-invariant feature transform (SIFT) algorithm [14]) to detect scale space extrema in the image and classify them using unsupervised learning.

Since there are only few images of QC available, it is not favourable to enlist supervised machine learning. And with the lack of translational periodicity, generating a satisfying probe for cross-correlation may lead to errors. Therefore the feature recognition method is the most promising for an expansion to pattern detection in quasicrystalline matter.

1.2.1. Features and Descriptors

Image features can be points, corners, edges, contours, and so on. They are mainly used for object recognition through matching between two or more images. Features may also help with image alignment for stitching and video stabilisation. Point features are referred to as "keypoints" and are the most useful when it comes to analysing images on the atomic scale, since atoms appear as points (or blobs). Keypoint detection algorithms are therefore sometimes called "blob detectors". While the approaches may be different, they all find areas or points with high contrast changes relative to their surroundings. For each point a "descriptor" is calculated, a vector containing gradient information along all directions in an area around the center of the keypoint. Because the description area depends on the keypoint size and orientation, descriptors are invariant under rotation and scale changes. Therefore they are very valuable for feature matching in object recognition and stabilisation. As these descriptors carry information on their local area, they can be useful in pattern recognition as well.

1.2.2. Clustering

While supervised learning can be helpful in pattern detection, it is required to train them using prior knowledge. If there were many QC images with outlined patterns available, this would be the appropriate approach. But due to the lack of such data, models would most likely under-perform.

Therefore, methods to learn from unlabeled data have to be deployed. An example of such a method would be hierarchical clustering, where objects are grouped and labelled without the need of any information input at all. In general, clustering algorithms join similar data points together based on a similarity measure. Hierarchical clustering may be performed top-down or bottom-up, the former separates points at each step while the latter joins them. The advantage of hierarchical clustering is that for a dataset with n points any grouping from 1 to n clusters can be investigated in only one algorithm execution [15].

Similar to the existing feature-based approaches for crystals ([12; 1]), clustering and image features are used together to extract the tiling from images of QC surfaces in this project.

2. Methods

2.1. Package Structure

For reference purposes, the package was named **QCP**, which stands for "**Q**uasi**C**rystal **P**attern extractor". The programming language used for QCP is Python 3. The purpose of QCP is to facilitate quasicrystal pattern detection and as such was designed to be effective and easy to use. The methods used are packed into a class with a default setup, making it easy to call and allowing access to all the data used along the way in the form of class variables. These can be used to fine tune the parameters and to gain information that might not be covered by the default outputs. The methods themselves can be called individually or bundled in the right order. Furthermore, the option to use a **.ini* file for the most important parameters is given to allow for a clearer overview when using QCP.

2.2. Atomic Resolution Imaging

Finding the tiling is integral in determining the structure of a QC. The direct way to determine the tiling is to analyse the placement of atoms in microscopic images of crystal surfaces. This section gives an overview of how such images are made and the kind of microscopes required.

2.2.1. Optical Microscope insufficiency

Microscopes are designed to observe material features unavailable to the naked eye due to their size. The earliest models used simple magnifying lenses and a light source to enhance observations. Those models are known as optical microscopes, and similar versions are still used today [16]. But these microscopes are limited by diffraction. The diffraction limit is set as

$$d = \frac{\lambda}{2n\sin\theta}, \tag{2.1}$$

where d is the minimum distance between two points so that they can be recognised as individual points, λ is the wavelength of the light, n is the refractive index of the medium and θ is half of the light cone angle that can enter a lens [17]. As visible light wavelength has a magnitude of 10^{-7}m, $\sin\theta < 1$ and atom sizes are of the magnitude 10^{-10}m, n would have to be at least 10^3 for atoms to be observable with visible light. But the refractive index for a very dense material such as diamond, for example, is ≈ 2.4 in the visible range [18], which is still 2 orders of magnitude smaller than what would be necessary. Therefore, other microscopy methods are needed in order to get image resolution on an atomic level.

2.2.2. Electron Microscopes

One of those methods uses electrons instead of light for observation, which allows for a higher resolution due to the small wavelength of electrons. For example, an electron accelerated with 300 kV would have a wavelength of $\lambda \approx 0.002\,\text{nm}$ [19]. While there are multiple kinds of electron microscopy, in particular Scanning Transmission Electron Microscopy (STEM) can be used to capture atom positions in quasicrystals [20]. In STEM, a narrow electron beam of around $0.1\,\text{nm} - 1\,\text{nm}$ is focused onto a thin sample, allowing the electrons to pass through or be scattered. As this beam scans across the sample, various data can be recorded depending on where the detectors are placed. The data required to record atom positions is found scattered at high angles, where an annular dark-field detector measures the intensity of the scattered beam. This is known as High Angle Annular Dark-Field (HAADF) imaging. The recorded electrons are the result of Rutherford scattering, where an electron passes close to the atom nucleus and as a result gets scattered by the forces acting on it. Since the amount of scattering and thus the recorded intensity is proportional to the atomic number of the atom, not only does this method reveal atom positions but also gives information on the nucleus [20].

2.2.3. Scanning Probe Microscopes

Another method commonly used to observe individual atoms only uses optics indirectly. Instead, a probe attached to a cantilever is moved across the sample surface. Two types commonly used are Atomic Force Microscopy (AFM) and Scanning Tunneling Microscopy (STM). AFM uses the force acting on the probe when close to the sample, while STM measures the current produced by electrons tunneling between the probe and the sample. In both cases, topological maps are created from the movement of the probe [21].

An AFM consists of three piezoelectric transducers that can move the probe across the sample surface in all three dimensions, a sensor (usually a light lever) and a feedback control unit. For atomic resolution the non-contact mode is employed, while the feedback control keeps the probe tip at a set distance from the sample so as to not damage either. In this mode, the cantilever is oscillated at or near its resonant frequency. The amplitude depends on the distance to the sample, as the forces acting on the probe influence the oscillation. The change in amplitude is then used by the feedback control [21].

While AFM can be used for the same purpose, STM is more reliable in reaching atomic resolution. This is because in STM only the atom at the lowest point of the tip interacts with the sample, while in AFM the whole probe can influence the reading [21]. If the distance between the probe tip and the sample is only a few tenths of a nanometre, then electrons can tunnel through when a bias voltage is applied. This electron current is then amplified and used by the feedback control unit to raise or lower the probe [22].

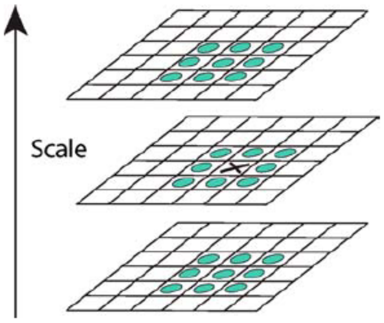


Figure 2.1.: Scale space extremum marked by a cross. It is an extremum if its value is higher or lower than the values of all the other points marked by blue dots. Source: Lowe (2004)

2.3. Atom Detection

2.3.1. Scale-Invariant Feature Transform

Since atoms in atomically resolved images appear as disks it is favourable to use a blob detector to find their locations. Since it has shown promising results when applied to images of crystal surfaces [12; 1], the scale-invariant feature transform (SIFT) algorithm was chosen to be applied.

SIFT is comprised of four steps, beginning with the detection of scale-space extrema. The scale space of an image is created by repeated convolution with a Gaussian function. Each convolution increases the standard deviation σ by a factor k. The convolutions are divided into octaves, where each octave ends with twice the σ as it started with. The factor k is thus chosen such that each octave can be divided into s intervals, therefore $k = 2^{1/s}$. Furthermore, with each doubling of σ and hence with each new octave, the image is scaled to a quarter of its size by removing every other pixel. The difference between each scale space images that differ by a factor k in an octave is then taken and scanned for extrema. An extremum in this case is a pixel that is a maximum or minimum relative to all the neighbouring pixels in the same image as well as the difference of Gaussian images from the scales above and below, as seen in figure 2.1 [14].

The second step in the algorithm is the localisation of the keypoints. By expanding the scale-space function $D(x, y, \sigma)$ (i.e. subtraction of Gaussian-convolved images) with the sample point shifted to origin into a Taylor series and setting its derivative to zero, the location of the extremum relative to the sample point can be inspected and, if needed, shifted. Furthermore, the value of $D(x, y, \sigma)$ at the true extremum location gives a measure of the local contrast and can be used to filter out faint points. Since edges tend to lead to extrema in scale-space, a principal curvature analysis is applied to filter them out as well. For this, a Hessian matrix

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{yx} & D_{yy} \end{bmatrix} \tag{2.2}$$

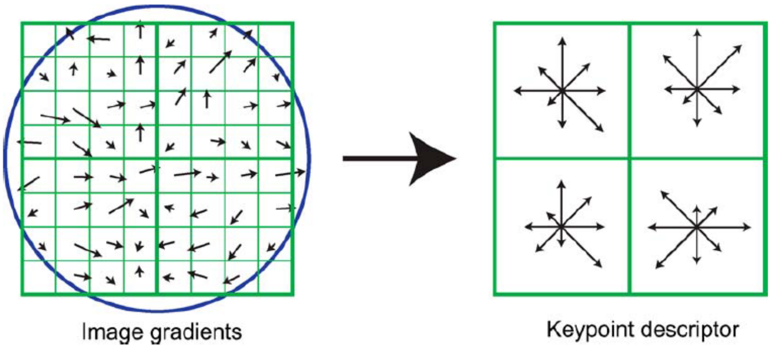


Figure 2.2.: Visualisation of a keypoint descriptor. On the left side, the local gradients in an 8×8 region around the keypoint are calculated. The blue circle is the Gaussian window for distributing weight to the gradients. In each quadrant the gradients are then summed up in an orientation histogram, seen on the right side. The result is a $2 \times 2 \times 8$ descriptor array. Source: Lowe (2004)

is calculated. The eigenvalues of this matrix describe the amount of curvature, one of them being the biggest curvature along any direction and the other one the curvature in the perpendicular direction. Comparing the trace and the determinant of H and thus comparing the magnitude relationship between eigenvalues allows for rejecting any keypoint whose eigenvalue ratio is above a certain limit [14].

After the filtering is done, each keypoint is assigned an orientation based on the gradients of nearby sample points. The contribution of each sample gradient is weighted by a circular Gaussian curve around the extremum. If there are more than one local gradients of comparable magnitude, multiple keypoints are created, each with a different orientation based on the dominant gradient [14].

The last step is the creation of the descriptor. As in the previous step, the gradients of nearby sample points at the same scale are computed and weighted by a Gaussian so that points close to the extremum have more impact. The gradient directions are rotated relative to the keypoint orientation. The descriptor region is then divided into 4×4 bins, each of which contain a histogram of all orientations they cover. Generally, 16×16 sample areas are used for descriptor calculation, which leads to 4×4 histograms with 8 orientations each, resulting in an 128-dimensional vector. An example of a smaller $2 \times 2 \times 8$ descriptor is shown in figure 2.2. The local image gradients are weighted by Gaussian window and transformed into histograms of gradients. Finally, the histograms are normalised and a threshold is applied to minimise effects of constant contrast changes [14].

2.3.2. Keypoint Detection

Upon initialising a class instance and providing an image, the SIFT algorithm is applied using `opencv`'s implementation[23]. The result is an array of scale space keypoints. Since the algorithm returns any extrema it deems acceptable, regardless of size or multiplicity, some unusable keypoints need to be removed before proceeding.

The first filtering step removes keypoints that are too small or too large. These wrongly sized keypoints originate from the lowest and highest octaves. Under assumption that most of the keypoints are atoms and thus have size values in a relatively small range, one can use the median value to represent the atom size and set a limit to the size disparity. This is done with an size threshold parameter, which sets this limit by multiplying the median size with it for the upper bound and dividing the median size by it for the lower bound. For example if the size threshold parameter would be 2, then the upper bound would be twice the median size and the lower bound would be half the median size. Following this, keypoints close to the image edge are filtered out. These edge points are very likely to have wrong locations (e.g. if the true keypoint center is outside the image) and thus could falsify the structure information. The minimum distance to the edge that a point can have is determined using an edge threshold parameter multiplied with the median size. The use of the median keypoint size to set the limits in these filtering steps ensures that they work for any image resolution. Lastly, any remaining multiples of keypoints are also removed by taking the first occurrence of a keypoint at a location and deleting all other ones.

After any incorrectly sized or positioned points are filtered out, the new median size is calculated and every keypoint size is set to be a fraction of the median size. The keypoint size determines the area in which the descriptor is calculated. The descriptors are what will be later used to classify the keypoints. A smaller keypoint means that the descriptor will focus on the central gradient, while a larger keypoint also takes neighbouring gradients into account. Since QC do not possess translational symmetry, the goal is to be able to distinguish between atoms and the space between them, which have opposite central gradients. The angle of the keypoint is also set to 0 to catch any periodicity that may arise when investigating a new material, as well as removing any random directional influences on the descriptor.

Finally, the descriptors are computed for the filtered and resized keypoints.

2.3.3. Keypoint Classification

The differentiation between atom keypoints and non-atom keypoints is done via clustering. Descriptors are vectors in a 128-dimensional space[14] and similar descriptors should form distinct clusters in this space.

To reduce computation time and remove any random elements, agglomerative (bottom-up) clustering was chosen. This clustering approach begins by setting each descriptor in its own cluster and merging two clusters together with each step. Which clusters are merged at each step is decided by the linkage criterion, which was set to be the Ward linkage. This linkage merges clusters such that the variance within clusters remains minimal by minimising the inter-cluster error sum of squares

$$\mathbf{ESS} = \sum_{i=1}^{n} x_i^2 - \frac{1}{n}\left(\sum_{i=1}^{n} x_i\right)^2 , \tag{2.3}$$

where x_i is the value associated with each data point, which in our case is the distance to the current cluster center [24]. The total ESS of the dataset is given by the sum of the ESS of each cluster. Following this criterion, a hierarchical tree of all cluster mergers is created.

The problem of where to cut the tree (i.e. how many clusters should the data be split up into) is solved by using the silhouette score. The silhouette score of a data point can be defined as

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}, \tag{2.4}$$

where $a(i)$ is the average dissimilarity of data point i within the cluster it was assigned to, that is the mean distance to all the other data points in the same cluster, and $b(i)$ is the minimum average dissimilarity to data points in other clusters [25]. By taking the average silhouette score $\overline{s}(k)$ for a number of clusters k one can determine the amount of clusters best suited to classify the data [26]. Given a range of values for k, the value with the highest $\overline{s}(k)$ is chosen.

Since the whole hierarchical clustering tree was calculated, it can be now cut at the desired amount of branches (clusters) and thus labels assigned to each data point. This specific clustering process is packed in a function and used at other points during the process as well and will simply be referred to as "clustering". In this case the amount of clusters is already known as 2, since we only want to differentiate between "atoms" and "no atoms", and the cluster amount range adjusted accordingly. By feeding the descriptors into this function, the associated keypoints are classified into their respective clusters. But it is not yet known which cluster is the one containing the atoms.

The approach to discerning the atoms is to investigate if there is any structure to the keypoints or if they are seemingly randomly distributed. To this end, for each point in a cluster the k nearest neighbors are found and their relative position as well as distance to the respective point determined.

To decrease computation time, scikit-learn's implementation of the kd-tree algorithm was used[27]. Instead of comparing distances between all data points, a binary search tree is used to find the nearest neighbours [28]. The root of the tree is the entire data set. At each node, the space is partitioned depending on the chosen key. For 2-dimensional data, the keys are the x and y values, splitting the data into subsets such that the two resulting tree branches contain all data points with values greater or smaller than the key value respectively. Performing this space division at each node while cycling the keys results in small subspaces called "buckets". During nearest neighbour search for a given data point, at each node the branch containing the subspace with the point is chosen until it arrives at a terminal node. The distance to the data points in the bucket are then calculated. During the search, a list of the closest k points is made and updated with each node traversal. Furthermore, upon arriving at a terminal node, it is checked if any partition line is closer than the current nearest point in the bucket. If such a line is found, branches are backtracked until arrival at the node associated with it and the other branch is taken to see if there are any points closer in that area. This proximity check is then performed along every branch until the root is reached again [28].

The cluster containing the atoms should exhibit a certain structure in the point locations, leading to a structure in the nearest neighbour distances as opposed to the increased randomness in the cluster without information. Therefore, clustering is applied to the nearest neighbour distances within the keypoint clusters. The cluster amounts are set to be between 2 and the amount of neighbours investigated k, since it is expected that atoms will have neighbours at the same distances without skipping one and on average more than 1 neighbour at a certain distance to be able to create tiles. The silhouette scores of the best fitting distance clustering for both keypoint clusters are compared and the higher one chosen to be the one representing the atoms.

Now that the classes are known, a support vector machine (SVM) is trained on the result to be able to assign any new keypoint to "atoms" or "no atoms" depending on its descriptor. SVMs are a supervised learning method where a hyperplane separating the data is sought. The separating hyperplane is of the form

$$y(x) = w^T \phi(x) + b \tag{2.5}$$

where $\phi(x)$ a feature space transformation of the data points, b a bias parameter and w the parameter to be optimised. Using Lagrangian multipliers a_n based on constraints enacted from the canonical representation of the hyperplane

$$t_n(w^T \phi(x_n) + b) \geq 1 \qquad n = 1, ..., N, \tag{2.6}$$

w can be expressed as a sum of the feature vectors $\phi(x)$ multiplied with a_n and t_n, which are corresponding target values $\in \{-1, 1\}$, as

$$w = \sum_{n=1}^{N} a_n t_n \phi(x_n). \tag{2.7}$$

With this, equation 2.5 contains an inner product, which allows the use of kernels. Kernels are symmetric functions of the feature vectors that can be used to replace the inner product, which is especially useful in case the data is not linearly separable [29].

The SVM was implemented using scikit-learn [27], for which the default kernel is a radial basis function, which depends only on the distance between two points. The trained SVM will be used to find any points missed by SIFT at a later point.

2.4. Tile Extraction

2.4.1. Symmetry Vectors

In order to find the tiles, the connections between atoms at the minimum shared distance need to be found. These connections are the vectors that make up the symmetry and mark the edges of the tiles. The k nearest neighbours for each atom have already been recorded, but it is not yet clear which distance cluster contains the symmetry vectors. It is not possible to take the cluster with the smallest average distance, as rhombic tiles might lead to distances smaller than the length of the symmetry vectors. Therefore, one

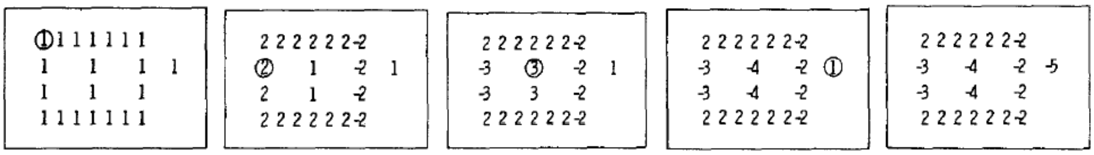


Figure 2.3.: Steps in the border following algorithm. From left to right, in each image the circle indicates the start of a new border that is completely followed before finding the next starting point. The result is an outer contour containing two holes and one external contour: in total 4 contours are found. Source: Suzuki et al. (1985).

needs to set k sufficiently small when searching for neighbours and then pick the distance cluster with the most members to get the correct vectors.

The nearest neighbour locations are shifted to be relative to origin and then clustering is performed on the relative positions of points in the cluster with the most members. Since polygonal quasicrystals only appear in 8-, 10- or 12-fold rotational symmetry, the cluster amount search can be limited to those numbers. Taking the median location of each cluster relative to origin gives the symmetry vectors. The median is preferred over the mean to remove outlier influence. Strictly speaking there are only 4 linearly independent vectors indicating the symmetry for each QC type, as they are projections from 4D space, but for simplicity all 8, 10 or 12 of them will be referred to as "symmetry vectors".

2.4.2. Contouring

Using the connection information obtained during the symmetry vector search, a binary image is created where lines are drawn at 1 pixel width between atoms. These lines are the tile edges. A contour detection algorithm is then applied to find all inner and outer borders, where the former contain the tiles. The algorithm was developed by Suzuki et al. (1985) and implemented using `opencv`[23]. It works by scanning a picture row by row top-down left to right and assigning values to the pixels depending on their surroundings. If a pixel at position (i, j) is 1 and preceded by a pixel of value 0 at position $(i, j-1)$, it is deemed to be the start of an outer border. Similarly, if a pixel of value ≥ 1 is succeeded by a pixel of value 0, it is set as the start of an inner border. Every time the start of a border is detected, a border following algorithm is applied, setting the values of the pixel to integers NBD[1]> 1. This works by scanning pixel surroundings clockwise to find where the border continues, taking a step along and then scanning counterclockwise to determine if the pixel is part of the right part of the border and setting it to $-$NBD, such as to not be counted as inner border beginnings. Upon reaching the starting pixel again, the border following stops and the row scanning continues. With each new border starting point, NBD is incremented by 1. The steps in this algorithm are shown using a simple example in figure 2.3. Through the row scan it can be also determined which borders are contained within others leading to a contour hierarchy [30].

[1] NBD is the denotation of the sequential border number used by Suzuki et al. (1985).

It is not yet clear which contours are valid tiles, so incorrect ones need to be filtered out. The first step in doing so is to sort out contours with corner amounts that do not conform to acceptable tile shapes. The contour corners are counted by matching atom keypoint locations to the contours. Since SIFT can return float values for the locations and the contour points are integer values, the best way to match keypoints is to set a small tolerance area around the contour and check which points fall into this area. Generally, surface quasicrystal tiles seem to have triangular, square, rhombic and sometimes hexagonal shapes[2]. Hence, any contour whose corner amounts are not 3,4 or 6 are saved as lists of matched keypoint indices in an array holding non-tile contours. The tile contours are saved in a separate array. This filtering should remove any invalid tiles if no detection errors arise from SIFT. Erroneous tiles can be filtered out some more during tile classification.

2.4.3. Tile Classification

The amount of corners per contour is not enough to be able to classify the tiles in the provisional list yet (e.g. telling squares and rhombi apart), therefore a dataset that can be used to classify the tiles must be created. To be able to tell different shapes apart, the internal angles need to be known, consequently the informative dataset will represent each contour as a vector of angles.

The matched keypoints for each contour are not ordered, so the first step is to find which triplets of keypoints enclose each internal angle. From the distance clustering the minimum and maximum distance between two corner points is known. Calculating the distance to other corners in the contour for each corner point, the distance limit information can be used to find which pair should be assigned to the respective point to form the triplet. If more than two points are found within the limits the contour is moved to the non-tile array as it is likely to be a detection error. After the triplet pairing is complete, the enclosed angle for each triplet is calculated and saved together as a feature vector. To enable classification, all vectors have dimensions equal to the highest number of corner points detected (6 by default). Angle vectors for tiles with less corners than the dimension are padded with zeros. Lastly, the entries in every vector are sorted so that they are in ascending order.

The angle data is then clustered to differentiate between the different shapes contained. Any remaining outliers can then be filtered out by choosing a maximum amount of clusters allowed to proceed. If the number of unique labels from the clustering is less than or equal to this maximum then all the tiles are accepted. If the number of unique labels exceeds the maximally allowed value, then the contours in the surplus clusters with the least amount of members are moved to the "non-tiles" array. Quasicrystal observations generally do not have more than 3 different recurring shapes making up the pattern (see Results and Discussion), so the default maximum is set to 3.

In order to properly display the tiles as individual building blocks of the pattern, the median angle values of each cluster is taken and saved as a representative of the shape

[2] Based on existing images, see Results and Discussion

cluster. Using these median angles, the relative spatial corner positions are calculated for display.

2.5. Bottleneck Circumvention

2.5.1. Detection Errors

The extent of the pattern detection is mainly restricted by the ability of SIFT to detect distinct points, which depends partly on the resolution of the image. Microscopic images tend to have relatively low resolution, so atoms that are close together might not get recognised individually. Another factor is noise, which could also lead to errors in the keypoint detection. So instead of tuning the SIFT parameters such that many atoms are detected along with some errors, it is more accurate to keep the detection strict with only a fraction of the atoms being detected, but without any errors. It is easier to find the missing atoms than to identify and remove errors.

2.5.2. Missing Atom Search

For a search to be successful, it is important to know where to look. Missing atoms can only be found inside contours in the "non-tiles" array. Therefore, a binary image is created where tiles are given a value of 1 and non-tile contours are represented by zeros. Adding the symmetry vectors to each keypoint location in the open contours and checking that they do not land on or close to a 1 leads to candidate locations for missing atoms.

The tiling is connected (except for vacancies), so at least 2 vectors coming from the contour points should be pointing to the same area. Using an acceptance threshold parameter scaled by the median keypoint size, any candidate point that is not in the acceptance area of another point, or two other points in the dodecagonal case (where at least 3 vectors point to the same area) is removed as an outlier, since there is little possibility to find an atom in the region surrounding it. The remaining groups are clustered together, with the clustering scaling with the number of candidate points in remaining in the contour. The mean of each cluster represents the most accurate guess for a potential atom location.

But not all of them will contain atoms, since the rotational symmetry vectors are just the "rule" for the tiling. Not every vector from every atom points to another atom, as opposed to translational symmetry. This is where the previously trained SVM is used. At each potential atom location a keypoint is created manually with the same size and direction as the naturally detected ones. The descriptors of these manual keypoints are calculated and then the support vector machine decides which keypoint belongs to the atom cluster. Depending on the accuracy of initial clustering, the SVM accuracy might not be able to draw a clear separation, so a threshold parameter (henceforth referred to as "decision boundary" in accordance with the parameter name) is introduced to only accept positive SVM decisions if they are a minimum distance from the separating hyperplane.

2.5.3. Filling

Subsequently to the decision, a check is performed to see if there are any accepted candidate keypoint pairs overlapping. This can happen if two open contours share an edge, in which case the point with the lower SVM decision score is discarded. Then, the accepted candidate atom keypoints and their descriptors are added to the general class instance array for keypoints and descriptors respectively. The missing atom search only searches for immediately connected atoms, therefore large open areas need multiple searches to fill in all the atoms. Since the tiling and connections changes after each search and fill, most of the pattern calculation process needs to be run again, starting from the nearest neighbour search. With each run, the open areas get smaller until they are either completely filled or no further accepted candidates are found.

3. Results and Discussion

3.1. Example Application

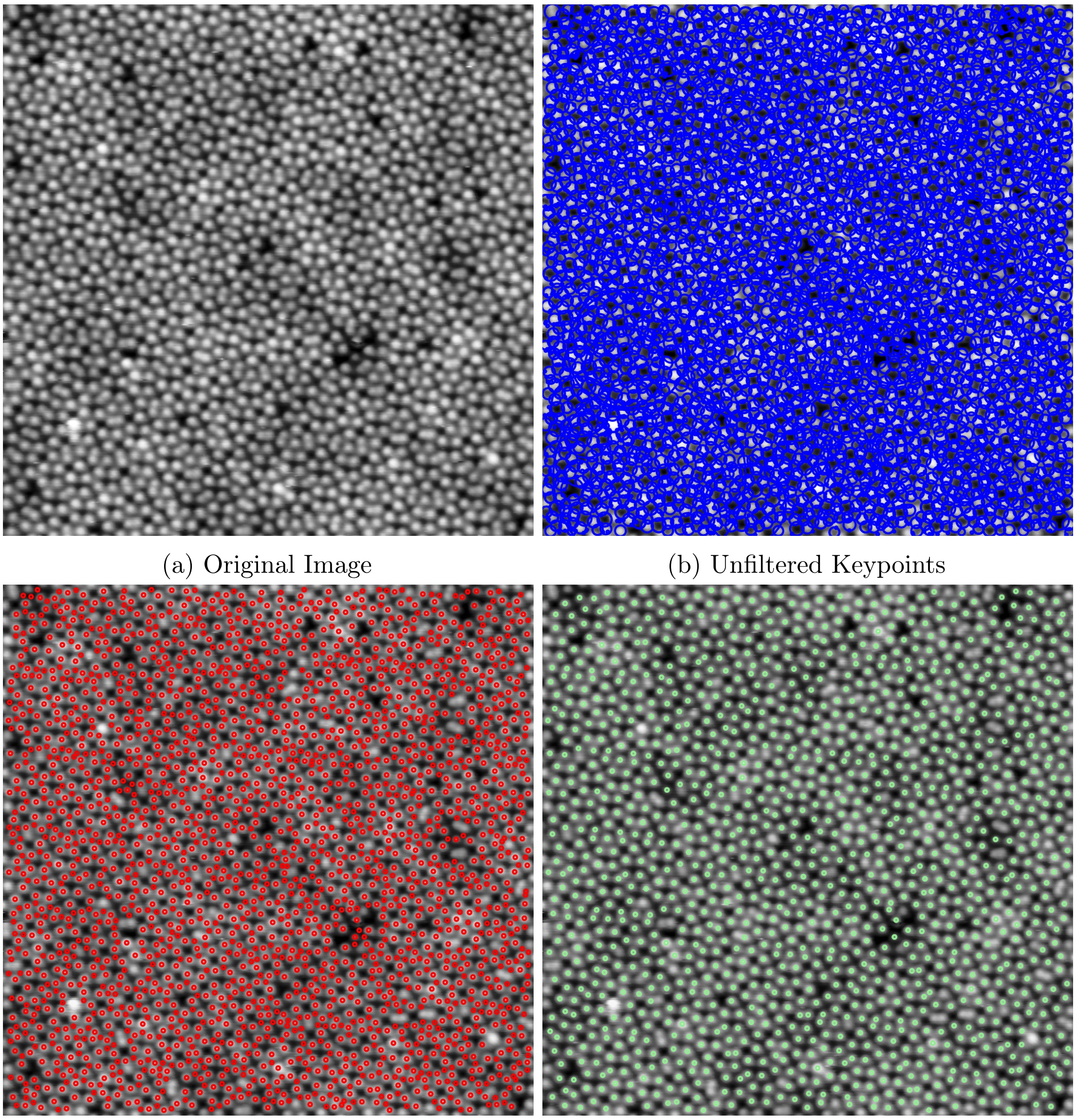
During the development of QCP, two images of different QC were used to check the functionality. Those were an STM image of Barium Titanate $BaTiO_3$ grown on Pt(111) to form a dodecagonal QC [31] and a HAADF image of a decagonal QC in the $Zn_{58}Mg_{40}Y_2$ alloy [32]. Having both images taken using different techniques and each depicting a different kind of QC helped in removing bias during development. In this section, the result of each major step during pattern recognition is shown using the former image.

3.1.1. Keypoint Detection and Classification

The original image that aided in development can be seen in figure 3.1a. The white dots are the atoms to be located.

While the default values as used in Lowe (2004) usually work, it helps to tweak the parameters of SIFT, since the application is very different to the intended use. The contrast threshold works well, but the edge threshold leads to some misses. As can be seen in the $BaTiO_3$ image in figure 3.1a, some dots can be very close together. Nevertheless, the edge threshold should not be made more lenient, as that might lead to erroneous keypoints between the adjoining dots. It is better to fill the missed points in later. The blurring factor is left as is as well, but the number of layers per octave is increased to 12 to increase the general number of detections at all scales. The result of applying SIFT with these parameters can be seen in figure 3.1b. The keypoints are drawn as blue circles characterized by their location and size (which adheres to the scale at which the keypoint was found).

For filtering, any keypoint whose size was bigger than 1.2 or smaller than 1/1.2 times



(a) Original Image

(b) Unfiltered Keypoints

(c) Filtered Keypoints

(d) Atom Keypoints

Figure 3.1.: **(a)** Grayscale version of the original $BaTiO_3$ image, as provided by Förster et al.. **(b)** Initial results of SIFT using the chosen parameters. Keypoints are displayed as blue circles with diameter equal to their respective size around the keypoint center. Due to many keypoint detections across all scales it looks chaotic. **(c)** Remaining Keypoints after filtering and resizing. **(d)** Only the Keypoints belonging to the cluster showing the most structure (i.e. the atoms).

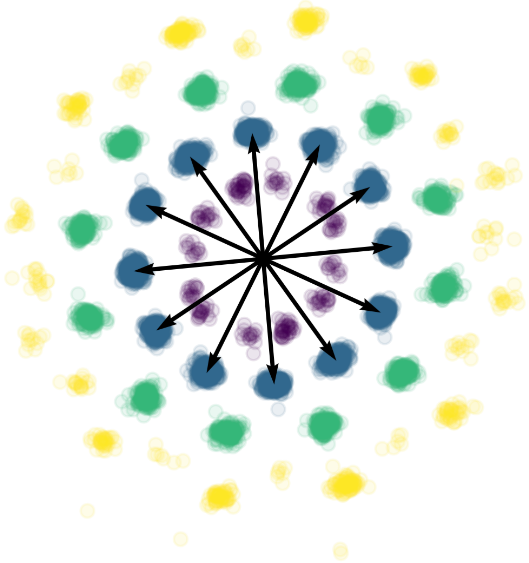
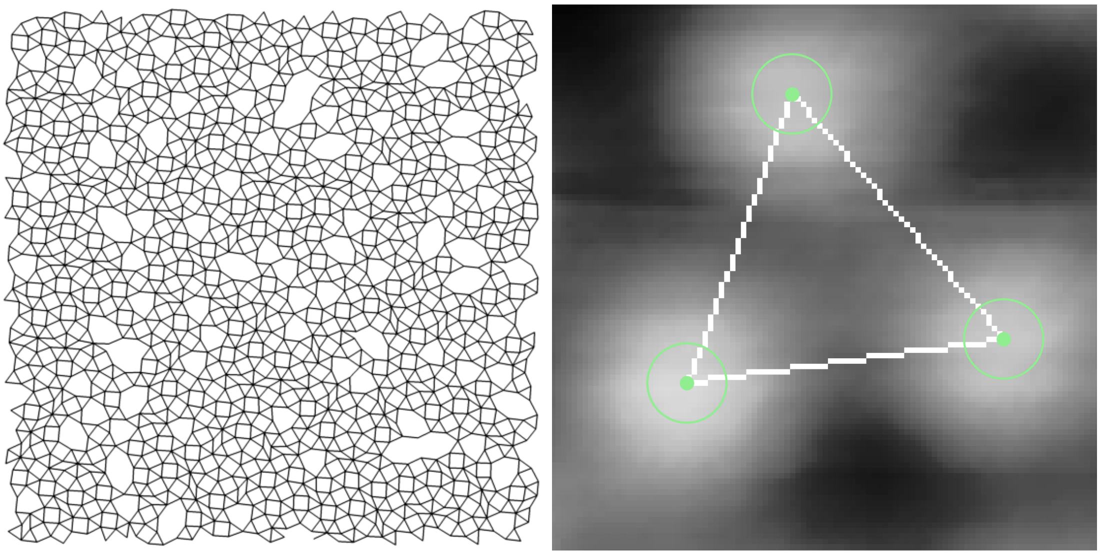


Figure 3.2.: Nearest neighbour distribution for the atoms in the $BaTiO_3$ QC as shown in figure 3.1a. The colours indicate the distance cluster, the arrows indicate the symmetry vectors.

the median keypoint size was removed. At the edges, any keypoint within 1 median size distance was removed as well. The default resizing value that works well is 0.33, meaning that the descriptor frame is shrunk by 66%, capturing only the central gradient. The remaining keypoints are shown in red in figure 3.1c. Upon closer inspection it can be seen that some red circles contain bright centers and some contain dark centers. It also becomes apparent that SIFT struggles with detecting points that are close together, as mentioned before that is a bottleneck to be be fixed later.

After the first clustering is applied and the nearest neighbours investigated, the keypoints deemed to be atoms through the nearest neighbour clustering structure analysis are circled in green in figure 3.1d. The placement of the non-atom keypoints also shows some form of structure, but it is always just a bit more chaotic, making the distinction possible. The nearest neighbour distribution for $k = 7$ neighbours for the atom keypoints is shown in figure 3.2. The colors denote how the nearest neighbours are clustered by distance, the blue cluster being the one with the most members. The arrows are the symmetry vectors, each pointing to the median value of the location clustering in the chosen distance cluster. It is clear from this graph that the image contains a dodecagonal QC, based on the number of vectors. There is also a similarity to the way diffraction patterns of QC appear.

While finding the optimal parameters required some manual adjustments to get from (a) to (d) in figure 3.1 with the best possible results, the default parameters of QCP ensure



(a) Keypoint Connections

(b) Single Contour with matched Keypoints

Figure 3.3.: **(a)** Connected atom keypoints showing tile edges of the $BaTiO_3$ QC as shown in figure 3.1a. **(b)** Example of a tile contour with the associated connected keypoints. The green circles show the scaled down size of the keypoint, the green dots are their centers.

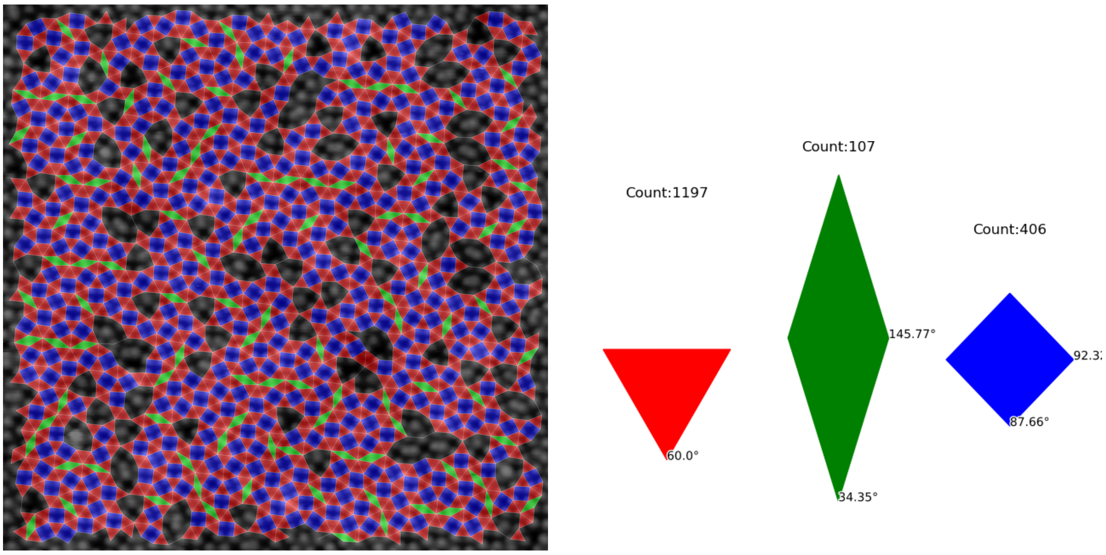
that for most images[1] the atom keypoints are detected without the need for interference. The default SIFT parameters are left as suggested by Lowe, with the exception of the number of layers, for which 12 was set as the new default. For the filtering, the size ratio defaults to twice/half the median size and the edge threshold defaults to 1. Furthermore, keeping the amount of nearest neighbours between 5 and 10 is a good choice, since all the atoms in the immediate surroundings should be covered in that range. Therefore, $k = 7$ was chosen as the default value for the nearest neighbour search.

3.1.2. Contouring and Tile Extraction

If, for every atom, a direct line is drawn to every one of its neighbours located in the blue cluster in 3.2, the result would be an image of the tile edges. While only 1-pixel width lines are drawn for optimal contouring, an example with slightly thicker lines for better visualisation is shown in figure 3.3a. Next to it, in figure 3.3b, an example of a single contour with its matched keypoints can be seen. While the corner points seem to coincide perfectly with the keypoint centers, this is not always the case. An offset of a few pixels can happen, therefore a nearest neighbour search needs to be conducted here as well using kd-tree.

Obtaining the corner points means that the angles spanning the shapes can be calculated,

[1] with the exception of images with high error influence. Such images need to be pre-processed before being analysed.



(a) Tiling

(b) Unique Tiles

Figure 3.4.: **(a)** The (temporary) final tiling of the $BaTiO_3$ QC, shown in color on top of the original image. The uncolored regions are caused by detection misses from SIFT or by natural vacancies.**(b)** The content of the tiling. The median corner angles and tile counts are displayed as well.

which are then clustered and the median value of each cluster represents each tile, as shown in figure 3.4b. Using the results of this clustering, the tiles can be colorcoded for a better tiling visualisation as seen in figure 3.4a, overlaid with a 50% transparency on the original image. While the general tiling makeup and structure can now be deduced, it is obvious that the points missed by SIFT leave unwanted gaps in the tiling.

3.1.3. Bottleneck Circumvention

Using the symmetry vectors to find potential missed points leads to the candidates seen in figure 3.5. Green dots signify that the candidate has been accepted as an atom keypoint, red dots means that it was rejected. The SVM application leads to satisfying results, not needing to apply a decision boundary. The final tiling for this test image is shown in figure 3.6a. The colour-coding of the tiles is different, since the colours of the tiles depend on the assigned label, which can differ if new entries are added to the data. With the exception of some open contours due to vacancies, the tiling is complete after the missing point search. The individual tiles can be seen in figure 3.6b. With this, the pattern recognition is successfully finished. Rejected candidates from vacancies can be used further to infer possibilities for a full tiling coverage.

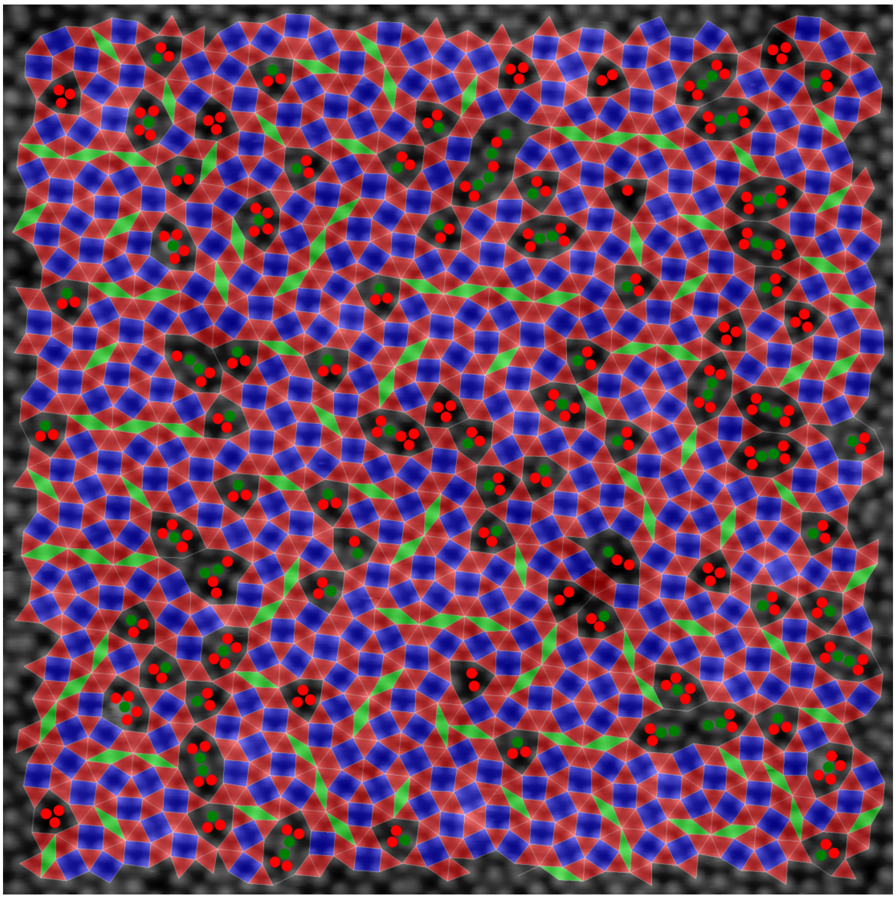
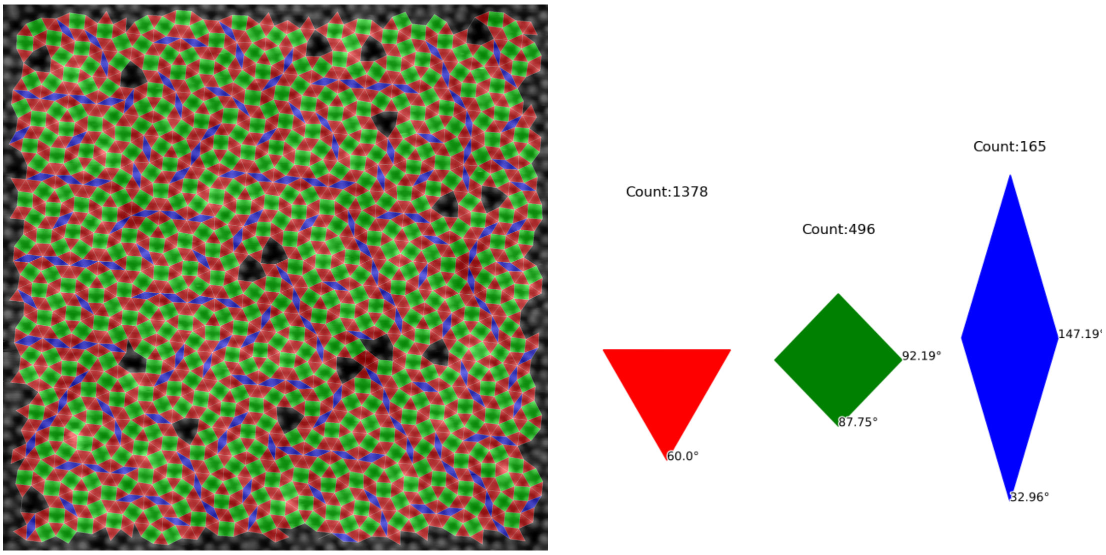


Figure 3.5.: Accepted and rejected atom keypoint candidates shown in the "open" contours in the $BaTiO_3$ QC. Green are accepted candidates, red are rejected ones.



(a) Filled Tiling

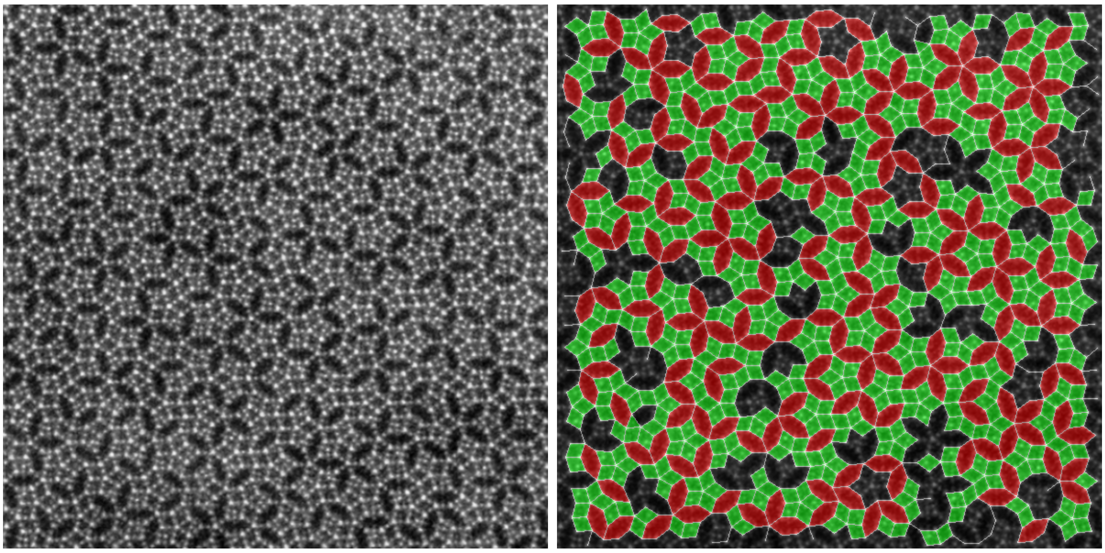
(b) Unique Tiles

Figure 3.6.: **(a)** The final tiling of the $BaTiO_3$ QC with the exceptions of vacancies (dark regions). **(b)** Calculated tiles along with their detected number and median internal angles.

3.2. Further Results

3.2.1. Decagonal QC

As mentioned before, a HAADF image of a decagonal QC in $Zn_{58}Mg_{40}Y_2$ was used for development as well to ensure that the pattern recognition can be used for broad applications. The grey-scale version of the original image can be seen in figure 3.7a. The default SIFT setting does not produce good results for this image due to atoms being relatively small with relative low contrast changes, but it is only a matter of decreasing σ to achieve good results. The best results were found with $\sigma = 3.4$ and a total of 24 layers per octave. Furthermore, since the atom sizes are small relative to the image size, the resizing parameter was increased to 0.66. All other parameters were left on default. The initial tiling result is shown in figure 3.7b. The shapes were automatically limited to 2 clusters, rhombi and hexagons, which already fill up most of the space. In the lower left and upper right portion of the image there are two rhombi who have not been filled in correctly, but instead are shown to be cut into a triangle, which is added to the rhombic cluster, and a pentagon that is discarded as it does not fit the tiling rules in QC. This error occurred due to mis-classification from the clustering algorithm, probably arising due to a location inaccuracy from SIFT or the natural deviation. The nearest neighbour distribution with color-coded clusters is shown in figure 3.8. It can be seen that the nearest and next-nearest neighbour distances are very close together, so a point in either cluster deviating too much might get misclassified. Nevertheless, the initial results are



(a) Original Image

(b) Tiling

Figure 3.7.: **(a)** Original image of the decagonal QC **(b)** Initial tiling, using SIFT with decreased σ and increased number of layers.

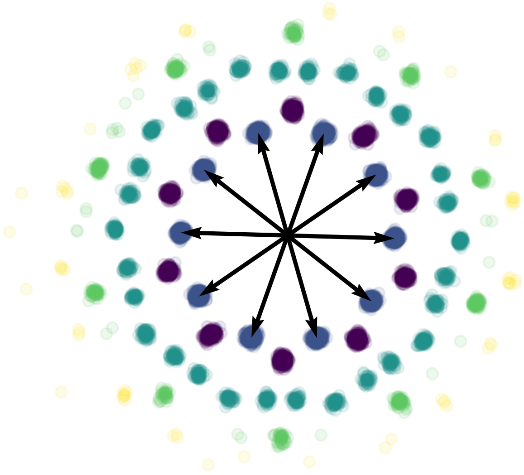
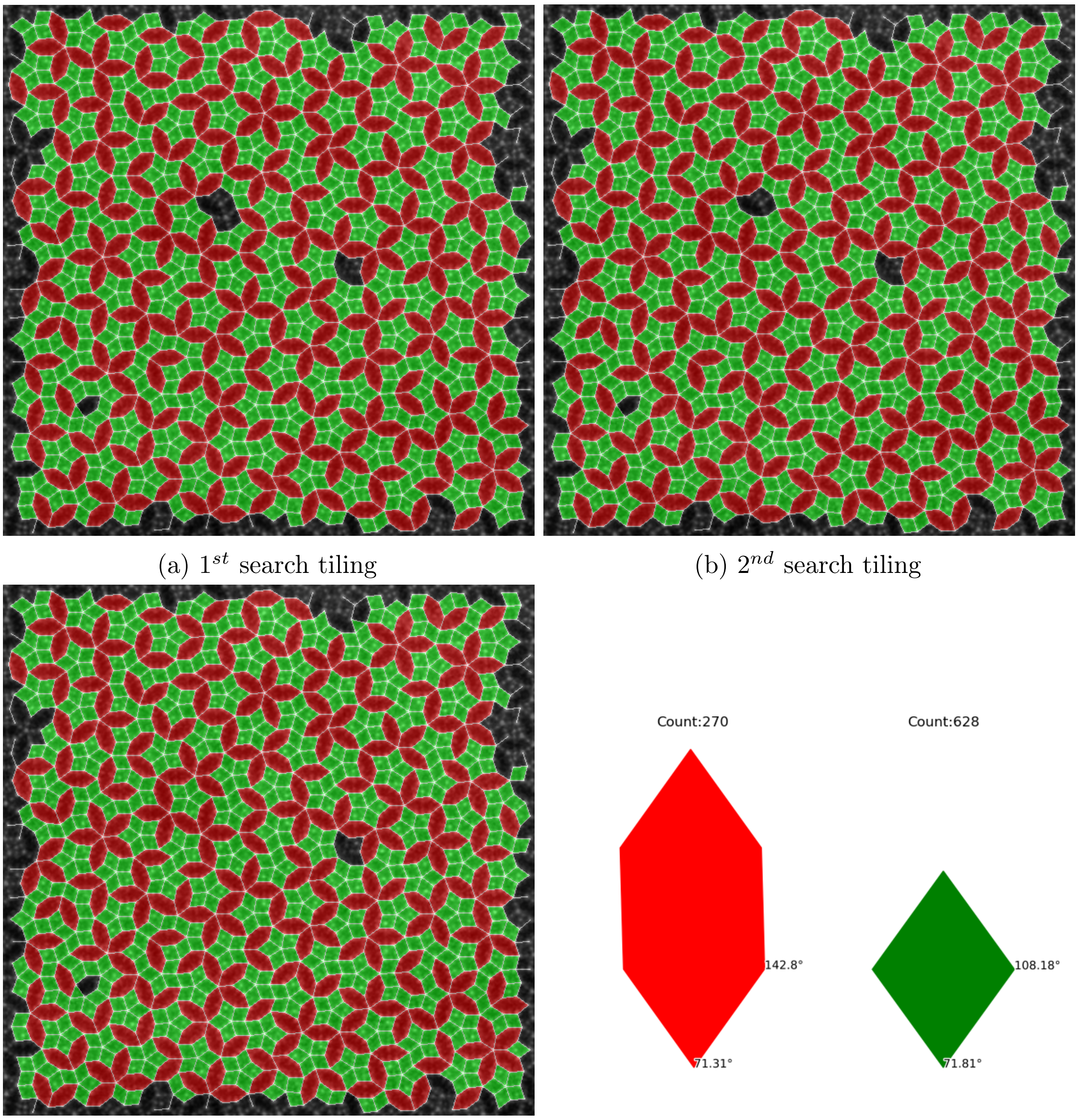


Figure 3.8.: Nearest neighbour distribution for the decagonal QC.



(a) 1^{st} search tiling

(b) 2^{nd} search tiling

(c) Final Tiling

(d) Individual Tiles

Figure 3.9.: Results for a decagonal QC. **(a)** Tiling after the first missing point search. **(b)** Tiling after the second search. **(c)** Final tiling with no more further accepted candidates. **(d)** Tile makeup in the final tiling.

already a good start for image analysis.

The decision boundary was left at 0 with no false positives and the search algorithm ran until no further accepted candidates were found. The tiling results from those runs can be seen in figures 3.9 **(a)**-**(c)**. The first search took care of most of it, but for one open contour it took a bit longer. With the exception of the 2 cut hexagons, there is one single open contour remaining, which is just a natural inconsistency in the tiling. It is referred to as "the boat tile" by Wang et al..

Changing the SIFT parameters and the search decision boundary has shown good results for the image of the decagonal QC. As mentioned before, this image along with the image of the $BaTiO_3$ QC were used to write and test the program. The following images are used to test the capabilities of the software and prove its flexibility.

3.2.2. Approximants

QC approximants are crystals, whose unit cells contain a quasiperiodic structure. If QC are a projection at an irrational angle, then approximants are thought to be a projection at an angle that approximates this irrational number [8]. They have thus a rotational symmetry like QC, but offer periodicity like regular crystals [33]. Furthermore, the atom arrangement can be expressed using the tiles that make up QCs [9], which leads to them being a good test subject for pattern extraction. With this it can be shown that global and local quasiperiodic patterns can be detected.

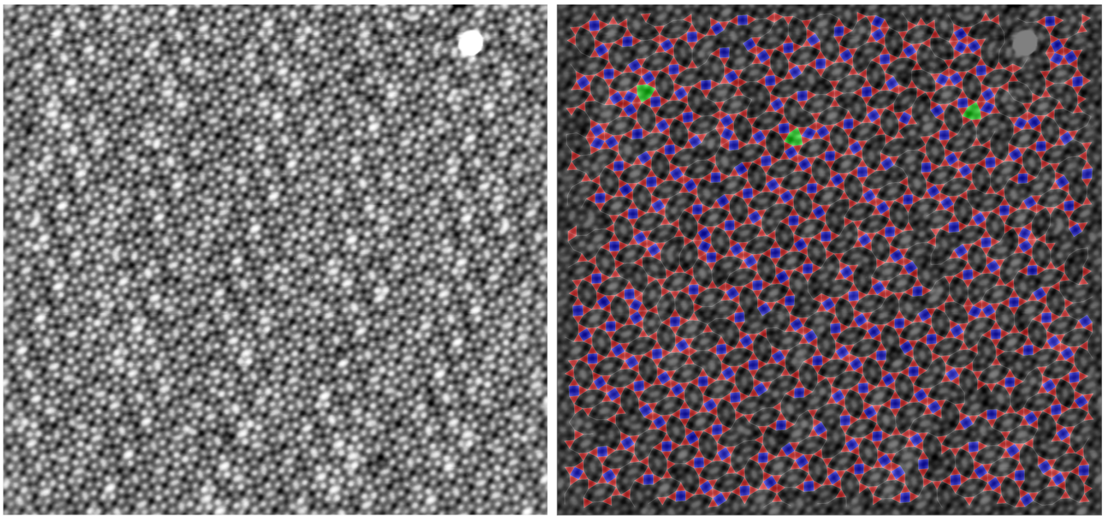
5R approximant

When grown on Pd(111), $BaTiO_3$ forms an approximant consisting of the same tiles as it's quasicrystalline form on Pt(111). While the tiling is locally quasiperiodic, it forms unit cells consisting of 40 Ba atoms and has been denoted as a **5R** approximant based on the number of rhombi per unit cell. Within this approximant structure, there also exist some other unit cells with different tiling ratios, namely **3R** and **8R** [34].

Since the images are very similar, it is enough to show that the algorithm works correctly on the **5R** images, which were provided by Wührl et al.. The original approximant image is shown in figure 3.10a in grey-scale. Using the default SIFT parameters and a keypoint size limit of 1.1, the resulting tiling can be seen in figure 3.10b. While it may not be a satisfying result yet, it is a good precedent for the missing point search. Fewer correct points are better than many points prone to errors. It is apparent that points close together are not detected, therefore there are no rhombi present. Instead, three contours with 6 corners were mistakenly identified as a hexagonal tile.

The nearest neighbour distribution in figure 3.11 confirms the 12-fold rotational symmetry of the approximant. Here again it can be seen that no points in close proximity were detected, as there are no keypoints closer together than the most prevalent inter-atom distance.

The final tiling for the image was able to be found after only 2 search runs. The first one, seen in figure 3.12a, finds most of the rhombi and eliminates the hexagon tiles. The second run fills the hexagon remainders, as well as anything else that was left from larger



(a) Original Image

(b) Tiling

Figure 3.10.: **(a)** Original image of the 5R approximant **(b)** Initial tiling, using SIFT with default parameters.

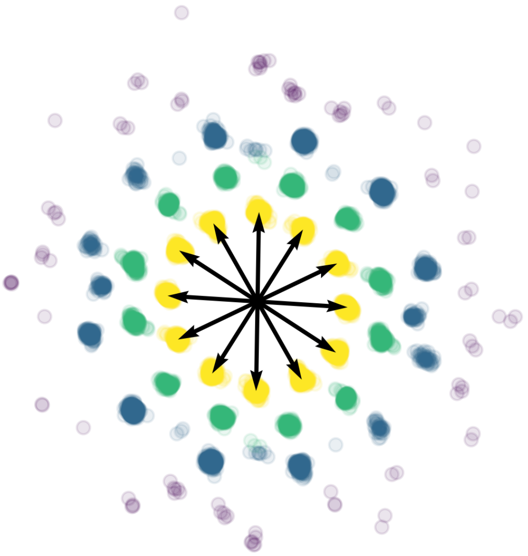


Figure 3.11.: Nearest neighbour distribution for the 5R approximant

contours, with the exception of one hexagonal open contour in the middle left area. The missing keypoint there is not classified correctly due to low contrast changes stemming from the other atoms in close proximity. Moreover, two errors appear due to 4 atoms being in close proximity. As can be seen in figure 3.12d, the pattern detection sees two squares, two triangles and three rhombi in an overlap. This is a natural inconsistency in the atom arrangement and thus not an algorithmic error, so there is no point in attempting to correct this.

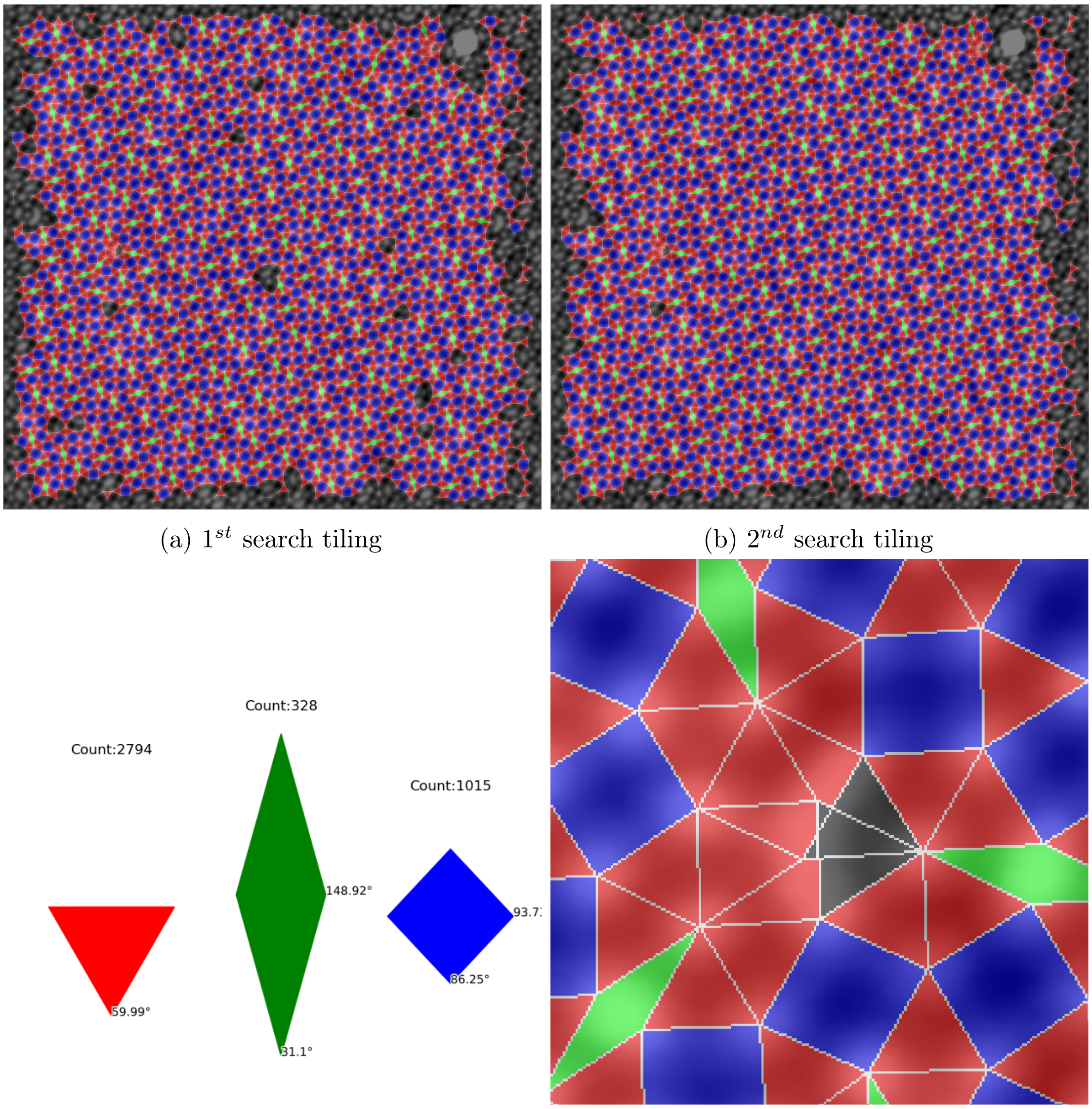
3.2.3. Liquid Quasicrystals

Quasiperiodic structures can also be found in matter other than metal alloys. A "Liquid Quasicrystal" (LQC) is a soft material made up of so-called micelles, spheres created from branching molecules (dendrons). These micelles are arranged in tilings similar to those of QC [35].

An image of such a LQC, made using phase imaging, where the contrast is created by phase differences between incident and diffracted beams, is shown in figure 3.13a. The image was provided by Zhang et al.. The dark spots in the image are not the micelles themselves, but rather the centers of the atomic structure surrounding them. Therefore, the locations of the micelles are at the dark spots, so they can be used to detect the structure. Since the image has many contrast fluctuations throughout, such as the large dark gaps or the small white lines between the dark spots, the edge threshold of SIFT was increased to twice its value, thus decreasing the amount of edge detections. Furthermore, to combat detections from noise, the blurring parameter σ was tripled, so that small features are barely detected. Lastly, the contrast threshold was increased to 0.1 to ensure that only the features with the highest contrast changes are accepted, which helps in ruling out low contrast features found in the dark gaps. The keypoint filtering size threshold was also lowered to 1.5, so as to be sure that only similarly sized features are considered.

The resulting initial tiling can be seen in figure 3.13b. The dark gaps were mostly avoided with the newly parametrised SIFT and the local quasiperiodic structure can already be seen in the brighter areas. Due to the prior known information that the tiling consists only of squares and triangles, the maximum amount of tile clusters was lowered to 2. If left at the default value 3, some open contours containing 6 equilateral triangles would be considered as hexagon tiles. Nevertheless, if the algorithm would be applied without the prior tiling information, it would still be successful in finding the real tiling (as part of the assumed tiling) and the maximum cluster amount can be altered after the missing point search or another completely new run can be performed.

Since the image itself is full of errors, the keypoint detection is inaccurate, leading to a chaotic nearest neighbour distribution, as seen in figure 3.14. There is no clear border between the distance clusters and the vector directions are inaccurate, as the neighbour clusters are not distinctly separate. The amount of nearest neighbours had to be lowered to 5, since anything higher would lead to higher distance clustering inaccuracy due to the feature location errors. Nevertheless, after searching for missing points, most of the detectable tiling can be seen in figure 3.15. Although the results are far from perfect, the performance was better than expected for an image with so many local contrast



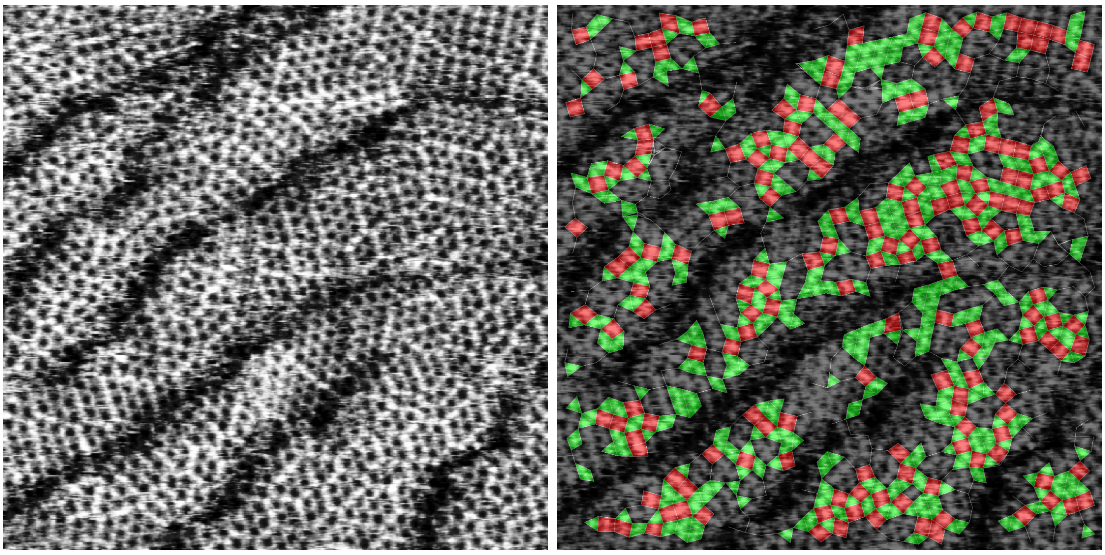
(a) 1^{st} search tiling

(b) 2^{nd} search tiling

(c) Individual Tiles

(d) Tiling Error

Figure 3.12.: Results for the **5R** approximant. **(a)** Tiling after the first missing point search. **(b)** Tiling after the second search. **(c)** Tile makeup in the final tiling. **(d)** Local error found in the middle left portion.



(a) Original Image

(b) Tiling

Figure 3.13.: **(a)** Original phase image of a LQC **(b)** Initial tiling using SIFT with tripled σ and doubled edge threshold.

variations.

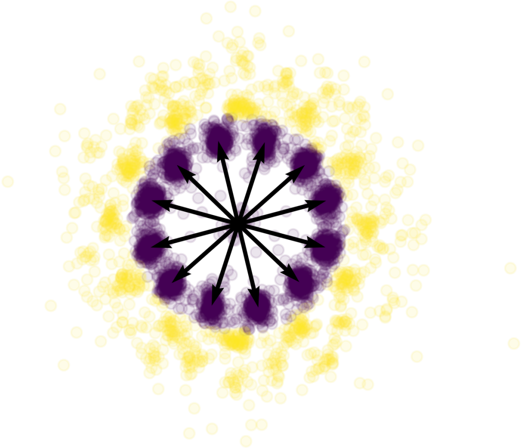


Figure 3.14.: Nearest neighbour distribution for the LQC phase image.

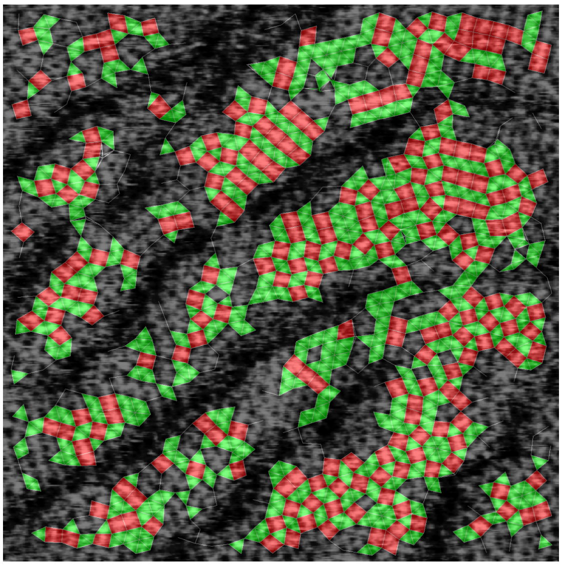


Figure 3.15.: The final tiling of the liquid QC after one run of missing point search. An attempt at a second run lead to false positive keypoint detections due to some initial features being inaccurate. This was expected from an image with many aberrations.

4. Conclusion

With the aim of facilitating pattern detection in atomic resolution images of quasicrystalline material surfaces, a python module was successfully developed and applied with promising results. Atom positions are found using features found through SIFT with subsequent filtering and clustering. Using a border following algorithm, tiles spanning the pattern are recognised from the inter-atom connections that are drawn based on nearest neighbour analysis. Atoms in close proximity missed by SIFT were found using the symmetry vectors and a SVM trained on the previously clustered features. Different types of tilings from various images were recognized with only few parameter changes. Recommended parametrisations for different image origins and resolutions were presented, which can be used as guides for future applications. The generalisation to all 2D QC surpassed expectations, but it might still be biased as there are not that many images to test the pattern recognition on. This lack of training data is also the reason as to why supervised learning from an external database was not applicable.

Furthermore, while the use of the algorithm does not work in a fully automated fashion as of now due to vast differences between images (such as resolution, relative atom size and material recorded), it does provide a massive reduction in the time spent analysing a pattern, or at least give a head start for a deeper analysis. Atom positions, edge lengths and other important information can be easily accessed through the instance attributes.

Although special attention was paid to execution speed, the module lacks in computational speed, with more detailed images taking a few minutes. Nearest neighbour search is used in multiple parts of the process, which can take a long time to complete. An option to reduce wait times would be to use a compiler such as `numba`36 in the missing point search for example.

The accuracy of the algorithm was found to depend heavily on the quality of the image, especially since the feature detection was not developed for such an application. As more QC images will be made in the future, this recognition software might help to set a foundation to build upon. With each further surface image, the generalisation of the pattern detector can be improved, for example with the emergence of an 8-fold QC image. While such an image could not be procured for testing, the algorithm successfully found and connected vertex points in an example image of of an 8-fold tiling, setting a good precedent for the future.

Glossary

Lagrangian Multiplier Variable λ introduced in a minimisation/maximisation problem by adding constraint $g(x) \neq 0$ to a function $f(x)$, such that one obtains the new Lagrangian function $\mathcal{L}(x, \lambda) = f(x) + \lambda g(x)$. Its partial derivatives are used to solve the original problem.

Lattice A periodic ordering of points in n-dimensional real space that is invariant under translation from the n vectors characterizing it

Refractive Index The proportion of the speed of light in vacuum to the speed of light in a given medium

Taylor Series Expansion A representation of a function $f(x)$ at a certain point x' as

$$\sum_{n=0}^{\infty} \frac{f^{(n)}(x')}{n!}(x - x')^n$$

Higher order terms can be left out to achieve an approximation of $f(x)$.

Acronyms

AFM Atomic Force Microscopy

ESS Error Sum of Squares

HAAADF High Angle Annular Dark Field

LQC Liquid Quasicrystal

QC Quasicrystal

QCP QuasiCrystal Pattern extractor

SIFT Scale-Invariant Feature Transform

STEM Scanning Transmission Electron Microscopy

STM Scanning Tunneling Microscopy

SVM Support Vector Machine

Bibliography

[1] Marco Corrias, Lorenzo Papa, Igor Sokolović, Viktor Birschitzky, Alexander Gorfer, Martin Setvin, Michael Schmid, Ulrike Diebold, Michele Reticcioli, and Cesare Franchini. Automated real-space lattice extraction for atomic force microscopy images. *Machine Learning: Science and Technology*, 2023.

[2] Dan Shechtman, Ilan Blech, Denis Gratias, and John W Cahn. Metallic phase with long-range orientational order and no translational symmetry. *Physical review letters*, 1984.

[3] T Ishimasa, H-U Nissen, and Y Fukano. New ordered state between crystalline and amorphous in ni-cr particles. *Physical review letters*, 1985.

[4] A. Yamamoto. Crystallography of quasiperiodic crystals. *Acta Crystallographica Section A*, 1996.

[5] F. Hoffmann. *Introduction to Crystallography.* Springer International Publishing, 2020.

[6] B. Grünbaum and G.C. Shephard. *Tilings and Patterns.* Spektrum Akademischer Verlag, 1987.

[7] S. Walter and S. Deloudi. *Crystallography of Quasicrystals: Concepts, Methods and Structures.* Springer Berlin Heidelberg, 2009.

[8] Enrique Maciá Barber. Chemical bonding and physical properties in quasicrystals and their related approximant phases: Known facts and current perspectives. *Applied Sciences*, 2019.

[9] Zbigniew M Stadnik. *Physical properties of quasicrystals.* Springer Science & Business Media, 2012.

[10] Sergei V Kalinin, Mark P Oxley, Mani Valleti, Junjie Zhang, Raphael P Hermann, Hong Zheng, Wenrui Zhang, Gyula Eres, Rama K Vasudevan, and Maxim Ziatdinov. Deep bayesian local crystallography. *npj Computational Materials*, 2021.

[11] Sergei V Kalinin, Colin Ophus, Paul M Voyles, Rolf Erni, Demie Kepaptsoglou, Vincenzo Grillo, Andrew R Lupini, Mark P Oxley, Eric Schwenker, Maria KY Chan, et al. Machine learning in scanning transmission electron microscopy. *Nature Reviews Methods Primers*, 2022.

[12] Nouamane Laanait, Maxim Ziatdinov, Qian He, and Albina Borisevich. Identifying local structural states in atomic imaging by computer vision. *Advanced Structural and Chemical Imaging*, 2016.

[13] Benjamin H Savitzky, Steven E Zeltmann, Lauren A Hughes, Hamish G Brown, Shiteng Zhao, Philipp M Pelz, Thomas C Pekin, Edward S Barnard, Jennifer Donohue, Luis Rangel DaCosta, et al. py4dstem: A software package for four-dimensional scanning transmission electron microscopy data analysis. *Microscopy and Microanalysis*, 2021.

[14] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 2004.

[15] G. James, D. Witten, T. Hastie, and R. Tibshirani. *An Introduction to Statistical Learning: with Applications in R.* Springer New York, 2013.

[16] Adam Wollman, Richard Nudd, Erik Hedlund, and Mark Leake. From animaculum to single-molecules: 300 years of the light microscope. *Open Biology*, 2015.

[17] Ariel Lipson, Stephen G. Lipson, and Henry Lipson. *Optical Physics.* Cambridge University Press, 2010.

[18] R.L. Aggarwal and A.K. Ramdas. *Physical Properties of Diamond and Sapphire.* CRC Press, 2019.

[19] D.B. Williams and C.B. Carter. *Transmission Electron Microscopy: A Textbook for Materials Science.* Springer, 1996.

[20] B. Fultz and J.M. Howe. *Transmission Electron Microscopy and Diffractometry of Materials.* Springer Berlin Heidelberg, 2013.

[21] P. Eaton and P. West. *Atomic Force Microscopy.* OUP Oxford, 2010.

[22] C. Julian Chen. *Introduction to Scanning Tunneling Microscopy.* Oxford University Press, 2007.

[23] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[24] Joe H. Ward. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 1963.

[25] Peter J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 1987.

[26] Leonard Kaufman and Peter J Rousseeuw. *Finding groups in data : an introduction to cluster analysis.* John Wiley & Sons, 1990.

[27] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 2011.

[28] Jerome H Friedman, Jon Louis Bentley, and Raphael Ari Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software (TOMS)*, 1977.

[29] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[30] Satoshi Suzuki et al. Topological structural analysis of digitized binary images by border following. *Computer vision, graphics, and image processing*, 1985.

[31] Stefan Förster, Klaus Meinel, René Hammer, Martin Trautmann, and Wolf Widdra. Quasicrystalline structure formation in a classical crystalline thin-film system. *Nature*, 2013.

[32] Weizhen Wang, Xinzhe Zhou, Zhiqing Yang, and H. Ye. A decagonal quasicrystal with rhombic and hexagonal tiles decorated with icosahedral structural units. *IUCrJ*, 2020.

[33] M Mihalkovic and M Widom. Quasicrystal approximants with novel compositions and structures. *MRS Online Proceedings Library*, 2003.

[34] Friederike Elisa Wührl, Oliver Krahn, Sebastian Schenk, Wolf Widdra, and Stefan Förster. Quasicrystal approximants in the two-dimensional ba-ti-o system on pd (111): A leed, xps, and stm study. *Physical Review B*, 2023.

[35] Ruibin Zhang, Xiangbing Zeng, and Goran Ungar. Direct afm observation of individual micelles, tile decorations and tiling rules of a dodecagonal liquid quasicrystal. *Journal of Physics: Condensed Matter*, 2017.

[36] Siu Kwan Lam, Antoine Pitrou, and Stanley Seibert. Numba: A llvm-based python jit compiler. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, pages 1–6, 2015.

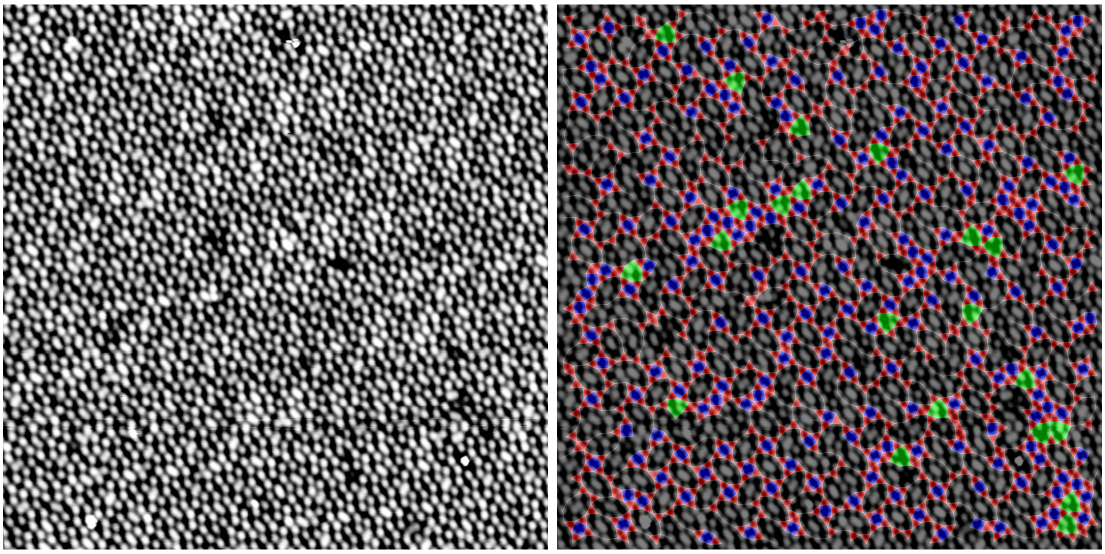
A. Appendix

A.1. Documentation

The code along with the documentation and instructions can be found at `https://github.com/InkAtom/QCP`

A.2. Approximants 3R and 8R

Along the **5R** $BaTiO_3$ approximant recorded by Wührl et al. (34) there were also areas in the surface of the material showing **3R** and **8R** unit cells. Images of these areas were used in testing the software as well, the results are presented here. The original image and the tiling in the first run are shown in figures A.1 and A.4 for the **3R** and **8R** approximant, respectively. The relatively similar nearest neighbour distributions are shown in figures A.2 and A.5. In both cases, the initial detection misses almost all points in close proximity, so the missing point search is applied. In both cases, running it once fills up most of the space. To reach the final tiling with no more missing points, both images needed 3 search runs in total. The results of the search runs for the **3R** approximant can be seen in figure A.3. As for the **8R** approximant, the tiling is shown in figure A.6. In this case, as was the problem with the **5R** approximant, there are some error regions due to multiple (valid) points in close proximity. Nevertheless, the pattern extraction is successful.



(a) Original Image

(b) Tiling

Figure A.1.: **(a)** Original image of the **3R** approximant **(b)** Initial tiling, using SIFT with default parameters.

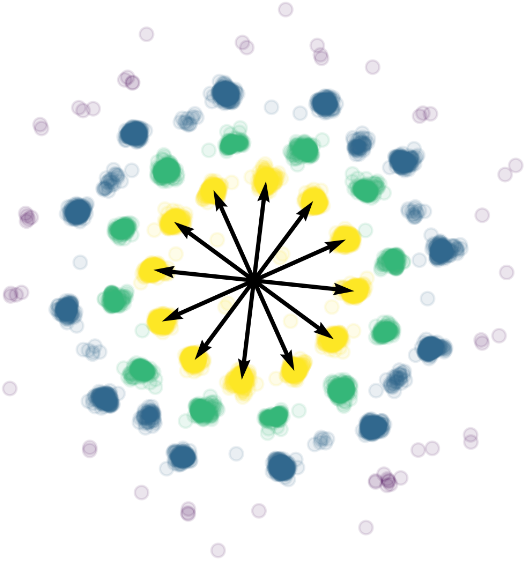
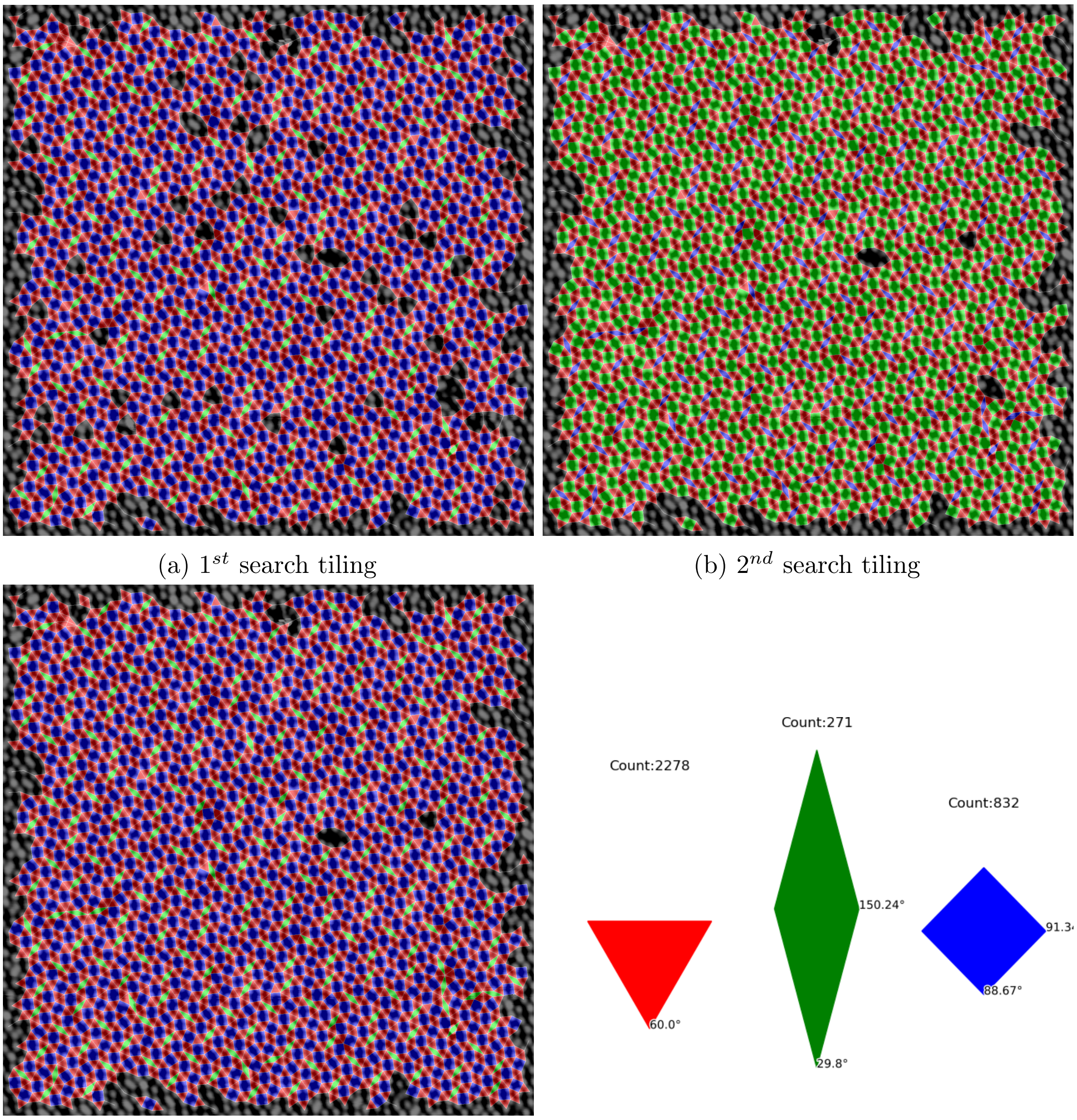


Figure A.2.: Nearest neighbour distribution for the **3R** approximant.



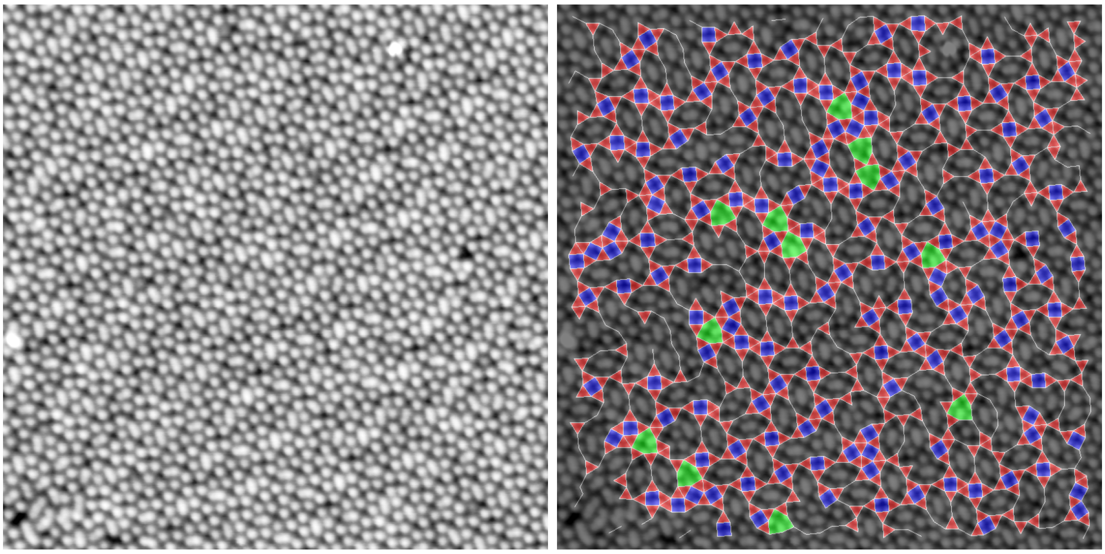
(a) 1^{st} search tiling

(b) 2^{nd} search tiling

(c) Final Tiling

(d) Individual Tiles

Figure A.3.: **(a)** Tiling of the **3R** approximant after the first missing point search. **(b)** Tiling after the second search. **(c)** Final tiling with no more further accepted candidates. **(d)** Tile makeup in the final tiling.



(a) Original Image

(b) Tiling

Figure A.4.: **(a)** Original image of the **8R** approximant **(b)** Initial tiling, using SIFT with default parameters.

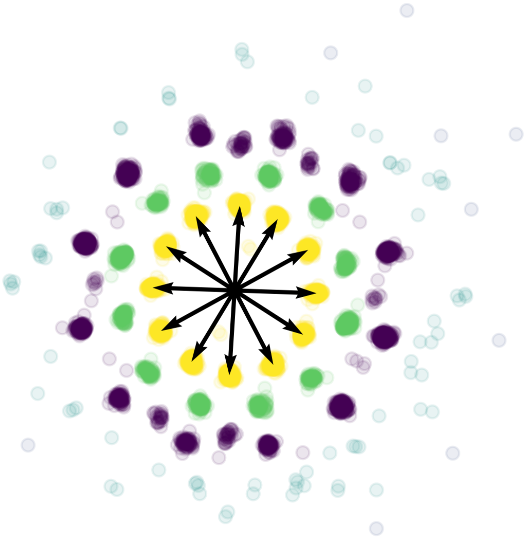
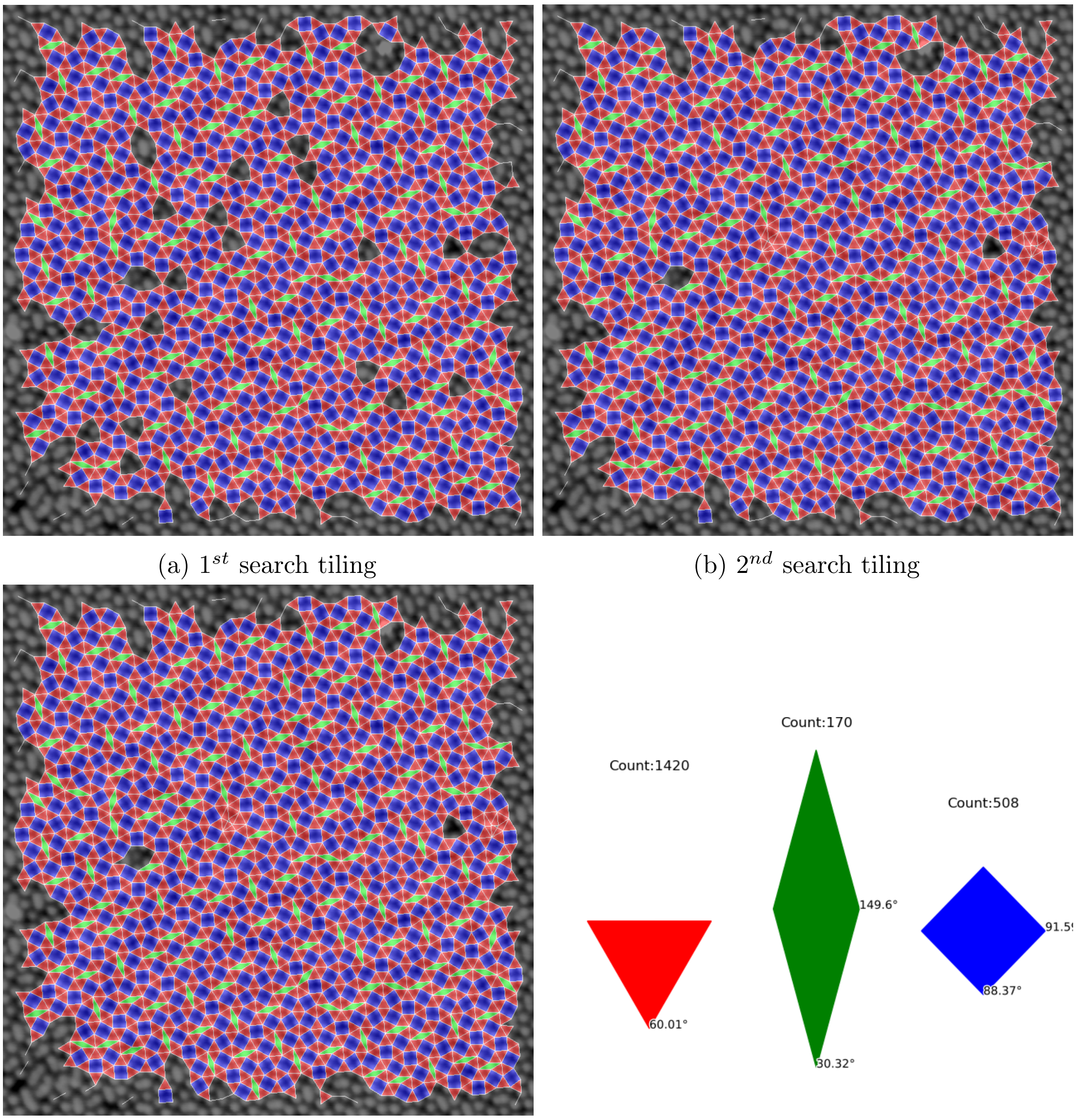


Figure A.5.: Nearest neighbour distribution for the **8R** approximant.



(a) 1^{st} search tiling

(b) 2^{nd} search tiling

(c) Final Tiling

(d) Individual Tiles

Figure A.6.: **(a)** Tiling of the **8R** approximant after the first missing point search. **(b)** Tiling after the second search. **(c)** Final tiling with no more further accepted candidates. **(d)** Tile makeup in the final tiling.