Freie Universität Berlin

Symmetry Detection and Approximation

Dissertation zur Erlangung des Doktorgrades

vorgelegt am

Fachbereich Mathematik und Informatik

der Freien Universität Berlin

im September 2012

von

Claudia Dieckmann

Institut für Informatik
Freie Universität Berlin
Takustraße 9
14195 Berlin
Claudia.Dieckmann@fu-berlin.de

Betreuer: Prof. Dr. Helmut Alt
Institut für Informatik
Freie Universität Berlin
Takustraße 9
D–14195 Berlin
Germany
alt@mi.fu-berlin.de

Gutachter: Prof. Dr. Kevin Buchin
Department of Mathematics and Computer Science
TU Eindhoven
5600 MB Eindhoven
The Netherlands
k.a.buchin@tue.nl

Vorlage zur Begutachtung: 13.09.2012
Termin der Disputation: 07.12.2012
Fassung vom: 4.02.2013

"Ein Spiegel ist ein Glas, das das Licht, das von uns reflektiert wird, wieder zurückwirft."

Linus im Juni 2011

Contents

Abstract

In this thesis, we will present algorithms to solve the following two closely related problems:

The first problem we will consider is to detect the symmetry group of a two-dimensional object even in the case where its representation is distorted by noise.

We will derive and analyze algorithms following different approaches in order to solve this problem.
One approach is to use the *discrete Fourier transform* in order to detect the symmetry group of the object. Here we assume the object to be represented by a gray-level image. The discrete Fourier transform is helpful in finding periodic structures in an input since it decomposes a signal into its fundamental frequencies. We will use this property in order to derive an algorithm which applies the discrete Fourier transform to the gray-level image and uses the result in order to determine the symmetry group of the represented object. The algorithm can be used for detecting finite as well as infinite symmetry groups.

Besides we will investigate a second method based on the probabilistic approach which is also used for solving shape matching problems. For the algorithms based on probabilistic methods we assume the object to be represented by a set of points, a set of polygonal curves or a set of parametrized curves.

The basic idea of these algorithms is to randomly choose two points out of the input set representing the object and compute the transformation (rotation or reflection) mapping the one point to the other. A vote will be generated for the computed transformation in transformation space. This procedure is repeated sufficiently often until dominant clusters in transformation space arise. The number of clusters with large numbers of votes refers to the number of symmetries of the object and thus can be used in order to compute the symmetry group of the object represented by the input set.

Both approaches described above result in algorithms which are robust against noise. Thus they derive the correct answers even if the symmetries of the object got lost during the process of computing its representation by a gray-level image or a set of geometric objects, respectively.

After detecting the symmetry group of an object represented by a point set which might be distorted by noise another interesting problem is to find a point set which is symmetric with respect to the symmetries in the detected symmetry group and which

is a close approximation of the input point set. The aim is to restore the symmetries which might have got lost during the process of representing the object in such a way that it can be processed by a computer. We assume a symmetric point set to be a close approximation of the input point set if each point of the (non-symmetric) input point set lies in the ε-neighborhood of a point in the symmetric point set. We ask this correspondence to be a bijection. This problem is called the *ε-Symmetry Detection Problem*(ε-SD problem).

The ε-SD problem was already studied by Iwanowski [18] and he proved it to be NP-complete in general. For some restricted versions of the ε-SD problem Iwanowski proved the decision problem to be in P. For those we will present polynomial time algorithms solving the corresponding optimization problems. Additionally we will present polynomial time algorithms for some restricted versions which were not considered until now. One possible restriction is to only allow point sets which are well-separated. Iwanowski proved the ε-SD problem to be in P in the case where no two points have a distance smaller than 8ε and proved it to be NP-complete in the case where the point set is at most $\frac{\varepsilon}{2}$-disjoint. We will improve this result by developing polynomial time algorithms for $4(1+\delta)\varepsilon$-disjoint point sets for each $\delta > 0$.

The algorithms developed in this thesis which detect the symmetry group of an object were implemented and tested using the programming language JAVA.

Zusammenfassung

In dieser Arbeit werden zwei eng miteinander verwandte Probleme betrachtet.

Zum Einen wird untersucht, wie die Symmetriegruppe eines zweidimensionalen Objektes bestimmt werden kann.

Wird ein Objekt so dargestellt, dass es mit Hilfe eines Computerprogramms untersucht werden kann, so kann es passieren, dass die Symmetrien des ursprünglichen Objekts in der Darstellung des Objekts nicht mehr vorhanden sind. Die Algorithmen, die in dieser Arbeit entwickelt werden, bestimmen die Symmetriegruppe eines Objektes auch dann, wenn die Darstellung des Objektes selber nicht mehr symmetrisch ist. Abhängig von der Darstellung des Objekts werden verschiedene Algorithmen vorgestellt und untersucht.

Eine Möglichkeit ist, das Objekt mit Hilfe eines Graustufenbilds (z.B. als .jpg oder .png Datei) darzustellen. In diesem Fall wird das Bild mit Hilfe der *diskreten Fourier-Transformation* analysiert und so die Symmetriegruppe ermittelt. Die Fourier-Transformation zerlegt ein Signal in seine Grundschwingungen und kann daher verwendet werden, um periodische Strukturen zu erfassen. Diese Eigenschaft wird genutzt, indem die diskrete Fourier-Transformation auf das Graustufenbild angewendet und mit Hilfe der entdeckten periodischen Struktur in diesem Bild die Symmetriegruppe des dargestellten Objekts ermittelt wird. Der so entwickelte Algorithmus kann verwendet werden, um sowohl endliche als auch unendliche Symmetriegruppen zu ermitteln.

Des Weiteren wird ein probabilistischer Ansatz verwendet, um die Symmetriegruppe eines Objekts zu ermitteln, wobei hier von der Repräsentation des Objekts als Punktemenge, Menge von Polygonzügen oder als Menge von parametrisierten Kurven ausgegangen wird.
Die Grundidee dieses Algorithmus' ist, zwei Punkte zufällig aus der Punktmenge, die das Objekt repräsentiert, auszuwählen und dann die Transformation (Rotation oder Spiegelung) zu berechnen, die den einen Punkt auf den anderen abbildet. Im Raum der Transformationen wird dann eine Stimme für die ermittelte Transformation erzeugt. Dieses Vorgehen wird ausreichend oft wiederholt, so dass sich Cluster im Transformationsraum bilden. Die Anzahl der Cluster mit einer großen Anzahl von Stimmen entspricht der Anzahl der Symmetrien des Objekts und kann daher verwendet werden, um die Symmetriegruppe des dargestellten Objekts zu bestimmen.

Mit Hilfe der beiden beschriebenen Ansätze werden jeweils Algorithmen entwi-

ckelt, die die Symmetriegruppe eines zweidimensionalen Objekts ermitteln, auch wenn die ursprünglichen Symmetrien des Objekts durch die Darstellung als Graustufenbild oder als Menge von geometrischen Objekten verloren gegangen sind, da beide Algorithmen robust gegenüber Störungen sind.

Nachdem die ursprüngliche Symmetriegruppe eines Objekts, das durch eine Punktmenge repräsentiert wird, ermittelt wurde, ist die Konstruktion einer Punktmenge mit der ermittelten Symmetriegruppe ein weiteres interessantes Problem. Ist die so erzeugte symmetrische Menge eine gute Approximation der Menge, die das Objekt darstellt, so ist sie eine *symmetrische* Darstellung des ursprünglichen Objekts. Wir betrachten eine symmetrische Punktmenge als gute Approximation einer Eingabemenge, falls jeder Punkt dieser Menge in der ε-Umgebung eines Punktes der symmetrischen Punktmenge liegt, wobei diese Zuordnung bijektiv sein muss.

Dieses Problem wurde von Iwanowski [18] untersucht und als *ε-Symmetry Detection Problem* (ε-SD Problem) bezeichnet, wobei er das Problem als Entscheidungsproblem formuliert. Iwanowski zeigte in seiner Arbeit, dass das ε-SD Problem im Allgemeinen NP-vollständig ist und für eingeschränkte Varianten in P liegt. Für einige dieser eingeschränkten Varianten werden in dieser Arbeit Polynomialzeitalgorithmen angegeben, die das zugehörige Optimisierungsproblem lösen. Außerdem werden bisher noch nicht untersuchte eingeschränkte Varianten betrachtet und es werden Polynomialzeitalgorithmen entwickelt, die diese entscheiden.

Das ε-SD Problem kann dadurch eingeschränkt werden, dass nur Punktmengen als Eingabe erlaubt werden, deren Punkte weit auseinander liegen. Iwanowski zeigte in seiner Arbeit, dass das ε-SD Problem in P ist, falls keine zwei Punkte der Menge einen Abstand kleiner als 8ε haben. In dieser Arbeit wird Iwanowkis Ergebnis verbessert, indem Polynomialzeitalgorithmen für Punktmengen angegeben werden, in denen keine zwei Punkte einen Abstand kleiner als $4(1+\delta)\varepsilon$ haben, für alle $\delta > 0$.

Die Algorithmen für die Symmetrieerkennung, die in dieser Arbeit vorgestellt werden, wurden mit Hilfe der Programmiersprache JAVA implementiert und getestet.

Acknowledgements

I was supported by many people while working on this thesis.

First of all I would like to thank my advisor Helmut Alt for suggesting the topic of this thesis and for many intensive and helpful discussions. I always felt welcome to ask for his time and opinion. I also thank him for excellent guidance not only while working on this thesis but also in terms of teaching and working with students.

I want to thank Uli Kortenkamp, the advisor of my diploma thesis, for suggesting to apply for this position and to work on this thesis. He also created the interactive geometry tool Cinderella which helped me a lot while proving or disproving my assumptions.

I want to thank the whole work group for the great atmosphere, many nice conversations as well as numerous funny and interesting coffee sessions. I thank Britta, Astrid and Ludmila for often spending the lunch time with me, Frank and Klaus for the coffee in the morning, and Oli for his incredible stories when entering the office. Especially I thank Lena not only for sharing the office with me, but also for good and productive discussions about my work.

I would also like to thank my family, especially my parents for raising, loving, and supporting me with their time and their trust in me. Having you as parents is a great fortune. I also thank my parents-in-law for their encouragement. All four are great and caring grandparents and were always there to take care of Linus and Jonathan when ever it was needed. I thank Karin for struggling through my interpretation of the English grammar and correcting the randomly positioned commas.

My children Linus and Jonathan I thank for being the way they are and for showing me what really is important in life whenever my head was stuck in clouds of theoretical computer science. To them I'm finally able to say: "The 'thick book' is finished now!"

I use this last line to thank my husband Jörg - for being Jörg.

Introduction

Symmetry arises in almost all objects in our world no matter if they are man-made or naturally grown. In flora and fauna, most of the creatures have at least one symmetry axis. Almost all animals exhibit mirror symmetry with respect to a plane. One famous exception is the flatfish, which has both eyes on one side, and thus is asymmetrical. Interestingly, the flatfish is symmetric at the beginning of its life, since it hatches with an eye on each side. Only when it is around 8 mm long, it starts to metamorphose and one eye moves to the other side.

Most people associate the term "symmetry" with beauty, harmony of proportions, regular structures and evenness. This might be the case since we are surrounded by symmetry in all our life.

No natural object is perfectly symmetric when considering symmetry from the mathematical point of view. Here an object is considered symmetric if the image of a transformation (the symmetry) applied to the object is the object itself.

The human perception does not distinguish between perfect symmetry in the mathematical sense and slightly distorted symmetry. Thus we consider the frontal view of a person or an animal to be symmetric even if, for example, the eyes are not exactly the same size or are not at perfectly symmetric positions.

Figure 0.0.1: Front view of a zebra is considered to be mirror symmetric. By the mathematical definition of symmetry this is not true.

There are studies on how the attractiveness of a person correlates to the amount of his or her physical symmetry [9].

Since humans are attracted by symmetry, it is not astonishing that symmetry can be found in numerous man-made objects.
In his book "Symmetry", Weyl [40] gives many examples of how symmetry is used in art and design throughout the ages.

The reason why man-made objects are symmetric differs depending on the purpose of the objects. Symmetry is helpful for ease of construction (floor plans), ease of recognition (trademarks) or distinction (street signs), functionality (furniture) or beauty (art and design).

In this thesis we will present algorithms detecting and restoring symmetries of two-dimensional objects. Since most of the objects we are surrounded by are three-dimensional we assume the objects to be preprocessed and represented in two dimensions. Most of the symmetric objects in three dimensions are symmetric with respect to a reflection plane or with respect to rotations around a line. Representing these objects in a way that the reflection plane is projected to a reflection line and the line which is the center of the rotation is projected to a rotation point, the symmetry information of the object is not lost during the projection.

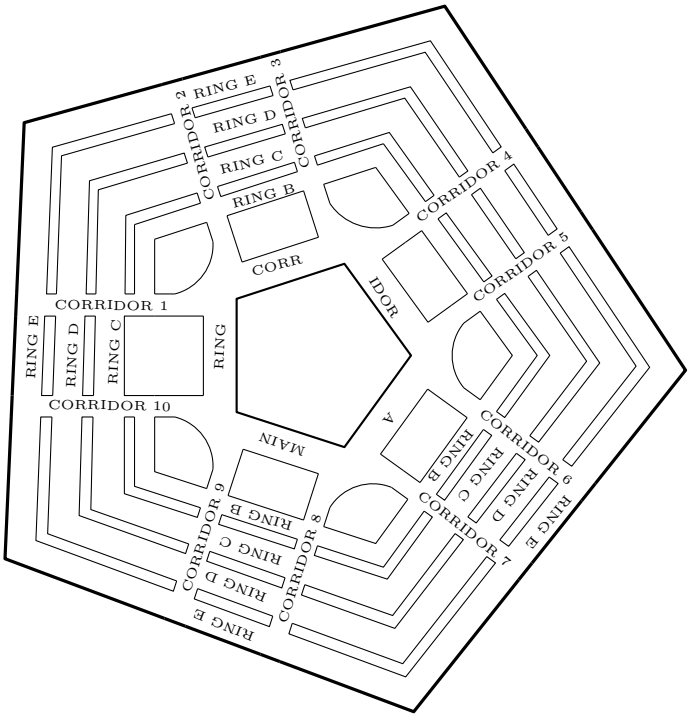
This projection can be done by taking a frontal view picture if the object considered is, for example, a person or an an animal. In the case of buildings, the floor plan is a representation of the building preserving the symmetries.

As argued above, not all objects which are considered to be symmetric by human vision are perfectly symmetric. Even if the original object contains perfect symmetry and even if the picture of the object is carefully taken in order to preserve the symmetries of the object, the two-dimensional object may not be perfectly symmetric any more due to the perspective.

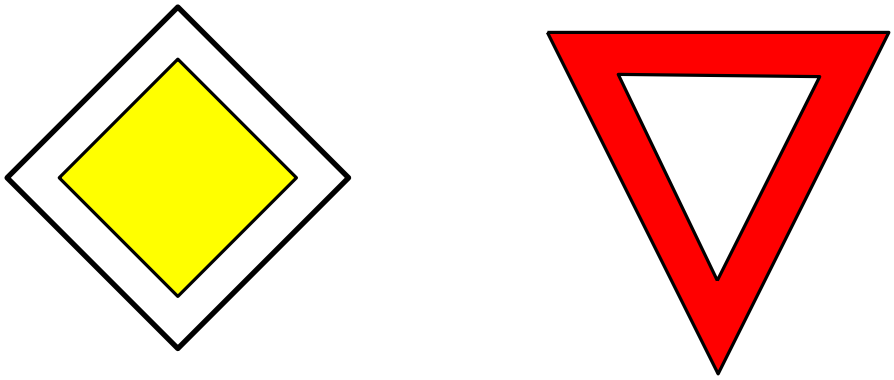
In this thesis we will present and discuss algorithms detecting the symmetry group of two-dimensional objects even if the symmetries got lost due to slight perturbations during the process of representing them in such a way that it can be processed by a computer. We will also present algorithms in order to reconstruct the symmetries of the original object.

Organization of this Thesis

In Chapter 1 we will introduce and define the notion of symmetry and symmetry groups. We will consider only point sets in the plane. A point set in the plane can exhibit symmetries with respect to rotations, reflections or translations. We will define those three affine transformations in Section 1.1.1 (rotations), Section 1.1.2 (reflections) and Section 1.2 (translations and glide-reflections), respectively.
The symmetry group of a point set is given by the symmetries of the set. In Section 1.1 we will define the *cyclic* and *dihedral* symmetry groups, which contain a finite number of symmetries. The dihedral groups contain reflections as well as rotations,



(a) Sketch of the floor plan of the pentagon.



(b) The yield sign is a regular triangle whereas the right of way signs shape is a square. Thus they are easy to distinguish.

Figure 0.0.2: Examples of symmetric man-made objects.

whereas the cyclic symmetry groups only contain rotations. The different cyclic and dihedral symmetry groups are distinguished by the number of rotations they contain. In Section 1.2 we will investigate *infinite symmetry groups*. In the plane there are two types of infinite symmetry groups, namely the *frieze groups* and the *wallpaper groups*. The translations contained in a frieze group all have the same direction whereas the translations in a wallpaper group are generated by two translations. We will see in Section 1.2 that a pattern with infinite symmetry group consists of a basic pattern which is repeated in one direction in the case of a frieze group

and in two directions in the case of a wallpaper group. The different frieze groups and wallpaper groups, respectively, are distinguished by the symmetries (half-turns, reflections, glide-reflections) of the basic pattern.
A decision tree for determining the frieze group of a pattern is given in Section 1.2.

In Chapter 2 we will present different approaches which can be used to detect the symmetry group of a two-dimensional object. In Section 2.2 we assume the object to be represented by a gray-level image. We will discuss how the symmetry group of the object can be detected by using string matching algorithms (Section 2.2.1), the correlation (Section 2.2.1) as well as the discrete Fourier transform (Section 2.2.2). We will also discuss the advantages and disadvantages of these methods. In Section 2.3 we will assume the object to be represented by a set of points, a set of geometric objects, or a set of parametrized curves. We will present algorithms that are based on the probabilistic approach which also is used for algorithms solving shape matching problems (see [2], [38]). We will explain the modifications needed in order to use this approach for detecting the symmetry group of the represented object.
All developed algorithms can be used to detect the symmetry group of the two-dimensional object even if it contains noise.

The algorithms described in Chapter 2 can be used to detect the symmetry group of a given input image, even if it does not contain perfect symmetry. In some applications, it is interesting to know how close the input image is to a perfectly symmetric image, or to find a perfect symmetric image that is a good approximation of the given one. One application might be to fix an originally symmetric image, where the symmetry got lost due to some preprocessing steps. Let for example the input be the photography of some symmetric shape, where the perfect symmetry got lost because of the position of the camera. Here it might be helpful to restore the symmetry of the object, by finding the symmetric shape that is close to the input picture. This problem is referred to as the ε-symmetry detection (ε-SD) problem. It was studied by Iwanowski [18] in his PhD thesis. He was able to prove the problem to be NP-complete in general and to be in P for the cyclic symmetry group C_2 and the dihedral symmetry group D_1. In Chapter 3 we investigate the ε-SD optimization problem and state polynomial time algorithms for many variants of this problem. We will also improve some of the results of Iwanowski.

We will close this thesis by testing the algorithms stated in Chapter 2 and Chapter 3 and discussing the experimental results.

Chapter 1

Symmetry Groups

In this chapter we will introduce the notions of symmetry and symmetry groups . We will start by defining the three basic affine transformations we will use throughout this thesis. These three affine transformations are the rotation, the reflection with respect to a reflection line, and the translation.

We will call an affine transformation a *symmetry* of a point set P, iff the image of P with respect to the affine transformation is P itself.

The set of symmetries of a point set P forms a group and is called the *symmetry group* of P. The symmetry group of a point set can be finite or infinite. Point sets having an infinite symmetry group are always infinite themselves whereas point sets having a finite symmetry group may be finite or infinite.

In the two-dimensional space there are two different kinds of finite symmetry groups. The *cyclic symmetry groups*, denoted by C_k, only contain rotations whereas the *dihedral symmetry groups*, denoted by D_k, contain rotations as well as reflections. The index $k \in \mathbb{N}^+$ in both cases denotes the number of rotations in the symmetry group.

Symmetry groups which contain translations are always infinite. A symmetry group where the set of generating translations is finite is either a *frieze group* or a *wallpaper group*. All translations in a frieze group are generated by one translation, therefore all translations in a frieze group have the same direction. Wallpaper groups contain two generating translations with different directions.

A point set in the plane having a frieze or wallpaper group as its symmetry group consists of a basic pattern which is repeated infinitely often. The size of such a pattern is given by the length of the generating translations. There is not only one well defined basic pattern for each point set, but there may exist several basic patterns. The dilatation of a basic pattern is one-dimensional in the case of frieze groups and two-dimensional in the case of wallpaper groups.

There are seven different frieze groups and seventeen different wallpaper groups. One distinguishes the different frieze groups and wallpaper groups, respectively, by the symmetries of the basic pattern.

In the following sections we will state the definitions and theorems about sym-

metries and symmetry groups needed in this thesis. A detailed introduction to affine transformations, symmetries and symmetry groups in the plane as well as in $\mathbb{R}^3$ can be found in the book "Transformation Geometry" of Martin [27].

We will only investigate symmetries of point sets in the plane throughout this thesis.

1.1 Finite Symmetry Groups

In this section we will investigate rotations and reflections and we will define the notion of symmetric point sets with respect to those two affine transformations. Furthermore, we will define the two types of finite symmetry groups, namely the *cyclic* and the *dihedral* symmetry groups. We also will give examples of point sets having cyclic and dihedral symmetry groups, respectively. Most of the definitions and properties of symmetries and symmetry groups we will present in this section can be found in the book of Martin [27].
Throughout this section we will use the following notations:

Notation 1.1.1. *We denote the image of a point $p \in \mathbb{R}^2$ with respect to a rotation or a reflection by $p' \in \mathbb{R}^2$.*
The Euclidean distance between two points $p, q \in \mathbb{R}^2$ is denoted by $d(p, q)$.

1.1.1 Rotations

We will start by defining the *rotation* of a given point $p \in \mathbb{R}^2$ around a rotation center $c \in \mathbb{R}^2$ by an angle α. A rotation is an affine transformation and can therefore be represented by a transformation matrix and a translation vector. For ease of understanding, we will give a geometrical definition of a rotation and will later on explain how the transformation matrix and the translation vector can be retrieved from this definition.

Definition 1.1.2. Let $p \in \mathbb{R}^2$ and $c \in \mathbb{R}^2$ be points in the plane and let $\alpha \in \mathbb{R}$ be an angle. The counterclockwise (ccw) rotational image of p is the unique point p', such that $d(p, c) = d(p', c)$ and the angle in ccw direction between $\overline{pc}$ and $\overline{p'c}$ is α.

The clockwise rotation is defined analogously. An illustration of Definition 1.1.2 can be found in Figure 1.1.1.

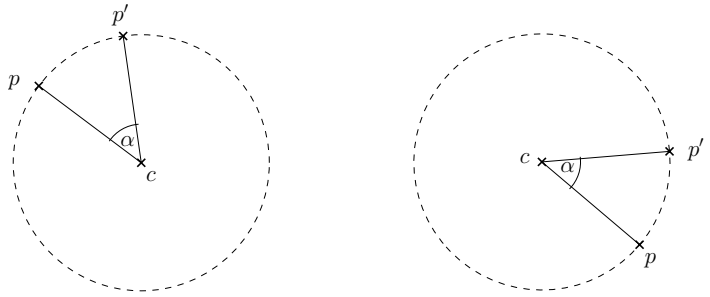


Figure 1.1.1: The clockwise (left) and counter clockwise rotation (right) of the point p around c by the angle α yields p', respectively.

The rotational image of a point p with respect to the rotation center c and angle α can be computed as follows:

Lemma 1.1.3. *Let a point $p \in \mathbb{R}^2$, a rotation center $c \in \mathbb{R}^2$ and an angle $\alpha \in \mathbb{R}$ be given. The counter clockwise rotational image of p rotated around c by α is given by:*

$$p' = R^\alpha(p - c) + c,$$

where R^α denotes the rotation matrix

$$R^\alpha = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix}$$

The proof can be found in the book of Martin [27].

We will abbreviate the rotation as follows:

Notation 1.1.4. *The transformation of rotating a point p* counterclockwise *around a rotation center c by an angle α is denoted by $\rho_c^\alpha(p)$.*

The transformation of rotating a point p clockwise *around a rotation center c by an angle α is denoted by $\rho_c^{-\alpha}(p)$.*

Let P be a point set in the plane, then $\rho_c^\alpha(P) = \{\rho_c^\alpha(p) | p \in P\}$ denotes the point set containing all rotational images with respect to c and α of points in P.

Using the definition and properties of rotations stated above, a point set having rotational symmetry is defined as follows:

Definition 1.1.5. Let $P \subset \mathbb{R}^2$ be a point set in the plane. Let $c \in \mathbb{R}^2$ be a rotation center and $\alpha \in \mathbb{R}$ be a rotation angle. The rotation ρ_c^α is a symmetry of P iff $P = \rho_c^\alpha(P)$. We call P *rotational symmetric* with respect to ρ_c^α.

An important property of the set of symmetries of a point set is that they form a group.

Lemma 1.1.6. *Let P be a set of points in the plane. Let P be symmetric with respect to the rotation $\rho_c^{\frac{2\pi}{k}}$ for a rotation center $c \in \mathbb{R}^2$ and a number $k \in \mathbb{N}^+$ and let k be maximal with this property. Let furthermore P be not symmetric with respect to any reflection or translation.*

Then the set of symmetries of P is $C_k = \{\rho_c^{j\frac{2\pi}{k}} | 0 \leq j \leq k-1\}$. The set C_k forms a cyclic group with respect to composition with generator $\rho_c^{\frac{2\pi}{k}}$ and is called cyclic symmetry group *of P.*

For the proof we refer to Martin [27].

Example 1.1.7. Consider the example depicted in Figure 1.1.2:

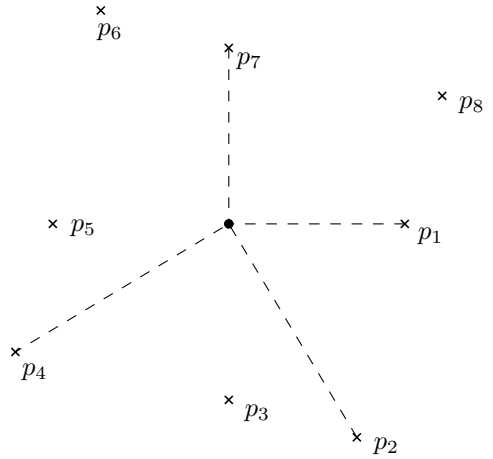
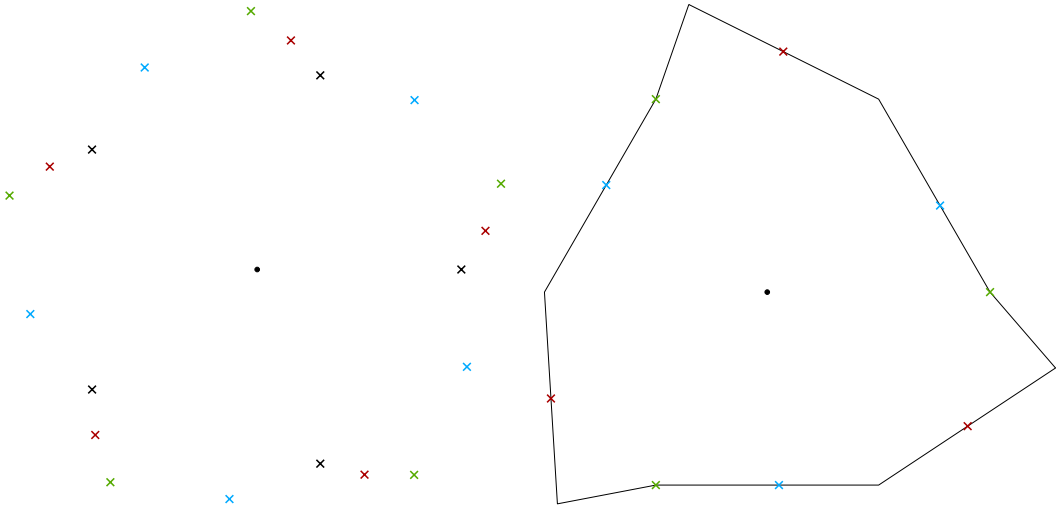


Figure 1.1.2: Example of a rotational symmetric point set with rotation center c and rotation angle $\frac{\pi}{2}$.

Figure 1.1.2 shows the point set $P = \{p_1, \ldots, p_8\}$ which has rotational symmetry group C_4. The rotation center is denoted by c and the generator of the cyclic group is in this case $\frac{\pi}{2}$. Some of these pairs are marked in Figure 1.1.2, e.g. $p_7 = \rho_c^{\frac{\pi}{2}}(p_1)$ and $p_2 = \rho_c^{\frac{\pi}{2}}(p_4)$.

We see in this example that the point set P can be divided into two subsets $P_O = \{p_1, p_3, p_5, p_7\}$ and $P_E = \{p_2, p_4, p_6, p_8\}$ each containing four elements. Both sets on their own are symmetric with respect to the rotations in the set $C_4 = \{\rho_c^{j\frac{\pi}{2}} | 0 \leq j \leq 3\}$.

A point set $|P| = n$ with symmetry group C_k is always the union of $\frac{n}{k}$ sets of size k where the elements of these subsets are the vertices of a regular k-gon and all k-gons have the same rotation center. A point sets containing infinitely many points and having symmetry group C_k is the composition of infinitely many regular k-gons all having the same rotation center (see Figure 1.1.3).



(a) C_5-symmetric finite point set.

(b) C_3-symmetric infinite point set. Some subsets of size three are marked with different colors.

Figure 1.1.3: Examples of point sets with cyclic symmetry group.

1.1.2 Reflections

In this section we will define and investigate reflections with respect to a reflection line l. The reflection is an affine transformation and can therefore be represented by a transformation matrix and a translation vector. As for the rotations, we will first give a geometrical definition of a reflection and state lateron the transformation matrix and the translation vector representing the reflection. Furthermore, we will define the notion of a mirror symmetric point set and introduce the *dihedral symmetry groups* D_k, which contain rotations as well as reflections.

We define a reflection as follows:

Definition 1.1.8. Let a point $p \in \mathbb{R}^2$ in the plane and a line $l \subset \mathbb{R}^2$ be given. Then $p' \in \mathbb{R}^2$ is the image of p with respect to the reflection given by l iff l is the perpendicular bisector of the line segment $\overline{pp'}$.

One can geometrically construct p' from p and l by first constructing $l^\perp$ as the line through p and perpendicular to l. Let c be the intersection point of l and $l^\perp$. The mirror image p' of p with respect to l is then the intersection point of $l^\perp$ and the boundary of the disk D, where D has center c and radius $d(c, p)$. This construction is illustrated in Figure 1.1.4.

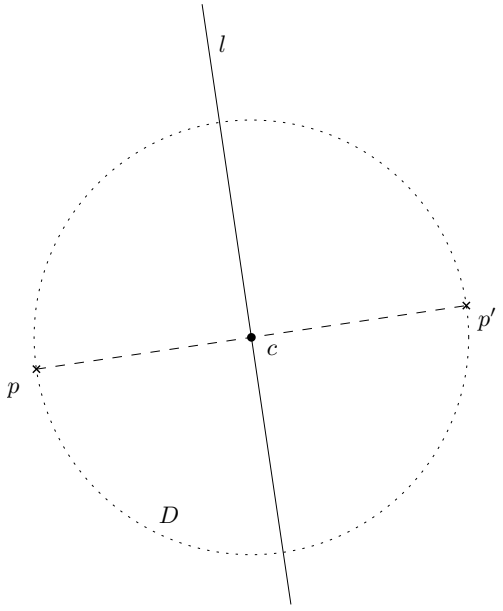


Figure 1.1.4: The construction of the mirror image of the point p with respect to the reflection line l.

The transformation matrix and translation vector which define a reflection are given by Lemma 1.1.9.

Lemma 1.1.9. *Let $p \in \mathbb{R}^2$ be a point in the plane and let $l(x) = mx + c$ be a line in the plane. The mirror image of p w.r.t l is given by the following transformation matrix and translation vector:*

$$\begin{pmatrix} p'_x \\ p'_y \end{pmatrix} = \underbrace{\begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix}}_{R^{\alpha}} \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}}_{S^x} \underbrace{\begin{pmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{pmatrix}}_{R^{-\alpha}} \underbrace{\begin{pmatrix} p_x \\ p_y - c \end{pmatrix}}_{t^{-c}} + \underbrace{\begin{pmatrix} 0 \\ c \end{pmatrix}}_{t^{c}}$$

where $\alpha = \arctan m$

Proof. A formal proof can be found by applying the affine transformation to p and verifying that l is the perpendicular bisector of p and p'. We only give the basic idea of the proof. The transformation matrix defining the reflection w.r.t the x-axis is given by the matrix S^x. In order to align the reflection line l with the x-axis, the scenery is shifted and rotated by applying the translation vector t^{-c} and the rotation matrix $R^{-\alpha}$. Applying the matrix S^x gives the mirror image with respect to the x-axis which corresponds to the line l. The scenery is rotated and shifted back by applying the inverse rotation and translation R^{α} and t^c, respectively. □

Notation 1.1.10. *The reflection with respect to the reflection line l is denoted by σ_l.*

The product of two reflections σ_{l_1} and σ_{l_2} is either a rotation in the case where the two lines intersect or a translation in the other case where the two lines are parallel.

Lemma 1.1.11. *Let two lines $l_1 \subset \mathbb{R}^2$ and $l_2 \subset \mathbb{R}^2$ be given.*

The affine transformation $\sigma_{l_2} \circ \sigma_{l_1}$ is the rotation $\rho_c^{2\alpha}$ iff l_1 and l_2 intersect in point c and α is the angle in counterclockwise direction between l_1 and l_2.

For two parallel lines l_1, l_2 the affine transformation $\sigma_{l_2} \circ \sigma l_1$ is a translation in direction perpendicular to l_1 and l_2 of length $2d(l_1, l_2)$, where $d(l_1, l_2)$ is the distance between l_1 and l_2.

For the proof we refer to the book of Martin [27].

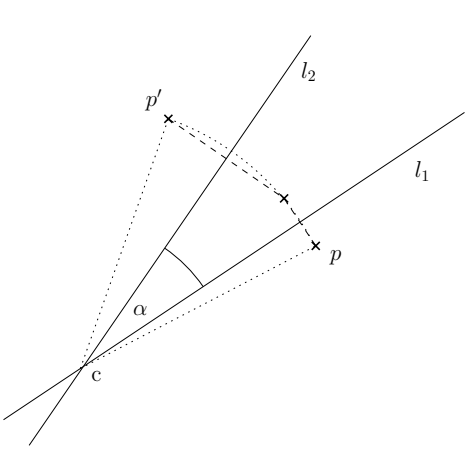
For illustrations of Lemma 1.1.11 see Figure 1.1.5.

The composition of a rotation and a reflection is a reflection again if the reflection line passes through the rotation center:

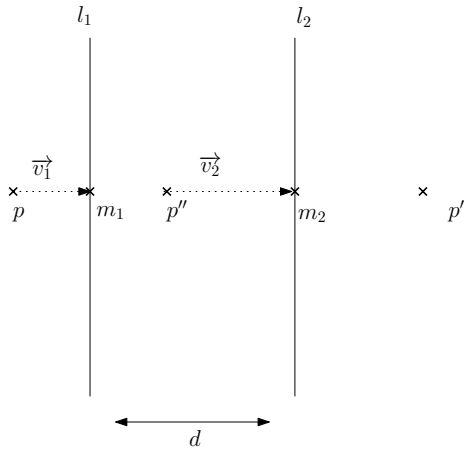
Lemma 1.1.12. *Let a reflection σ_l and a rotation ρ_c^{α} be given. The product $\sigma_l \circ \rho_c^{\alpha}$ is a reflection $\sigma_{l'}$, iff c lies on l. The line l' also passes through c and the angle between l and l' is $\frac{\alpha}{2}$.*

For the proof we again refer to the book of Martin [27].
For illustration of Lemma 1.1.12 see Figure 1.1.6.

A mirror symmetric point set is defined as follows:



(a) The product $\sigma_{l_2} \circ \sigma_{l_1}$ is the rotation $\rho_c^{2\alpha}$.



(b) The composition of two reflections where the reflection lines are parallel is a translation.

Figure 1.1.5: Illustration of Lemma 1.1.11.

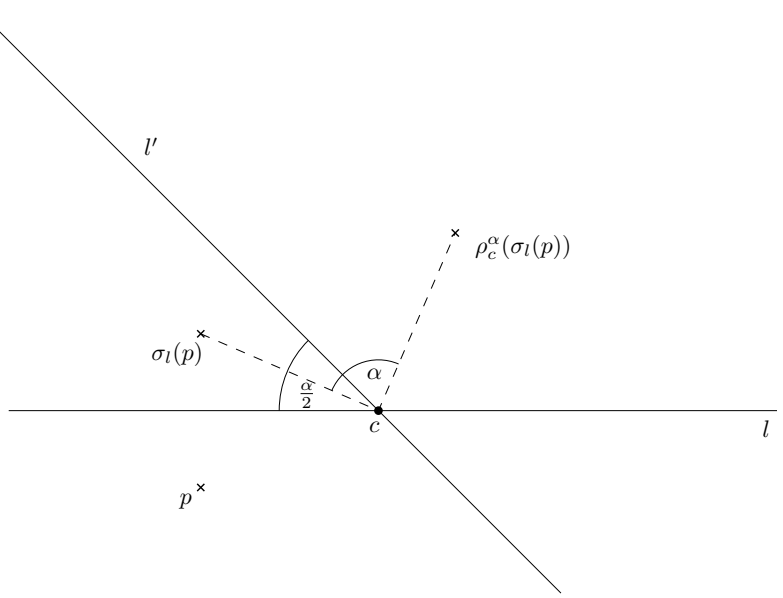


Figure 1.1.6: Illustration of Lemma 1.1.12.

Definition 1.1.13. Let $P \subset \mathbb{R}^2$ be a point set in the plane and $l \subset \mathbb{R}^2$ be a line. We call P *mirror symmetric* with respect to the reflection line l, iff $P = \sigma^l(P)$.

Lemma 1.1.14. *Let P be a point set which is symmetric with respect to $\rho_c^{\frac{2\pi}{k}}, k \in \mathbb{N}$ for some rotation center $c \in \mathbb{R}^2$ and with respect to σ_l for some reflection line l passing through c and let k be maximal with this property. Then the set of symmetries of P is $D_k = \{\rho_c^{j\frac{2\pi}{k}} | 0 \leq j \leq k-1\} \cup \{\sigma_{l_j} | l_j = \rho_c^{j\frac{\pi}{k}}(l), 0 \leq j \leq k-1\}$.*

D_k forms a group and is called the dihedral symmetry group *of P.*

A proof can be found in the book of Martin [27].

The only finite symmetry groups in the plane are the cyclic and the dihedral symmetry groups. For a proof we refer to Martin [27].

Example 1.1.15. As an example of a point set $P = \{p_1, \ldots, p_{12}\}$ with symmetry group D_4 see the point set depicted in Figure 1.1.7.

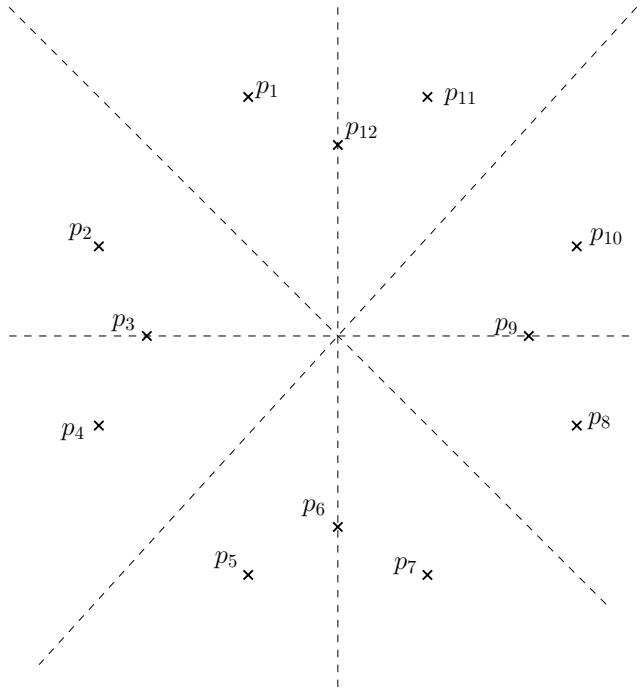


Figure 1.1.7: Example of a D_4-symmetric point set. The dashed lines are the four reflection lines.

1.2 Infinite Symmetry Groups

In this section we will introduce the two types of infinite symmetry groups in the plane, namely the *frieze groups* and the *wallpaper groups*. Frieze groups as well as wallpaper groups always contain translations which are defined as follows:

Notation 1.2.1. *Let a point $p \in \mathbb{R}^2$ in the plane and a vector $v \in \mathbb{R}^2$ be given. We denote by $p' = \tau_v(p)$ the image of the translation τ_v, iff $p' = p + v$.*

Definition 1.2.2. The translation τ_v is a symmetry of a point set $P \subset \mathbb{R}^2$, iff $\tau_v(P) = P$. We call P *translational symmetric* w.r.t τ_v.

Some point sets are symmetric w.r.t the product of a translation and a reflection, although they are not symmetric w.r.t. the reflection itself:

Definition 1.2.3. Let a reflection σ_l and a translation τ_v be given. The product $\gamma_{l,v} = \sigma_l \circ \tau_v$ is called a *glide-reflection*.

1.2.1 Frieze Groups

In the case where the translations in the set of the symmetries of a point set P all have the same direction, the symmetry group is called *frieze group*. There are seven frieze groups in the plane. They are distinguished by the rotations and reflections which are contained additionally to the translation in the symmetry group.

The only rotations and reflections that are contained in a frieze group are half-turns and reflections where the reflection line either has the same direction as the translation vector or is perpendicular to it. A proof of these properties can be found in the book of Martin [27].

In the case where the symmetry group of a point set P is a frieze group, P consists of a basic point set $P_B \subset P$ which is translated with respect to the translations in the symmetry group. The length of the basic pattern is given by the length of the generating translation of the frieze group. The symmetry group is defined by the symmetries of P_B.

The scheme depicted in Figure 1.2.1 for determining the frieze group of a point set P is taken from the book of Martin [27].

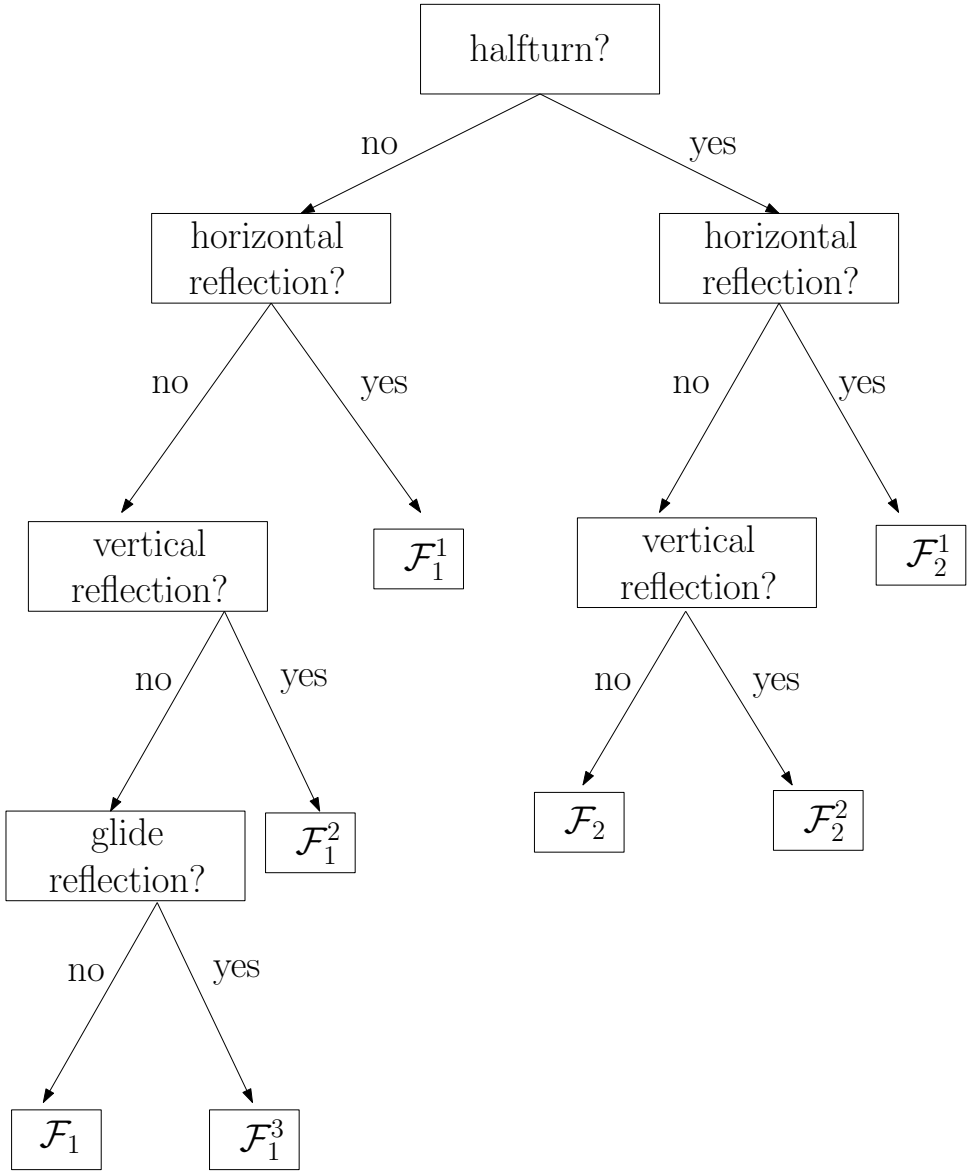
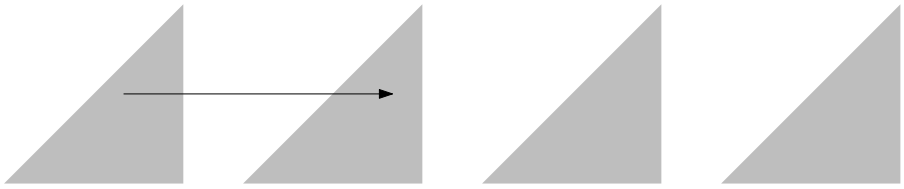
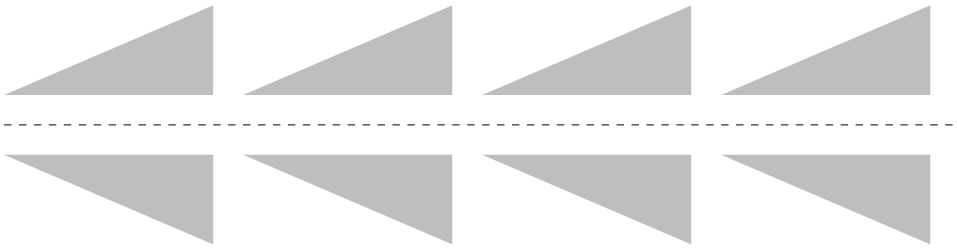


Figure 1.2.1: Scheme for determining the frieze group of a point set in the plane. The frieze group of a point set P can be determined by the symmetries of the basic point P_B. By the terms "horizontal reflection" and "vertical reflection" we mean reflection at a line parallel to the direction of the translation vector and reflection at a line perpendicular to the direction translation vector, resp.

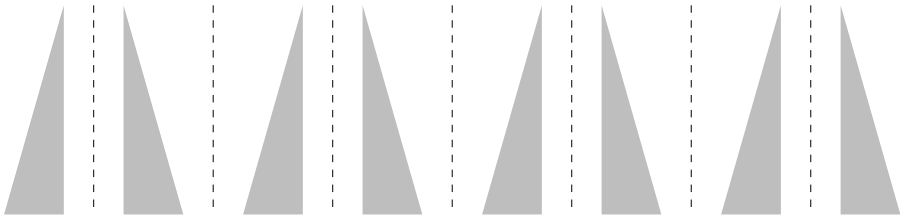
We give an example for each of the seven frieze groups in Figure 1.2.2.



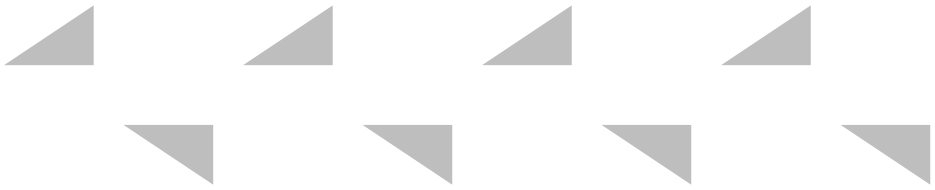
(a) Point set having frieze group F_1. The symmetries are given by the translation.



(b) Point set having frieze group F_1^1. The symmetries are given by the translation and a reflection at a line parallel to the direction of the translation.



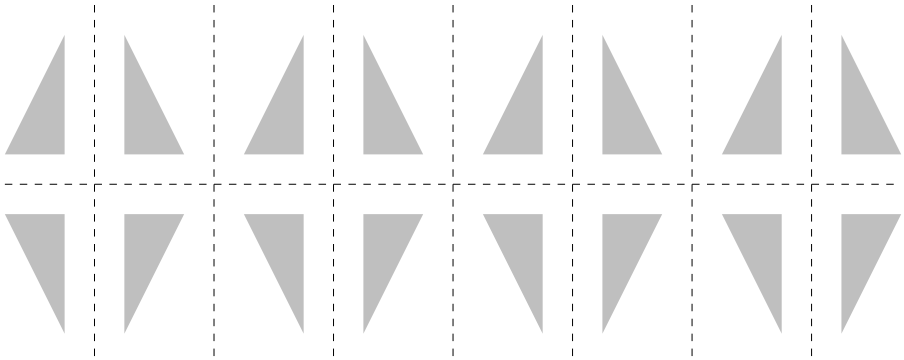
(c) Point set having frieze group F_1^2. The symmetries are given by the translation and reflections at lines perpendicular to the direction of the translation.



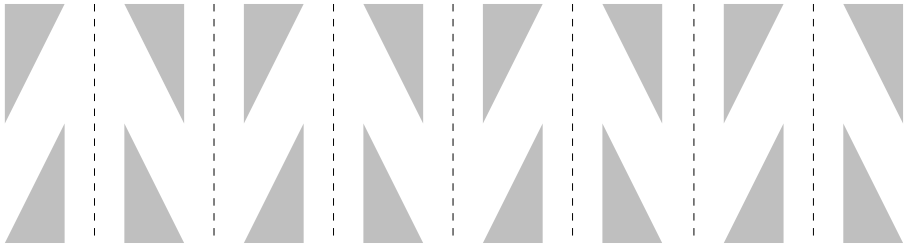
(d) Point set having frieze group F_1^3. The symmetries are given by a glide-reflection.

Figure 1.2.2: Examples of the seven frieze groups.

(e) Point set having frieze group F_2. The symmetries are given by the translation and a halfturn.



(f) Point set having frieze group F_2^1. The symmetries are given by the translation, a halfturn and the reflection at a line parallel and at lines perpendicular to the direction of the translation .



(g) Point set having frieze group F_2^2. The symmetries are a translation, a halfturn and reflections at lines perpendicular to the direction of the translation.

Figure 1.2.2: Examples of the seven frieze groups (cont.).

1.2.2 Wallpaper Groups

A pattern having a wallpaper group as its symmetry group has the property that the dilatation of its basic pattern is two-dimensional. There are seventeen different wallpaper groups. We will not consider wallpaper groups in detail in this thesis. We refer to the book of Martin [27] for further reading on wallpaper groups.

Chapter 2

Symmetry Detection

In this chapter we will develop algorithms which analyze real world images and detect the symmetry group of the objects depicted. We will present algorithms detecting finite as well as infinite symmetry groups.

In Section 2.2 we will explain how methods used in image processing such as the cross-correlation and the discrete Fourier transform can be used in order to detect the symmetry group of an image.

In Section 2.3 we will assume the image to be preprocessed in a way that the object depicted in the image is represented by polygonal curves. To this representation we will apply probabilistic algorithms in order to detect the symmetry group of the object. We will apply techniques known from pattern matching, which is used to determine a transformation from a set of given transformations which maps best the one object to the other (see [2]). One approach is to randomly choose a certain number (dependent on the degree of freedom of the transformation) of sample points from each object and compute the transformation mapping the one point set to the other. A vote for this transformation is generated in transformation space. After taking sufficiently many votes, clusters arise in transformation space. The cluster with the largest number of votes is the one representing the transformation which best maps the one object to the other. In Section 2.3 we will explain how this approach can be used in order to detect the symmetry group of an object represented by a polygonal curve.

Both approaches can be used even if the input images contain noise.

2.1 Related Work

The problem of determining the symmetry group of an object is well studied. There are many algorithms solving this problem using different approaches. Some of them, especially earlier works ([41],[6]), assume the input points to be in exact symmetric positions.

Wolter et al. [41] present algorithms for exact symmetry detection in two and three dimensions: In order to determine the finite symmetry group of a point set in $\mathbb{R}^2$, they first sort the points in the set by their polar coordinates around the centroid. Points with the same distance to the centroid are grouped together in a subset. For each subset the points are represented by the difference between the polar angle coordinate of the point and its successor. For each subset the result is a string $A =< s_0, \ldots, s_{n-1} >$ consisting of numbers representing the differences between the angles. The number of rotational symmetries of a subset is the number of positions where pattern A occurs in text AA minus one. Computing this number for all subsets separately by using a string matching algorithm and taking the greatest common divisor of all these numbers supplies the number of rotational symmetries of the whole point set. In order to decide if the point set is symmetric with respect to a reflection line, a similar algorithm is used.

In contrast, Atallah [6] concentrates on detecting symmetry axes. He also encodes the point set in the same way as Wolter et al. [41] by a string and asks whether the letters of the string can be shifted so that the string is a palindrome. The number of possible palindromes generated by the shift operation then is the number of symmetry axes.

The drawback of the algorithms of Wolter et al. [41] and Atallah [6] is that the input has to be precise, since a point set where only one point is shifted slightly will not be considered symmetric anymore. Thus for realistic inputs where the exact symmetry of a real-world object might get lost by the process of transferring it into a point set they are not viable.

Zabrodsky et al. [42] approach this problem by not considering symmetry to be a binary feature of an object (it either has a certain symmetry or not), but as a continuous feature. They introduce a "Symmetry Distance" (SD) which states the amount of a certain symmetry the object has. They consider a space Ω of shapes of a given dimension where each shape $P \in \Omega$ is represented by a sequence of n points $\{P_i\}_{i=0}^n$ and define the metric $d : \Omega \times \Omega \mapsto \mathbb{R}$, $d(P, Q) = \sum_{i=0}^n ||P_i - Q_i||^2$. The "Symmetry Transform" (ST) of a shape P for a certain symmetry group is the symmetric shape $Q \in \Omega$ closest to P w.r.t d and the "Symmetry Distance" (SD) of P then is the distance to its ST. They use the ST and SD of a shape in order to reconstruct the symmetry of an object which might have got lost during the process of transferring its representation into a point set.
We will investigate a similar problem in Chapter 3: For a point set P we want to construct a symmetric point set Q which is a good approximation of P. The distance

measure we will consider is $\min_{1 \leq i \leq n} \{||P_i - Q_i||\}$.

Zabrodsky et al. [42] assume the symmetry group of symmetry transform of the input shape to be given. However, if one wants to use this algorithm in order to restore the symmetry of an originally symmetric object which was lost during the process of representing it by a point set, one needs algorithms which detect the symmetry of a point set even if its representation is distorted by noise. This setting is a realistic one. Thus there are many algorithms detecting the symmetry group of an object even if its representation contains noise.

One way to represent the object is by an image, e.g. as a .jpg or .png file. Those files mostly originate either from a picture taken by a camera or are created by a drawing program. In both cases it is likely that the original symmetries got lost during this process, either because of the perspective of the camera or because of the impreciseness generated while handling the drawing utility.

Detecting the symmetry of an object represented by an image belongs to the area of image processing and thus it is not astonishing that approaches used in image processing algorithms are also used for symmetry detection algorithms.

Matsuyama et al. [29] use the discrete Fourier transform in order to analyze the texture given in a gray-level image. They detect the placement rule of the texture elements. They assume that the texture results from the repetition of a basic pattern and determine the translation vectors.

Liu and Collins [26] use the autocorrelation in order to extract meaningful building blocks from a repeated pattern. Furthermore, they use the theory of frieze and wallpaper groups in order to find a small set of candidate motives that exhibit local symmetry. They apply their algorithms in [25] in order to detect the wallpaper group of a pattern even if it is only a part of the input image or if it is given by an imperfect real-world image.

The Fourier transform and its relatives are also used to detect the finite symmetry group of an object depicted in a real-world image. Keller and Shkolnisky [23] use the angular correlation which is computed by using the pseudo-polar Fourier transform in order to detect the rotational and reflectional symmetries of two-dimensional objects.

Johansson et al. [20] also determine rotational as well as reflectional symmetry of a gray-level image. They do not apply the normalized convolution directly to the image, but use a preprocessing step in order to extract local orientation, and afterwards they apply normalized convolution to the orientation image where they use rotational symmetry filters as basis functions.

Shen et al. [39] use generalized complex moments in order to detect the number of reflection axes as well as the number of rotational symmetries of an object represented by a gray-level image.

Derrode and Ghorbel [12] use the analytical Fourier-Mellin transform in order to compute motion parameters between two gray-level images depicting objects with the same shape but distinct scale and orientation. Using the same image twice as input for the algorithm, they are able to detect symmetries of the object depicted in

the image.

Another approach is to represent the object by a point set, a set of edges and vertices or a set of parametrized curves.

Mitra et al. [33] assume a two- or three-dimensional object to be represented by a set of points. For randomly chosen pairs of points they compute the reflection line or rotation mapping these points together and vote for this affine transformation in transformation space. Cluster of votes arise in transformation space and the cluster with the largest number of votes corresponds to the symmetries of the object. Using this method, the authors present algorithms for detecting symmetries of the object. The presented algorithms are used to find local symmetries, meaning parts of the object where the reflected or rotated image also is part of the input. The algorithms are not used for detecting the symmetry group of the object.

In [34] this method is used to reconstruct the symmetry of an object or to "symmetrize" an non-symmetric object.

Imiya and Fermin [16] use a randomized voting method for testing planarity of a two-dimensional object in $\mathbb{R}^3$ and detecting the motion of this object in three-dimensional space. In [17] they use the presented algorithms in order to detect reflectional and rotational symmetries of a polygon.

2.2 Symmetry Detection Using the Fourier Transform

In this section we will present algorithms that analyze realistic input images given, for example as .jpg files and detect the symmetry group of the pattern depicted in these images. We will present algorithms detecting infinite symmetry groups as well as finite symmetry groups. We will explain the algorithms for frieze patterns extensively and explain later on how these methods can also be used in order to analyze patterns having finite symmetry group.

A frieze pattern is given by an infinite number of repetitions of a basic pattern. The symmetry group of the frieze pattern is defined by the symmetries of this basic pattern, as explained in Section 1.2.1. The first task in order to detect the symmetry group of a frieze pattern is therefore to determine the basic pattern. Since in real world applications the input cannot be infinite, the input frieze pattern is a finite part of an infinite frieze. In this section we assume that the input frieze pattern is given by an integer number of repetitions of the basic pattern. We also assume the basic pattern to be repeated significantly often.

One way to extract the basic pattern of the frieze is to analyze how often it is repeated in the frieze pattern. In the following sections we will explain how the number of repetitions of the basic pattern in the frieze pattern can be determined by using either string matching algorithms or correlation or the discrete Fourier transform. We will state and analyze algorithms for each method and also discuss the advantages and disadvantages of the different approaches.

2.2.1 Analyzing Frieze Patterns Using String Matching

In string matching theory the task is to find all occurrences of a pattern P in a text T. Here P and T are supposed to be words over some finite alphabet Σ.

We will use string matching algorithms in order to determine the number of repetitions of the basic pattern in the input frieze pattern. The approach is similar to the one of Wolter et al. [41]. The difference is that Wolter et al. [41] determine the finite symmetry group of a point set, whereas we analyze objects with infinite symmetry group which are given by an $m \times n$ gray-level pixel image. Moreover, Wolter et al. [41] assume the input to be symmetric, whereas we allow the input to contain noise.

Formulation as String Matching Problem

Let us assume that the input image is a gray-level image with values between 0 (black) and 255 (white). Let furthermore the image consist of m rows $R_1, \ldots, R_m$ and n columns. Thus we can define the alphabet to be $\Sigma = \{0, \ldots, 255\} \subset \mathbb{N}$. We can also assume each row R_i of the input frieze pattern to be a text $T_i \in \Sigma^*$ on its own where $1 \leq i \leq m$.

Suppose the frieze pattern is defined by a basic pattern B of length l. We denote the m rows of B by $R_1^B, \ldots, R_m^B$. Again, we interpret each row $R_i^B, 1 \leq i \leq m$ as a

word $T_i^B \in \Sigma^*$. Since we assume the basic pattern to be repeated an integer number of times in the input frieze pattern, T_i^B can be found at $\frac{n}{l}$ positions in T_i, where $1 \leq i \leq m$. More precisely, the positions where the word T_i^B is placed in T_i are given by the set $\{lj + 1 | 0 \leq j \leq \frac{n}{l} - 1\}$. We could determine the length l_i of the basic pattern T_i^B by applying a string matching algorithm to the text $\tilde{T}_i = T_i T_i$ and the pattern T_i. The second position where T_i occurs in $\tilde{T}_i$ gives the length of the text T_i^B, see Wolter et al. [41]. The number of repetitions of the basic pattern in the frieze pattern is then given by $k_i = \frac{n}{l_i}$.

The number of repetitions of the global basic patterns for all rows is given by the greatest common divisor k of the values $k_1, \ldots, k_m$. The length of the global basic pattern is given by $l = \frac{n}{k}$.

There are several algorithms solving the string matching problem. A famous one is the one by Knuth et al. [24], which runs in time $O(|T| + |P|)$, where $|T|$ is the length of the text and $|P|$ is the length of the pattern.
A basic idea of the string matching algorithm of Knuth, Morris, and Pratt is to compute the prefix function of the pattern. The prefix function reflects how well the pattern matches itself. Thus we can directly use the prefix function in order to determine the length of the basic pattern of the frieze pattern.
The prefix function of a pattern P is defined as follows (see Cormen et al. [11]):

$$\pi[q] = \max\{k | k < q \text{ and } P_k \sqsupset P_q\}, 1 \leq q \leq |P|$$

where P_l denotes the prefix of P of length l and $P_k \sqsupset P_q$ means that P_k is a suffix of P_q.

Lemma 2.2.1. *Let T, $|T| = n$ be a text representing a frieze pattern and let π be the prefix function of T. Let B, $|B| = l$ be the text representing a basic pattern of the frieze pattern and let l be minimal with this property. The length of B is then given by $l = n - \pi[n]$.*

Proof. T represents a frieze pattern whose basic pattern is represented by B. Thus $T = B^k$, where $k = \frac{n}{l}$. Therefore $T_{n-l} = B^{k-1} \sqsupset T_n = B^k$ and thus $\pi[n] \geq n - l \Leftrightarrow l \geq n - \pi[n]$.
For $l' = n - \pi[n]$ assume $l > l'$. Then $T_{\pi[n]} \sqsupset T_n$ and thus $T[i] = T[i + l']$, for all $1 \leq i \leq n - l'$ and thus T would contain a basic pattern B' of length l', which is a contradiction to the assumption that B represents the basic pattern of the frieze represented by T with minimal length. Thus $l \leq n - \pi[n]$.
This proves that $l = n - \pi[n]$. □

Computing the prefix function for a pattern of length n takes $O(n)$ time, thus we can determine the length of the basic pattern of the frieze pattern represented by T, $|T| = n$ in time $O(n)$.
Asymptotically the running time remains the same even if we apply the whole string matching algorithm of Knuth, Morris, and Pratt since in our case the pattern has length n and the text has length $2n$ and thus the algorithm runs in time $O(n)$.

The greatest common divisor of two numbers c_1 and c_2 can be computed in time $O(\log n)$ by using the algorithm of Euclid where $\log n$ is the number of bits needed to represent c_1 and c_2. This algorithm has a total running time $O(m(n + \log n)) = O(mn)$ since the string matching algorithm is applied to m rows and the greatest common divisor of m numbers, all smaller than n, is computed.

The algorithm explained above can only be applied if the input does not contain noise and if the basic pattern is repeated an integer number of times. In that case, the basic pattern can be found as exact copies in the frieze pattern and the algorithm based on the string matching approach gives the correct answer. This situation is given in artificially constructed input patterns but not in realistic inputs. We need to adapt the algorithm in order to allow noise and distortion to be contained in the input.

One possibility would be to add a tolerance threshold δ to the string matching algorithm that allows two gray-level values to be considered as equal if they differ by at most δ. This approach also fails for realistic inputs. Consider e.g. an input image with a white background where at one position in the background there is a black pixel. At this position the comparing method of the string matching algorithm would not return the two pixels to be equal unless $\delta = 255$. The problem of the string matching approach is that the question if the pattern is found in the text is decided locally at each pixel and not globally over the whole text.

A more convenient possibility is to compute all differences between the positions compared and to compute their average. Thus for each position $j, 1 \leq j \leq n$ where we apply the pattern T to the text $\tilde{T} = TT$ we compute the value

$$c_j = \frac{1}{n} \sum_{i=1}^{n} |T[i] - \tilde{T}[i + j]|.$$

For position j we say that the pattern T is found in the text $\tilde{T}$ if the value c_j does not exceed the threshold δ. This is more convenient for the problem we are considering, but the running time increases to $O(n^2)$ since we need $O(n)$ time for each index j and $1 \leq j \leq n$.

Solving the String Matching Problem Using Correlation

The string matching problem can also be solved by using correlation. Correlation is used in image processing algorithms, e.g. in order to compare two images or to locate objects in an image. Various applications can be found in textbooks concerning image processing and pattern recognition. A survey of image registration techniques is given by Brown [8].

Definition 2.2.2 (Correlation)**.** Let two vectors $a = (a_0, \ldots, a_{n-1})$ and $b = (b_0, \ldots, b_{n-1})$ over a field K be given. The correlation of a and b is the vector $c = (c_0, \ldots, c_{2n-2})$, where

$$c_k = \begin{cases} \sum_{i=0}^{k} a_i b_{n-1-k+i} & k < n \\ \sum_{i=k-n+1}^{n-1} a_i b_{n-1-k+i} & k \geq n \end{cases}$$

In order to use correlation for algorithms solving the string matching problem one needs to assure that the correlation value c_k is high if the pattern P is found at position k in the text T and low otherwise.

This can be achieved by using normalized cross-correlation.

Let $P \subset \Sigma^*$ and $T \subset \Sigma^*$ consist of m and n letters, respectively. Let $P[i]$ and $T[i]$ denote the ith letter of P and T, respectively. The vector $(d_1, \ldots, d_{n-m})$ is given by

$$d_j = \frac{\sum_{i=0}^{m-1} P[i]T[i+j]}{\sqrt{\sum_{i=0}^{m-1}(T[i+j])^2}}, 1 \leq j \leq n-m$$

If the pattern is found at position j, the value d_j is the maximum of all values $d_0, \ldots, d_{n-m}$, for a proof see Rosenfeld and Kak [36]. The normalized cross-correlation is invariant under multiplication with a constant. This means, that it is possible, that d_j is the maximum but that the pattern P is not exactly found at position j in text T. In this case however, P can be scaled by a constant and this modified pattern is found at position j at T. This is convenient in the case where P and T are representing gray-level images. The fact that P is allowed to be scaled by a constant factor only means, that the image represented by P is found in the image represented by T, even if it is lighter or darker as T.

We can write the fraction defining d_j as follows:

$$d_j = \frac{d_j^{\text{num}}}{\sqrt{d_j^{\text{den}}}},$$

where

$$d_j^{\text{num}} = \sum_{i=0}^{m-1} P[i]T[i+j] \quad \text{(I)}$$

$$d_j^{\text{den}} = \sum_{i=0}^{m-1} (T[i+j])^2 \quad \text{(II)}$$

Defining

$$c_k^{\text{num}} = \begin{cases} \sum_{i=0}^{k} a_i^{\text{num}} b_{n-k+i-1}^{\text{num}} & k < n \\ \sum_{i=k-n+1}^{n-1} a_i^{\text{num}} b_{n-k+i-1}^{\text{num}} & k \geq n \end{cases}$$

and

$$c_k^{\text{den}} = \begin{cases} \sum_{i=0}^{k} a_i^{\text{den}} b_{n-k+i-1}^{\text{den}} & k < n \\ \sum_{i=k-n+1}^{n-1} a_i^{\text{den}} b_{n-k+i-1}^{\text{den}} & k \geq n \end{cases}, \text{ where}$$

$$a^{\text{num}} = (P[0], \ldots, P[m-1], \underbrace{0, \ldots, 0}_{n}), b^{\text{num}} = (\underbrace{0, \ldots, 0}_{m}, T[0], \ldots, T[n-1]),$$

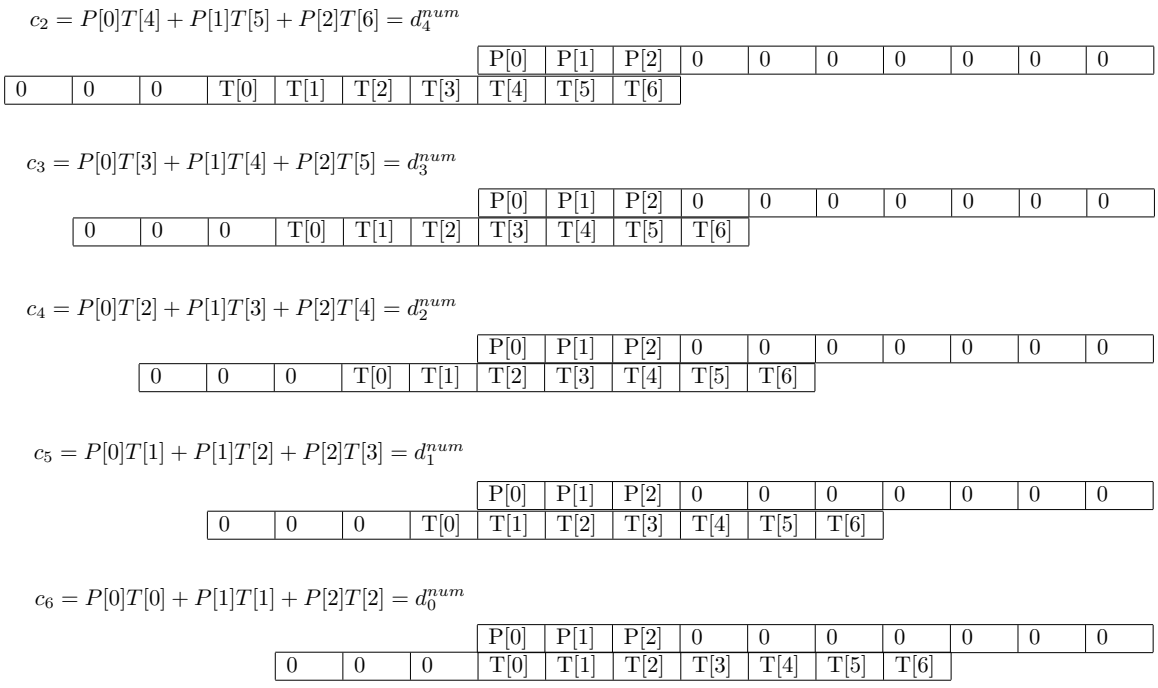


Figure 2.2.1: Illustration of the computation of d_j^{num} using correlation.

$$a^{\text{den}} = (1, \ldots, 1, \underbrace{0, \ldots, 0}_{n}) \text{ and } b^{\text{den}} = (\underbrace{0, \ldots, 0}_{m}, (T[0])^2, \ldots, (T[n-1])^2)$$

gives $d_j^{\text{num}} = c_{n-1-j}^{\text{num}}$ and $d_j^{\text{den}} = c_{n-1-j}^{\text{den}}$, f.a. $j \in \{0, \ldots, n-m\}$.

As in the previous section we consider each row separately and apply the string matching algorithm which uses correlation to the text $\tilde{T}_i = T_i T_i$ and the pattern T_i.

In the case where the pattern $T[0] \ldots T[n]$ is found in the text $\tilde{T}[0] \ldots \tilde{T}[2n]$ at position j the value of d_j is $d_j = \sqrt{\sum_{t=1}^{n} (\tilde{T}[t+j])^2}$ by definition of d. Since we assume the input image to contain noise, we allow the value d_j to differ from $\sqrt{\sum_{t=1}^{n} (\tilde{T}[t+j])^2}$ by some small value δ and still consider the pattern T to be found in $\tilde{T}$ at position j.

For the text in each row we are able to compute the number $k_i, 1 \leq i \leq m$, which denotes the number of appearances of T_i in $\tilde{T}_i$, by computing the correlation of the two vectors as explained above. Again, the length l of the basic pattern is given by $l = \frac{n}{k}$, where k is the greatest common divisor of $k_1, \ldots, k_m$.

The running time of the algorithm using correlation is given by the time needed to compute m times the correlation of two vectors of size n and the running time needed to compute the greatest common divisor of the numbers $k_1, \ldots, k_m$. As stated in Section 2.2.1 computing the greatest common divisor of the numbers $k_1, \ldots, k_m$ takes $O(m \log n)$ time. Computing the correlation of two vectors of size n takes time $O(n \log n)$ by using the fast Fourier transform. This is due to the fact that the cor-

relation correlates to the convolution which correlates to the problem of multiplying two polynomials. The product of two polynomials of degree n can be computed in time $O(n \log n)$ by using the fast Fourier transform (see [11]). Determining the number of repetitions of the basic pattern in a frieze pattern using correlation can be done in $O(mn \log n)$ time.

2.2.2 The DFT and Infinite Symmetry Groups

The Fourier transform (FT) is an operation which transforms a function of time into a function of frequency.
The Fourier transform has applications in signal processing where a signal is decomposed into its frequencies. The discrete Fourier transform (DFT) is the discrete equivalent to the Fourier transform. In signal processing it is used whenever the signal is given by a number of discrete values rather than a continuous function.
The DFT is also used in many algebraic algorithms as e.g. algorithms computing the product of two large numbers or two polynomials. A survey of the DFT and its application in algebraic algorithms is given by Mateer [28].
In image processing, the image can be considered to be a discrete signal. The DFT can be used in order to analyze the structure of this image. The DFT decomposes the image into its "frequencies". Thus the result of the DFT applied to an image provides information about the periodic structure of the image (see [29], [23]).

In this section we will explain how the discrete Fourier transform can be used in order to extract the basic pattern of a frieze pattern.

Prior to explaining the algorithm we take a closer look at the DFT:

Definition 2.2.3. Let a vector $a = (a_0, \ldots, a_{n-1}) \in \mathbb{C}^n$ be given. The DFT of a is the vector $y = (y_0, \ldots, y_{n-1}) \in \mathbb{C}^n$, where

$$y_k = \sum_{j=0}^{n-1} a_j e^{\frac{i2\pi jk}{n}}$$

The fast Fourier transform (FFT) computes the DFT of a vector of length n in time $O(n \log n)$ (see [10]). Similar to the correlation of two vectors, the complex number y_k indicates whether the two vectors $a = (a_0, \ldots, a_{n-1})$ and $e_k = (e^0, e^{\frac{i2\pi k}{n}}, \ldots, e^{\frac{i2\pi(n-1)k}{n}})$ are similar or not. The magnitude of y_k is large if both vectors are equal and small if they differ at many positions. The values of the vector $e_k, 0 \leq k \leq n-1$ are derived from the periodic function e^{ixk}, $x \in \mathbb{R}$, with period $\frac{2\pi}{k}$ which is sampled at n equidistant angles $\frac{2\pi j}{n}, 1 \leq j \leq n-1$. Thus in the case where the magnitude of the value y_k is large, the input vector is similar to a function with period $\frac{2\pi}{k}$ and therefore we assume to find a vector of length $\frac{n}{k}$ in the input vector which is repeated k times.

As in the previous sections, we consider the input frieze pattern to be represented by a gray-level pixel image. For the i^{th} row R_i we consider the values to be given by the vector T_i. As described above, we expect to find a basic pattern B_i of length

$\frac{n}{k}$ repeated k times in T_i if the magnitude of the value y_k is large or, to be more precise, the largest of all values $y_0, \ldots, y_{n-1}$.
The basic idea of the algorithm can be described as follows:

1. Compute the discrete Fourier transform by using FFT of the vector T_i.
2. Find the entry y_k in the vector $y = (y_0, \ldots, y_{n-1})$ with the largest magnitude.
3. The length of the basic pattern is $\frac{n}{k}$ and it is repeated k times.
4. The basic pattern is given by the first $\frac{n}{k}$ columns.

In order to determine the number of repetitions of the basic pattern for the whole input frieze we first compute the number of repetitions of the basic pattern for each row separately and then compute the greatest common divisor of all these numbers.

Algorithm 2.2.1 states the procedure for determining the number of repetitions of the basic pattern in an input frieze pattern by directly using the discrete Fourier transform.

Algorithm 2.2.1 Computing the number of repetitions of the basic pattern of a frieze pattern using the DFT.

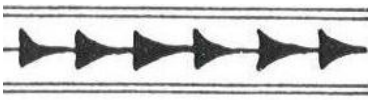
ExtractPatternFourier(I)
// The two-dimensional array representing the image is denoted by I. $I[i]$ represents the i^{th} row of the image.
for $i = 1$ to I.NumberOfRows() **do**
  $C[i]$= FFT($I[i]$);
  $M[i]$ = Magnitudes($C[i]$);
end for
//For each row $M[i], 1 \leq i \leq m$ compute the index where $M[i][j], 1 \leq j \leq n$ is maximal. The array MVR *contains that index for each row.*
for $i = 1$ to I.NumberOfRows() **do**
  maximum = $M[i][1]$;
  for $j = 2$ to I.NumberOfColumns() **do**
    if maximum $< M[i][j]$ **then**
      maximum := $M[i][j]$;
      MVR$[i] = j$;
    end if
  end for
end for
//Compute the greatest common divisor of the values in MVR.
gcd = GCD(MVR);
return gcd;

Computing the discrete Fourier transform of m vectors of size n takes $O(mn \log n)$ time. The greatest common divisor of m numbers smaller than n can be computed in time $O(m \log n)$. The overall running time of Algorithm 2.2.1 is $O(mn \log n)$.

Remark 2.2.4. For real world inputs, it depends on the amount of noise contained in the input if the algorithm extracts the correct basic pattern. The Fourier transform is a quite fault-tolerant transform, but still there are input images where we need to be careful by interpreting the results or even where the approach fails.

In the remaining part of this section we will give an example in order to visualize how Algorithm 2.2.1 works.

Example 2.2.5. We consider the finite part of a frieze pattern given in Figure 2.2.2 (a).



(a) Example of a frieze. The basic pattern is repeated six times.

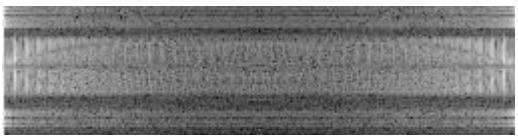
(b) The extracted basic pattern.

Figure 2.2.2: The input pattern and the extracted basic pattern.

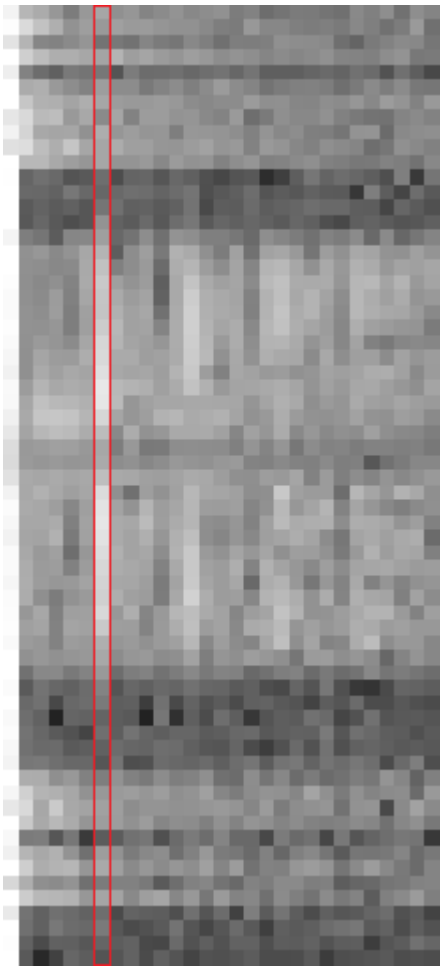
The basic pattern is an arrow (see Figure 2.2.2 (b)) which is repeated six times. The magnitudes of the value computed by the fast Fourier transform applied to each row separately is depicted in Figure 2.2.3 (a). Figure 2.2.3(b) shows a close-up of the magnitudes of Figure 2.2.3(a). The gray-values represent the magnitudes of the complex numbers computed by the fast Fourier transform. The larger the magnitude the lighter is the pixel. Dominating frequencies are the ones that are represented by light gray pixels or even white ones. When taking a close look at the close-up depicted in Figure 2.2.3(b) we see that for many rows the largest number of light pixels can be found in the sixth column (highlighted in Figure 2.2.3). Thus we can conclude that for these rows the length of the basic pattern is $\frac{n}{6}$.

Applying the procedure described above we would compute the length of the base pattern for each row separately and afterwards compute the greatest common divisor of all these numbers. The input frieze in this example has rows which completely consist of white pixels. For these rows, the highest value of the Fourier transform is achieved at y_0 since the input function is constant. In fact for each row the magnitude of the value y_0 is high. This is due to the fact that the input is finite.

Getting back to the example, we see that for all rows where the index of the value with highest magnitude is greater than two, this index is equal to six and therefore the basic pattern is indeed repeated six times in the frieze. Thus we can extract the basic pattern from the frieze pattern by taking the first $\frac{n}{6}$ pixels of each row.



(a) The image of all the magnitudes.



(b) A close-up and a part of the image of the magnitudes.

Figure 2.2.3: The image of the magnitudes of the discrete Fourier transform applied to the rows of the image given in Figure 2.2.2.

2.2.3 The DFT and Finite Symmetry Groups

In this section we will explain how to detect the finite symmetry group of a given input figure by using the procedures explained above. Basically, we will explain how to transform the input figure which depicts a shape with symmetry group C_k or D_k into a frieze pattern. Additionally, we will explain how the algorithms detecting the length of the basic pattern can be used in order to find the symmetry group of the input shape. Again we assume the input shape not to be perfectly symmetric but possibly to be distorted by noise. In the previous sections we assumed the input to be given in a reasonable way. We will assume the same for the inputs we consider in this section. First, we assume the input to be given as a gray-level pixel image and since we search for cyclic or dihedral symmetry groups, we assume the input to be quadratic. The number of rows and the number of columns is denoted by n. Second, we assume the rotation center to be in the middle of the picture, more precisely around the pixel in row $\frac{n}{2}$ and column $\frac{n}{2}$.

Cyclic Symmetry Groups

Assume we are given an input shape which has cyclic symmetry group C_k. Rotating the shape around its rotation center c by a multiple of $\frac{2\pi}{k}$ yields the same shape again.
We can sweep the image by a ray through c that is rotated around c. Let S_i be the square pixels which are intersected by the ray after the rotation by the angle $i\alpha$. The value of α will be determined later. Since the shape has symmetry group C_k, $S_i = S_{i+\frac{2\pi}{k}}$.
We can interpret each S_i as a column of a gray-level pixel image. The result is a frieze pattern where the basic pattern is repeated k times.
We can solve the problem of finding the rotational symmetry group of an input shape by applying algorithms determining the number of repetitions of a basic pattern in a frieze pattern.

Example 2.2.6. Consider the example depicted in Figure 2.2.4. The scanning algorithm explained above results in the frieze pattern depicted in Figure 2.2.5.

It might be the case that the input shape is the composition of several shapes with different cyclic symmetry groups. Considering the corresponding frieze pattern, the length of the global basic pattern is given by the greatest common divisor of the lengths of the basic patterns defined by the rows of the frieze pattern. Computing the number of repetitions of the basic pattern supplies the cyclic symmetry group number of the input shape.

Detecting the number of rotational symmetries of an input shape can be done by Algorithm 2.2.2. We assume the image to be given as a gray-level image of size $n \times n$ and the gray values to be stored in an $n \times n$ array I.

Figure 2.2.4: Example of an input shape for finite symmetry detection.

Figure 2.2.5: Frieze pattern generated from the input shape in Figure 2.2.4.

Dihedral Symmetry Groups

After generating a frieze pattern from the input shape and determining the number of repetitions of the basic pattern, we know the number of rotational symmetries in the symmetry group. In order to decide if the symmetry group additionally contains reflections, we need to investigate the basic pattern. In the case where we started the cyclic scan at an axis of reflection, the constructed basic pattern itself contains an axis of reflection. Since we neither know if the shape contains an axis of reflection nor where it is located, it is not likely that we start the scan accidentally at an axis of reflection.

Example 2.2.7. An example of a D_7-symmetric figure is given in Figure 2.2.6. Applying the scanning algorithm explained above gives the frieze pattern depicted in Figure 2.2.7.

The basic pattern of the generated frieze pattern is depicted in Figure 2.2.8.

Observe that the basic pattern depicted in Figure 2.2.8 does not contain an axis of symmetry although the original input shape depicted in Figure 2.2.6 does contain an axis of symmetry. Nevertheless we can use the basic pattern in order to decide if the original input shape contains mirror symmetry. We only need to consider the basic pattern repeated twice (see Figure 2.2.9 (a)) and the reverse of the basic pattern (see Figure 2.2.9 (b)). We then apply a string matching algorithm in order to determine if the reverse of the basic pattern is found in the doubled basic pattern.

Algorithm 2.2.2 Detecting the number of rotational symmetries of an input shape.

RotationalSymmetries(I)

Generate a frieze pattern from I by using a rotational scan.

Use Algorithm 2.2.1 in order to determine the number of repetitions k of the basic pattern.

return k;

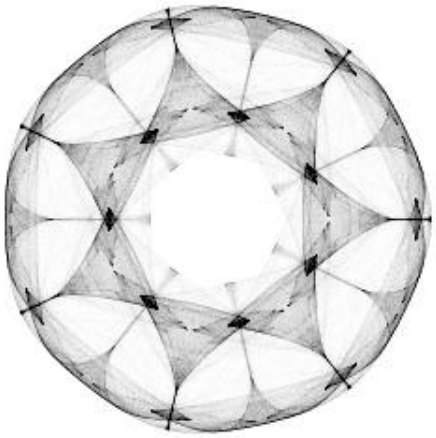


Figure 2.2.6: Example of an input shape for finite symmetry detection.

In this case the original input shape is symmetric with respect to a reflection line and thus has symmetry group D_k, in the other case it has symmetry group C_k.

Algorithm 2.2.3 detects the finite symmetry group of a given input shape. It first applies Algorithm 2.2.2 to the input in order to determine the number k of rotational symmetries in the symmetry group. Afterwards it uses the above described algorithm in order to decide whether the input shape has cyclic symmetry group C_k or dihedral symmetry group D_k.

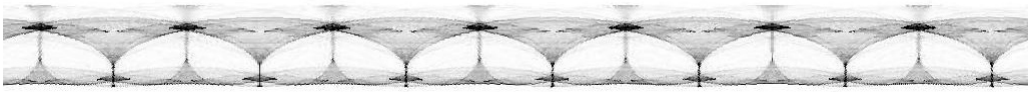
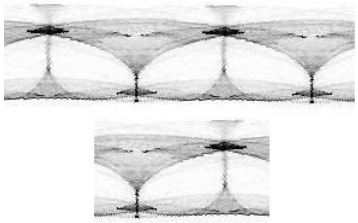


Figure 2.2.7: Frieze pattern generated from the input shape in Figure 2.2.6.

Figure 2.2.8: Basic pattern of the frieze in Figure 2.2.7.

(a) The doubled basic pattern.

(b) The reversed basic pattern



(c) The reversed basic pattern is found in the doubled basic pattern.

Figure 2.2.9: Deciding whether the original input shape is symmetric with respect to a reflection line.

Algorithm 2.2.3 Detecting the finite symmetry group of an input shape.

FiniteSymmetryGroup(I)
Generate a frieze pattern from I by using a rotational scan.
Use Algorithm 2.2.1 in order to determine the number of repetitions k of the basic pattern.
Extract basic pattern B;
Generate reversed basic pattern B^R;
Generate doubled basic pattern B^2;
if B^R is found in B^2 **then**
  return D_k;
else
  return C_k;
end if

2.3 The Probabilistic Approach

In this section we will explain how probabilistic methods can be used in order to determine the symmetry group of a given shape.
The algorithms we will develop in this section work on a finite number of simple objects such as straight lines or smooth curves. Thus we need the input shape to be represented in such a way.
Real world objects are often given by image files. Following the arguments given in the last section, the original symmetry of the object might have got lost during the process of representing it by an image file. In order to transform an image file representing the original input shape into a representation by a set of simple objects, image processing techniques such as edge detection are needed. It is most likely that even if the original symmetry of the object was maintained while representing it by an image file, it will get lost due to the image processing techniques used to represent it as a finite set of simple objects.
The algorithms we will develop in this section are robust against noise contained in the input. Thus they can be used to determine the symmetry group of a real world input even if the original symmetry of the shape got lost during the process of representing it as a finite set of simple objects.
The probabilistic approach we will use is also used and well investigated in shape matching theory. In shape matching theory one considers the question how alike two given shapes are under a set of transformations like similarity maps, shear transformations or general affine maps. Shape matching algorithms using the probabilistic approach were described and analyzed by Scharf [38]. The basic idea is to randomly choose a certain number (depending on the degree of freedom of the transformation) of points from each set and compute the corresponding transformation between these two points. In transformation space, a vote for this particular transformation is generated. After choosing sufficiently many random points, clusters of votes will arise in transformation space. The cluster with the largest number of votes will provide with high probability a transformation which is close to the best transformation between the two input sets.

We will use the probabilistic approach in order to detect the finite symmetry group of a given input shape. For the symmetry detection problem, only one input shape is given and the question is how alike it is to itself under rotation and reflection. We will therefore choose randomly a pair of points from the point set and compute the rotation around a given rotation center or reflection mapping one point to the other. As in the algorithms used in shape matching theory, clusters will arise in transformation space. Each cluster represents a rotation or reflection, respectively, that maps a large number of point pairs together. An input shape having symmetry group C_k, even if it is not perfectly symmetric but contains noise, will generate clusters in transformation space with a large number of votes at the rotations with rotation angles $\frac{2\pi}{k}, \frac{4\pi}{k}, \ldots, \frac{(k-1)2\pi}{k}$. The problem of finding the symmetry group can therefore be solved by counting the number of clusters with a large number of votes.

This number is the order of the symmetry group. In order to decide if the shape has a cyclic or a dihedral symmetry group, one needs to check the clusters representing the reflections computed between randomly chosen pairs of points. We will give all the details in the following sections.
Since clustering is a crucial point in probabilistic algorithms, we will start by introducing and explaining the notion of clustering algorithms. In Section 2.3.1, we will state the probabilistic algorithm for detecting rotational symmetry. Afterwards, we will show that the order of the symmetry group can indeed be computed out of the clusters with a large number of votes and how this can be done. In Section 2.3.2 we will give all the details of the algorithm detecting reflection lines in a point set using the probabilistic approach.

Clustering

After generating sufficiently many votes, one needs to find clusters arising in transformation space. The topic of clustering a set of objects is well studied and is used whenever a large set of data is supposed to be segmented into sets of similar objects. Applications can be found, for example, in computer graphics or pattern matching. One part of computer graphics is the segmentation of images into parts with the same property, for example with the same color. In pattern matching, one wants to group patterns with similar shapes into a cluster in order to distinguish them from those patterns that have a different appearance. A good and detailed discussion of clustering algorithms and their applications is given by Jain et al. [19].
As there are many different applications for clustering, there are many different approaches for clustering algorithms. The main difference between the different approaches is whether the algorithm works top-down or bottom-up. In the first case, the whole set is considered as one cluster at the beginning and is divided into smaller sets in each iteration step. This is done until a predefined number of clusters is obtained or a given threshold is passed. In contrast, clustering algorithms working bottom-up first assume each single object to be a cluster by itself and join two or more clusters to one larger cluster in each step. When using a bottom-up approach, the desired number of clusters or a threshold must be given in order to terminate the process before all clusters are united in one large cluster containing all objects.
The objects are modeled as points in a d-dimensional vector space where d is the number of features of the object. In order to compare two objects and compute their alikeness, a distance measure on the vector space needs to be given.
We will explain the concrete implementation of the clustering algorithms used for the rotational and reflectional symmetry detection in Sections 2.3.1 and 2.3.2, respectively.

2.3.1 Rotational Symmetry

In this section, we will explain how the cyclic symmetry group of a given input shape can be determined by using the probabilistic approach. As indicated above, the basic idea is to choose two points randomly and compute the rotation mapping these two points together. Unfortunately, there is not only one well-defined rotation with this property, but infinitely many. This is the case since for two given points p_1 and p_2 there is one rotation mapping p_1 to p_2 for each rotation angle between 0 and 2π. The rotation centers differ for each angle, but all rotation centers lie on the perpendicular bisector of p_1 and p_2, see Figure 2.3.1.

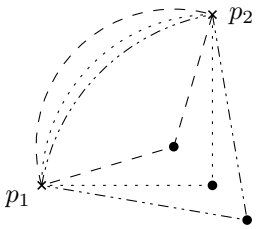


Figure 2.3.1: For two points p_1 and p_2, there exists a rotation mapping p_1 to p_2 for each angle in $[0, 2\pi]$. All rotation centers lie on the perpendicular bisector of $\overline{p_1, p_2}$.

Since we need one well-defined rotation for each pair of points, we will fix a rotation center and compute the unique rotation mapping p_1 to p_2 with respect to the chosen rotation center. We will use the center of mass of the input shape as the rotation center, since it can be computed easily from the input set and it is a good approximation of the real rotation center of the input shape, as we will see in Chapter 3.

The general idea of the algorithm detecting the cyclic group of a given point set by using probabilistic methods is stated in Algorithm 2.3.1.

Algorithm 2.3.1 Probabilistic algorithm for cyclic symmetry group detection.

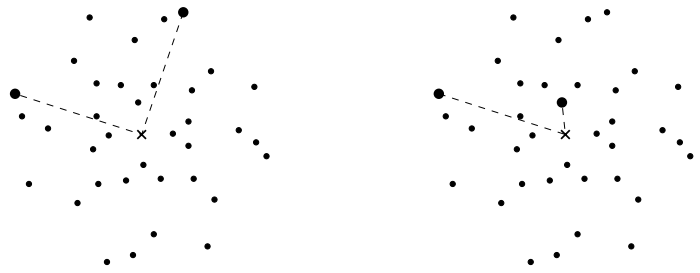
DetectRotationalSymmetryGroup(Shape $\mathcal{S}$)
  Compute the center of mass of S.
  repeat
    Choose two points p_i and p_j from the input shape $\mathcal{S}$.
    Compute the rotation $\alpha_{i,j}$ mapping p_i to p_j w.r.t. the center of mass.
    Vote for α_{ij} in transformation space.
  until enough samples taken
  Cluster the votes in transformation space.
  Compute symmetry group from the clusters with large number of votes
  return Computed symmetry group;

We will give the details of the algorithm in the following part of this section. For some statements in the algorithm we will need to distinguish between different

variants of the problem in order to give the concrete answer. For example, the way of randomly choosing two points p_i and p_j depends on the way how the input shape is given. Also, the number of random sample pairs of points needed depends on the given shape. If the shape is given by a finite set of points, it may not even be necessary to choose the pairs randomly, but one could simply compute all rotations defined by all possible pairs of points. We will investigate the possible variants of the problem and how to adopt the algorithm later on in this section. We will start by explaining how to randomly choose pairs of points, depending on the way the shape is represented.

Input Shape is Represented by a Finite Set of Points

First, we will consider the case where the input shape is given by a finite set of points. As stated in Algorithm 2.3.1 we use the center of mass as the rotation center. For two arbitrary points, however, the center of mass will not be the center of a rotation mapping the two points together since the distances between the two points and the center of mass may not be the same (see Figure 2.3.2).



(a) A pair of points allowing the computation of a rotation.

(b) No appropriate rotation can be computed.

Figure 2.3.2: A rotation can only be computed if the distances to the rotation center are nearly the same.

For this reason we alter the process of randomly choosing two points. For a given value δ, we partition the point set into subsets $\mathcal{S}_i = \{p \in \mathcal{S} | (i-1)\delta \leq d(p,c) \leq i\delta\}$ where c is the center of mass and $d(p,c)$ denotes the Euclidean distance between p and c. We divide the point set into subsets where the points of $\mathcal{S}_i$ lie in an annulus with width δ and distance $(i-1)\delta$ to the center of mass (see Figure 2.3.3).

We want to choose valid pairs of points, where both points lie in the same subset and we want the distribution to be uniform. Let m be the number of subsets $\mathcal{S}_1, \ldots, \mathcal{S}_m$. Consider the interval $[0, \sum_{j=1}^{m} |\mathcal{S}_j|^2]$. By choosing $a \in [0, \sum_{j=1}^{m} |\mathcal{S}_j|^2]$ randomly, we get a subset $\mathcal{S}_i$, where $\sum_{j=1}^{i-1} |\mathcal{S}_j|^2 < a \leq \sum_{j=1}^{i} |\mathcal{S}_j|^2$. Thus the probability of choosing a certain subset $\mathcal{S}_i$ is $\frac{|\mathcal{S}_i|^2}{\sum_{j=1}^{m} |S_j|^2}$. from the chosen subset $\mathcal{S}_i$ we choose two points uniformly at random. The property for such a pair to be chosen

Figure 2.3.3: Point set and its partition using annuli of width δ

is therefore $\frac{|\mathcal{S}_i|^2}{\sum_{j=1}^{m}|S_j|^2}\frac{1}{|\mathcal{S}_i|^2} = \frac{1}{\sum_{j=1}^{m}|S_j|^2}$.
Thus by applying the above described procedure, we choose a valid pair of points uniformly at random from the $\sum_{j=1}^{m}|S_j|^2$ valid pairs.

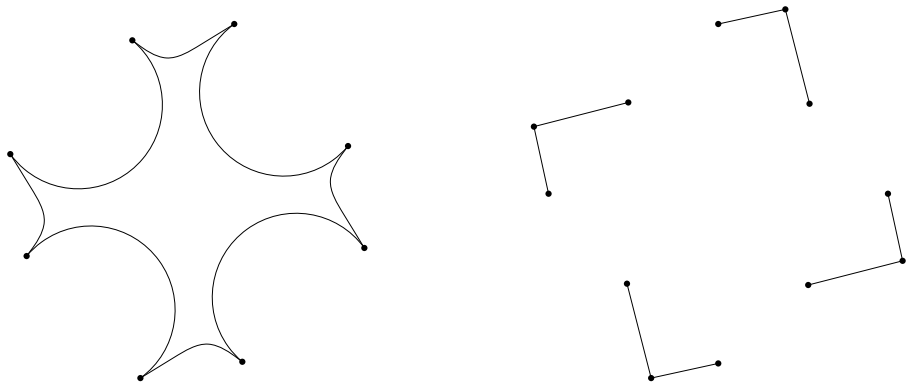
The number of subsets can be computed by δ and the maximum distance between the center of mass and a point in P. Algorithm 2.3.2 states how to choose two random points, so that their distance to the center of mass differs by at most a small value δ.

Algorithm 2.3.2 Choosing two random points of a shape represented by a finite set of points.

ChooseRandomPoints$(P = \{p_1, \ldots, p_n\}, \delta)$
  $c = \frac{1}{n}\sum_{i=1}^{n} p_i$;
  $d = \max\{d(c,p)|p \in P\}$;
  $m = \frac{d}{\delta}$;
  for $i = 1$ to m **do**
    $S_i = \{p \in P|(i-1)\delta \leq d(c,p) \leq i\delta\}$;
  end for
  Choose a random value $a \in [0, \ldots, \sum_{j=0}^{m}|S_j|^2]$;
  $i = 1$;
  sum $= |S_1|^2$
  while $a >$ sum **do**
    $i = i + 1$;
    sum $=$ sum $+|S_i|^2$;
  end while
  Choose p_1 and p_2 randomly from S_i;
  return (p_1, p_2);

Input Shape is Represented by a Finite Set of Smooth Curves

In contrast to the case where the input shape is given by a finite set of points, we will now consider inputs consisting of infinitely many points. We assume that the shape is represented by a parametrization of a curve C. The curve is allowed to be split up into a finite set $\mathcal{C}$ of n segments of smooth curves such as straight lines, circular arcs etc. The shape does not need to be connected. In the case where the shape itself or parts of it enclose a region, we only consider points on the boundary of that region. For examples see Figure 2.3.4.



(a) A C_4-symmetric shape consisting of circular arcs and splines.

(b) A C_4-symmetric shape consisting of straight lines.

Figure 2.3.4: Two examples of shapes given by a finite number of segments of smooth curves.

In the experiments we will do later on in order to evaluate the presented algorithms, the input shape will be represented by a set $\mathcal{L} = \{l_1, \ldots, l_n\}$ of n straight line segments. Let the straight line segment l_i be defined by $l_i = \overline{p_i q_i}$, thus $l_i = \{\lambda p_i + (1-\lambda)q_i\} | 0 \leq \lambda \leq 1\}$. The set $\mathcal{L}$ of straight line segments can be represented by a parameterized curve $C : [0,1] \mapsto \mathbb{R}^2$ where for each straight line segment $l_i \in \mathcal{L}, 1 \leq i \leq n$ we define a parameterized curve $C_i : [a_{i-1}, a_i] \mapsto \mathbb{R}^2$ where $a_i = \frac{\sum_{j=1}^{i} |l_j|}{\sum_{j=1}^{n} |l_j|}$ and

$$C_i(\lambda) = \left(\frac{\lambda - a_{i-1}}{a_i - a_{i-1}}\right) p_i + \left(1 - \frac{\lambda - a_{i-1}}{a_i - a_{i-1}}\right) q_i, \lambda \in [a_{i-1}, a_i].$$

The parameterized curve $C : [0,1] \mapsto \mathbb{R}^2$ is then defined by:

$$C(\lambda) = C_i(\lambda), \text{ for } \lambda \in [a_{i-1}, a_i].$$

By using this parameterization we can choose a point uniformly at random from the set of straight line segments by choosing a value $\lambda \in [0,1]$ uniformly at random. The randomly chosen point is then given by $C(\lambda)$.

In the case where the input shape was represented by a finite number of points, we used the center of mass as rotation center. For an input shape represented by an infinite number of points, we will compute the center of mass of all the points on the curve C.

Observation 2.3.1. *Let a set $\mathcal{L} = \{l_1, \ldots, l_n\}$ of straight line segments be given, where $l_i = \overline{p_i q_i}, 1 \leq i \leq n$. The center of mass of all points on the straight line segments in the set $\mathcal{L}$ is given by:*

$$\frac{\sum_{i=1}^{n} \frac{p_i+q_i}{2} d\left(p_i, q_i\right)}{\sum_{i=1}^{n} d\left(p_i, q_i\right)}.$$

For a shape represented by a finite set of straight line segments, the process of randomly choosing two points is given in Algorithm 2.3.3, where p_i and q_i denote the endpoints of the line segment l_i, f.a. $1 \leq i \leq n$.

Algorithm 2.3.3 Choosing two random points of a shape represented by a finite set of straight line segments.

ChooseRandomPoints$(L = \{l_1, \ldots, l_n\})$
$d = \sum_{i=1}^{n} d(p_i, q_i)$;
$c = \frac{1}{d} \sum_{i=1}^{n} \frac{(p_i+q_i)d(p_i,q_i)}{2}$;
Choose a value $\lambda \in [0, 1]$ uniformly at random.
//Determine the line segment which is given by the value of λ. *Find index* i *so that* $a_{i-1} \leq \lambda < a_i$
$i = 1$;
//preLength represents $a_{i-1} \cdot totalLength$
preLength $= 0$;
//length represents $a_i \cdot totalLength$
length $= |l_1|$;
totalLength $= \sum_{j=1}^{n} |l_j|$;
while $\lambda > \frac{\text{length}}{\text{totalLength}}$ **do**
  $i = i + 1$;
  preLength $=$ length;
  length $=$ length $+|l_i|$;
end while
//$\lambda \in [a_{i-1}, a_i]$ *needs to be true.*
$\lambda = \lambda - preLength$;
//Compute the coordinates of the randomly chosen point p_1.
$p_1 = \left(\frac{\lambda - \text{preLength}}{\text{length} - \text{preLength}}\right) p_i + \left(1 - \frac{\lambda - \text{preLength}}{\text{length} - \text{preLength}}\right) q_i$;
Compute the disk $D = (r, c)$ where $r = d(p_1, c)$;
//Compute P *as the set of all points of the intersection of the input shape with the disk* D.
$P = \{\lambda p_i + (1 - \lambda) q_i | \lambda \in [0, 1] \wedge i \in \{1, \ldots, n\} \wedge d(p, c) = r\}$;
Choose random point $p_2 \in P$;
return (p_1, p_2);

Computing the Rotation Angle

The two procedures explained above assure the two randomly chosen points to have approximately the same distance to the center of mass. In fact, in the case of a shape represented by an infinite number of points, the two points do have exactly the same distance to the rotation center c, which is assured by Algorithm 2.3.3. The angle of the rotation mapping these two points together with respect to a given rotation center can be computed as follows:

Observation 2.3.2. *For two given points p_1 and p_2 and a rotation center c, the angle α between the two line segments $\overline{p_1c}$ and $\overline{p_2c}$ is given by*

$$\alpha = \arccos\left(\frac{d(p_1,c)^2 + d(p_2,c)^2 - d(p_1,p_2)^2}{2d(p_1,c)d(p_2,c)}\right)$$

Applying the above arguments, we derive a more concrete version of Algorithm 2.3.1 which is stated in Algorithm 2.3.4. We will no longer distinguish between the different representations of the shape, but use a procedure corresponding to the representation in order to randomly choose two points of the shape with approximately the same distance to the center of mass.

Algorithm 2.3.4 Probabilistic algorithm for rotational symmetry group detection

DetectRotationalSymmetryGroup(Shape $\mathcal{S}$, δ)
  c :=CenterOfMass($\mathcal{S}$);
  repeat
    Choose two random points p_1, p_2 using Algorithm 2.3.2 or Algorithm 2.3.3;
    $\alpha := \arccos\left(\frac{d(p_1,c)+d(p_2,c)-d(p_1,p_2)}{2d(p_1,c)d(p_2,c)}\right)$;
    Vote for α in transformation space;
  until enough samples taken.
  Cluster the votes in transformation space;
  Compute symmetry group from the clusters with large number of votes;
  return Computed symmetry group;

Transformation Space and Clustering

In the case of rotational symmetry group detection, the transformation space is the one dimensional interval $[0, 2\pi]$ since we vote for angles between 0 and 2π. Prior to stating the clustering algorithm we use for clustering the rotation angles, we will explain why it is possible to compute the cyclic symmetry group from the cluster. What needs to be proven is that the angles representing the cluster with the largest number of votes are indeed the rotation angles of a cyclic symmetry group. So, for an input set having symmetry group C_k, we need to show that the angles $0, \frac{2\pi}{k}, \frac{4\pi}{k}, \ldots, \frac{(k-1)2\pi}{k}$ represent the k clusters with largest number of votes.
Suppose for a moment that the input point set $\mathcal{S}$ yields perfect symmetry. Then $\mathcal{S}$ can be decomposed into a set of regular k-gons all having the same rotation center. The angle computed by two points of the same regular k-gon is in the set $\{\frac{2\pi i}{k} | 0 \leq i \leq k-1\}$. There may also be angles computed that are not in this set, since it is possible for two regular k-gons to have the same distance to the center of mass. In that case two points not belonging to the same k-gon might be chosen. The angle between the corresponding line segments might be any angle in the interval $[0, 2\pi]$. However, after taking sufficiently many sample points, the majority of votes will be at the angles $\{\frac{2\pi i}{k} | 0 \leq i \leq k-1\}$. We will prove this by the following lemma where we assume the point set to be finite.

Lemma 2.3.3. *Let P, $|P| = n = mk$, $m, k \in \mathbb{N}$ be a finite point set with symmetry group C_k. Consider the rotation angles between all pairs of points having the same distance to the center of mass. Then each angle of the set $\{\frac{2\pi i}{k} | 0 \leq i \leq k-1\}$ arises n times. Each angle in $[0, 2\pi] \setminus \{\frac{2\pi i}{k} | 0 \leq i \leq k-1\}$ arises at most $(n-k)$ times.*

Proof. We start by proving the angle $\frac{2\pi i}{k}$ to be computed n times. Since P has symmetry group C_k, rotating an arbitrary point $p \in P$ around the center of mass c by $\frac{2\pi i}{k}$ results in another point in P. Since P contains n points, this angle is computed n times. This is true for each angle $\frac{2\pi i}{k}$, $0 \leq i \leq k-1$.
Since we assume P to be C_k-symmetric, there exists a partition of P into m regular k-gons $Q_1, \ldots, Q_m$ which all have the same rotation center c, which is the center of mass of P. Let $\mathcal{Q} = \{Q_1, \ldots, Q_m\}$ denote the set of these regular k-gons. The only case where two points are considered that do not belong to the same regular k-gon arises when the vertices of at least two regular k-gons, Q_i and Q_j have the same distance to the center of mass.
Suppose $q_{(i,1)} = \rho_c^\alpha(q_{(j,1)})$ for two vertices $q_{(i,1)}$ and $q_{(j,1)}$ of Q_i and Q_j respectively. We can assume that $\alpha < \frac{2\pi}{k}$, otherwise we can renumber the vertices of Q_i and Q_j. In the case where $q_{(i,1)} = \rho_c^\alpha(q_{(j,1)})$ this is also true for all the other vertices of the two regular k-gons, so $q_{(i,t)} = \rho_c^\alpha(q_{(j,t)})$ f.a. $1 \leq t \leq k$. Thus the angle α occurs k times. The same is true for the angles $\{\alpha + \frac{2\pi i}{k} | 0 \leq i \leq k-1\}$.
In the case where there are l regular k-gons $Q_1, \ldots, Q_l \in \mathcal{Q}$ so that there exist l regular k-gons $Q'_1, \ldots, Q'_l \in \mathcal{Q}$ so that $Q_i = \rho_c^\alpha(Q'_i)$ there are overall $l \cdot k$ votes for the angle α. Since $l \cdot k = n \Leftrightarrow l = \frac{n}{k}$, there are n votes for the angle α iff for all points

$q \in P$ there is a point $q' \in P$, so that $q = \rho_c^\alpha(q')$. In this case, P has symmetry group $\frac{2\pi}{\alpha} > k$ since $\alpha < \frac{2\pi}{k}$.
Thus $l \leq m-1$ and therefore α arises at most $lk \leq (m-1)k = \left(\frac{n}{k} - 1\right) k = (n-k)$ times. □

In the proof for the perfectly symmetric case, we divided the computed angles into two sets, namely $A_1 = \{\frac{i2\pi}{k} | 0 \leq i \leq k-1\}$ and $A_2 = [0, 2\pi] \setminus A_1$. When considering not perfectly symmetric point sets, the computed angles will not be in the sets A_1 or A_2, but in the union of the intervals $\bigcup_{\alpha \in A_1} ([\alpha - \varepsilon, \alpha + \varepsilon])$ or $\bigcup_{\alpha \in A_2} ([\alpha - \varepsilon, \alpha + \varepsilon])$ for some small ε. This is due to the fact that the input set may be perturbed. Therefore, two computed angles that would be the same in the perfect symmetric case differ in the case where noise is added by at most some value ε. The counting argument, however, as given in Lemma 2.3.3 stays the same.

In a setting where the input sets are perfectly symmetric, no clustering would be necessary. The next step after computing the rotation angles would be to compute the symmetry group from the angles with largest number of votes. For real world inputs, however, the computed angles are distributed around the angles belonging to the regular k-gon. The clustering algorithm in our approach is used to bundle those angles together that actually represent the same angle, but differ slightly due to the noise in the input data. However, for an input set containing a tenable amount of noise, we expect the votes representing the same rotation angle to lie close to that angle.

A convenient clustering algorithm to solve this problem is a bottom-up approach. We use a hierarchical clustering algorithm as described by Johnson [21]. In this approach, a distance measure is given for the objects to be clustered. In the first step, each element is seen as a cluster for itself. The distance between each pair of objects is computed. The two clusters with the minimum distance are joined to one cluster and all distances between this new cluster and all the remaining clusters are computed. This procedure is iterated until the minimum distance between two clusters exceeds a given threshold.
The transformation space we need to consider in the application of detecting rotational symmetry is the one dimensional interval $[0, 2\pi]$.
We start by sorting the votes representing the computed rotation angles in increasing order.

Before we start to cluster the votes representing the computed rotation angles, we can think of them as points on the line segment representing the interval $[0, 2\pi]$. For the clustering algorithm we always merge the two clusters with minimum distance, thus we only need to compute the distances between two consecutive clusters. The distance of two clusters is defined as the difference between the average value of the elements in the clusters. We first search for the minimum distance value between two clusters. If that value is smaller than a given threshold, we unite these two clusters. We only need to recompute the distance to the clusters to the left and to the right of the new cluster. We iterate this procedure until all distances between clusters are

greater than the given threshold.

We first give the notations used in the clustering algorithm and afterwards we state the algorithm itself:

Notation 2.3.4. *The i^{th} cluster is denoted by C_i and for each cluster we store the following values:*

Representative: *The representing angle is denoted by α_i.*

Number of elements: *The number of angles in the cluster is given by $|C_i|$.*

Distance: *The distance to the next cluster is given by d_i, where $d_i := \alpha_{i+1} - \alpha_i$.*

Note that we need to compute the distances between the representing angles in a circular fashion, since an angle of $2\pi - \varepsilon$ for a small ε is close to 0. We therefore compute the distance $d_n = 2\pi + \alpha_1 - \alpha_n$.

For a cluster C_i containing the angles $\alpha_{i_1}, \ldots, \alpha_{i_{|C_i|}}$ we use the arithmetic mean as the representing angle α_i. Therefore, we compute this angle as $\alpha_i = \frac{1}{|C_i|} \sum_{j=1}^{|C_i|} \alpha_{i_j}$. In the case where we merge two clusters C_i and C_j we can compute the new representative of the cluster $C_{i,j} = C_i \cup C_j$ by using the representatives and numbers of elements of C_i and C_j.

Observation 2.3.5. *Let two clusters C_i and C_j be given. After merging the two clusters, the new representative is given by*

$$\alpha_{i,j} = \frac{1}{|C_i| + |C_j|}(|C_i|\alpha_i + |C_j|\alpha_j)$$

and can be computed in constant time.

Merging two neighboring clusters C_i and C_{i+1}, results in a new cluster which we denote by C_i'. The two former clusters are deleted from the set of clusters and replaced by the new cluster C_i'. We only need to update the two distance values defined by the cluster C_{i-1} and C_{i+2}. This can be done by computing $d_{i-1}' = \alpha_i' - \alpha_{i-1}$ and $d_i' = \alpha_{i+1} - \alpha_i'$, where α_i' is the representative of the new cluster C_i' as computed in Observation 2.3.5.

Algorithm 2.3.5 Clustering algorithm

Cluster$(\{\alpha_1 \dots \alpha_n\})$
  //Initialize the clusters:
  for $i := 1$ to n **do**
    //Each angle itself forms a cluster.
    C_i. Representative $= \alpha_i$;
    //Compute the distance to the successive angle.
    $d_i = \alpha_{i+1} - \alpha_i$;
    C_i. Distance $= d_i$;
    C_i. NumberOfElements $= 1$;
    //Add the new generated cluster to the set of clusters.
    $\mathcal{C} := \mathcal{C} \cup \{C_i\}$
  end for
  //Find the index of the smallest distance value.
  $i =$ IndexMinDistance$(\mathcal{C})$;
  while C_i. Distance $\leq \delta$ **do**
    //Unite the two clusters.
    //Compute the representative of the new cluster as given in Observation 2.3.5.
    $\alpha_i := \frac{1}{d_i + d_{i+1}} (d_i \alpha_i + d_j \alpha_j)$
    //Update the distances to the new representative.
    $d_{i-1} := \alpha_i - \alpha_{i-1}$;
    $d_i := \alpha_{i+2} - \alpha_i$;
    C_i. Distance $= d_i$;
    C_i. Representative $= \alpha_i$;
    C_i. NumberOfElements $= C_i$. NumberOfElements $+ C_{i+1}$. NumberOfElements;
    //Delete the assimilated cluster C_{i+1} from the set of cluster $\mathcal{C}$.
    Delete$(\mathcal{C}, C_{i+1})$;
    $i =$ IndexMinDistance$(\mathcal{C})$;
  end while

We will now analyze the running time of the clustering algorithm given by Algorithm 2.3.5.
The clusters are organized in a doubly linked list and the distance values d_i, $1 \leq i \leq n$ are organized in a priority queue. Each cluster has a pointer to its distance value and each distance value in the priority queue has a pointer to its cluster. If, for example, a heap is used to implement the priority queue, one step of finding the minimum distance and merging two clusters can be done in $\Theta(\log n)$ time. Computing the new values α_i, d_i and d_{i-1} can be done in $\Theta(1)$ time as described in Algorithm 2.3.5. Updating the values d_i and d_{i-1} can be done in $\Theta(\log n)$ time by deleting the elements from the heap and inserting new ones with the new values.
The overall running time is dependent on the number of clusters that are merged together. Thus it is dependent on both the input and the value δ. An upper bound for

the clustering algorithm is $O(n \log n)$ since at most n clusters can be merged together until the input set forms one cluster for itself. Since we already need $\Omega(n \log n)$ time in order to sort the clusters and insert the distance values into the heap, the running time of the clustering algorithm given by Algorithm 2.3.5 is $\Theta(n \log n)$.

Remark 2.3.6. Until now, we left open how to choose the value of δ. The choice of δ is dependent on the input. If the input set is highly distorted, but the order of the cyclic group is small so that there is a large gap between two rotation angles, a large value of δ is necessary to cluster the different angles produced by the distorted input data. If the distortion of the set is only slight and the order of the cyclic group is large, we will lose rotation angles in the case where the value of δ is not small enough. Finding a good value for δ is a balance between smoothing the contained noise and distinguishing between the different rotation angles.
By testing this symmetry detection algorithm on real world input images of the MPEG7 database, δ to be between 5° and 10° turned out to be a good choice. Since we are computing with radians instead of degree, $\delta = \frac{2\pi}{72} \approx 0.087$ is used in our implementation of the algorithm.

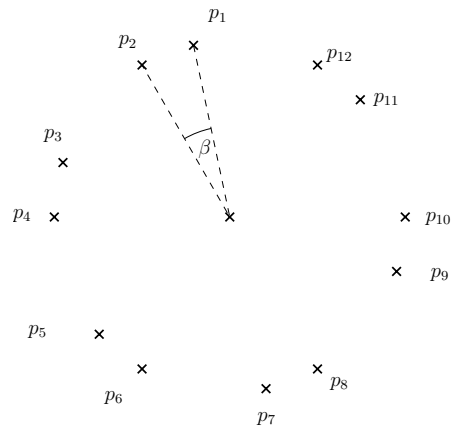
Computing the Order of the Symmetry Group

What remains to be done is to explain how the order of the cyclic symmetry group can be computed from the clustered rotation angle votes. We will start by explaining how the algorithm works for perfect symmetric input set and state later on how the presented algorithm needs to be modified in order to work for input sets containing noise.

Suppose we computed a sufficiently large number of rotation angles and voted in transformation space. After applying the clustering algorithm (see Algorithm 2.3.5), we obtain a list of rotation angles sorted in decreasing order by the number of votes. By Lemma 2.3.3 we expect the point set to have rotational symmetry with respect to the first k rotation angle values. Let us consider the case where the input set P is finite and we compute the rotation angles between each pair of points where the points have equal distance to the center of gravity. Let $A = \{\alpha_i | 1 \leq i \leq k\}$ be the set of angles, so that P is rotational symmetric with respect to each $\alpha \in A$. The number of votes is the same for each angle $\alpha \in A$ by Lemma 2.3.3. Furthermore, the number of votes for all the other angles that might be computed in the voting process is strictly smaller, which also is given by Lemma 2.3.3. We visualize this behavior using the following example:

Example 2.3.7. Suppose we are given the finite and C_k-symmetric point set $P = \{p_1, \ldots, p_{12}\}$ as depicted in Figure 2.3.5. This point set is the union of the vertex sets of two regular hexagons which have the same rotation center and whose smallest enclosing disks have the same radii. The first hexagon has the vertices $P_1 = \{p_{2i-1} | 1 \leq i \leq 6\}$ and the second hexagon has the vertices $P_2 = \{p_{2i} | 1 \leq i \leq 6\}$. Let P_1 be the counter clockwise rotated image of P_2 w.r.t the common rotation cen-

ter which is the center of mass of P and the rotation angle which we denote by β. Suppose we compute the rotation angles between all pairs of points in P. Since P_1 is a regular hexagon, for each angle $\frac{j2\pi}{6}$, $1 \leq j \leq 6$ there is a pair of points $\left(p_{2i-1}, p_{(2(i+j) \mod 12)-1}\right)$ for each $1 \leq i \leq 6$ that generates the rotation angle $\frac{j2\pi}{6}$. The same is true for the point set P_2. Thus each angle $\frac{j2\pi}{k}, 1 \leq j \leq 6$ arises 12 times. Since P_1 is the rotational image of P_2 with rotation center β, the angle $j\beta, 1 \leq j \leq 6$ is computed by the pair $(p_{2i-1}, p_{2(i+j-1) \mod 12}), 1 \leq i \leq 6$. Thus each angle $j\beta, 1 \leq j \leq 6$ arises only 6 times.



(a) The finite input set consists of two regular sixgons.

Angle	Votes
0	12
$\frac{2\pi}{6}$	12
$\frac{2\pi}{3}$	12
π	12
$\frac{4\pi}{3}$	12
$\frac{5\pi}{3}$	12
β	6
$\beta + \frac{2\pi}{6}$	6
$\beta + \frac{2\pi}{3}$	6
$\beta + \pi$	6
$\beta + \frac{4\pi}{3}$	6
$\beta + \frac{5\pi}{3}$	6
$\frac{2\pi}{6} - \beta$	6
$\frac{2\pi}{3} - \beta$	6
$\pi - \beta$	6
$\frac{4\pi}{3} - \beta$	6
$\frac{5\pi}{3} - \beta$	6
$2\pi - \beta$	6

(b) The corresponding number of votes per angle.

Figure 2.3.5: Illustration of 2.3.7

As we consider a perfectly symmetric input set in Example 2.3.7, the symmetry group of the input set can be determined by counting the number of angles with the maximal number of votes. In the more realistic case where the input set is disturbed by noise or in the case where we consider an infinite input set and need to compute the votes by choosing random pairs of points, this clear distribution of the votes is not guaranteed. As we will see in the experimental results, it is often the case that the first k clusters in the list of clusters ordered by the number of votes in decreasing order indeed represent the angles $0, \frac{2\pi}{k}, 2\frac{2\pi}{k}, \ldots, (k-1)\frac{2\pi}{k}$. In most of the examples, there is even a large gap between the number of votes created by these angles and the next highest number of votes. But this property is not given in all cases. Thus to determine the order of the symmetry group by simply counting the

number of clusters until the number of votes is decreased by a constant fraction does not always suffice. We therefore combine the two following strategies of determining the order of the symmetry group:

Strategy 1. The first strategy is to traverse the list of clusters sorted in decreasing number of votes until the number of votes of two successive clusters differs by a factor of at least 2. Suppose such a significant jump in the number of votes is detected after k clusters. We conjecture the symmetry group of the input set to be C_k and verify this by checking that the angles representing the k located clusters are exactly the ones in the set $\{i\frac{2\pi}{k} \pm \delta | 0 \leq i \leq k-1\}$, where we allow an appropriate error δ for each angle.

It might be the case that Strategy 1 fails since the gap between the clusters defining the order of the symmetry group and the other clusters is not large enough. In this case we use the following strategy in order to detect the symmetry group of the input set.

Strategy 2. Let α be the representing angle of the cluster with the largest number of votes. In the case where this angle is zero, we consider the representing angle of the cluster with the second largest number of votes. Since α is supposed to be a rotation angle of a cyclic symmetry group C_k for some $k \in \mathbb{N}$ which is to be determined, $\alpha = i\frac{2\pi}{k}$ for some $i \in \{1, \ldots, k-1\}$. Since $2\pi = k\frac{2\pi}{k}$, $\frac{2\pi}{k}$ is the greatest common divisor of both 2π and α if i and k are coprime. For two values $x, y \in \mathbb{R}$ we call $z \in \mathbb{R}$ the greatest common devisor of x and y, iff $x = az$ and $y = bz$, where $a, b \in \mathbb{N}$ and z is maximal with this property. The well-known Euclidean algorithm for computing the greatest common divisor can also be used to compute the greatest common divisor of two real numbers as defined above. The modulo operation used in the Euclidean algorithm is extended to the real numbers by defining

$$x \bmod y := x - \left\lfloor \frac{x}{y} \right\rfloor y$$

Let us assume for a moment, that $\alpha = i\frac{2\pi}{k}$ and that i and k are coprime.
By computing $\beta = \gcd(2\pi, \alpha)$ we get the angle of the rotation which generates the group. Therefore the order of the symmetry group is given by $\frac{2\pi}{\beta}$.
The angles we use in order to compute the order of the symmetry group are computed by the clustering algorithm described above. These angles will not exactly be a multiple of $\frac{2\pi}{k}$ due to the facts that we assume the input not to be exact and that the value representing the cluster is the arithmetic mean of all values in the cluster. Let $\tilde{\alpha}$ be the representing angle of the cluster with largest number of votes. We have to deal with the problem that $\tilde{\alpha}$ is used to determine the order of the symmetry group but is not equal to the exact value α of the angle of a rotation in the symmetry group. Suppose $\tilde{\alpha} = \alpha + \delta$ where $\delta \in \mathbb{R}$ is the value of the error made.
We will now explain how to adapt the Euclidean algorithm for computing the greatest common divisor of 2π and α even if the second parameter is not the exact value

of α but is the value of $\tilde{\alpha}$. In order to do so we take a closer look at the Euclidean algorithm for computing the greatest common divisor of two numbers:
For two numbers a_0 and a_1 the Euclidean algorithm produces a sequence of values $a_0, a_1, a_2, \ldots, a_n$, where $a_i = a_{i-2} \bmod a_{i-1}$ and $a_n = 0$. The greatest common divisor of a_0 and a_1 is then a_{n-1}.
Consider the sequence produced by $a_0 = 2\pi$ and $a_1 = \alpha$. Assuming that $a_n = 0$, the order of the symmetry group is given by $k = \frac{a_0}{a_{n-1}}$ by the above arguments.
Using the extended Euclidean algorithm we can not only compute the greatest common divisor of a_0 and a_1, but also two values $s_i, t_i \in \mathbb{Z}$, so that $a_i = s_i a_0 + t_i a_1$, and $|t_i| \leq \frac{a_0}{\text{ggt}(a_0,a_1)}$ $1 \leq i \leq n$.

Lemma 2.3.8. *Let $\tilde{\alpha} = \alpha + \delta$, where $\alpha = \frac{j2\pi}{k}$, $1 \leq j \leq k-1$ and $|\delta| < \frac{2\pi}{k^2}$ and j and k are coprime. Let $\tilde{a}_0, \tilde{a}_1, \ldots, \tilde{a}_m$ be the sequence produced by $\tilde{a}_0 = 2\pi$ and $\tilde{a}_1 = \tilde{\alpha}$. Then*

1. *$\tilde{a}_i = \frac{j_i 2\pi}{k} + c_i\delta$, $j_i \in \mathbb{N}, c_i \in \mathbb{Z}$*
2. *There exists $m \in \mathbb{N}$, so that $j_m = 0$ and $j_i > 0$, for all $1 < i < m$.*
3. *$|c_m| = k$, where m is the index defined in 2.*

Proof. 1. We use induction in order to proof the claim.
Basis:
$\tilde{a}_0 = 2\pi = k\frac{2\pi}{k} + 0\delta$, so $j_0 = k$ and $c_0 = 0$ and
$\tilde{a}_1 = \frac{j2\pi}{k} + \delta$, so $j_1 = j$ and $c_1 = 1$.
Thus the claim holds for $i \in \{0, 1\}$.
Inductive Step:

$$\begin{aligned}
\tilde{a}_i &= \tilde{a}_{i-2} \mod \tilde{a}_{i-1} \\
&= \tilde{a}_{i-2} - \left\lfloor \frac{\tilde{a}_{i-2}}{\tilde{a}_{i-1}} \right\rfloor \tilde{a}_{i-1} \\
&= 2\pi\frac{j_{i-2}}{k} + \delta c_{i-2} - \underbrace{\left\lfloor \frac{2\pi\frac{j_{i-2}}{k} + \delta c_{i-2}}{2\pi\frac{j_{i-1}}{k} + \delta c_{i-1}} \right\rfloor}_{d_i \in \mathbb{N}} \left(2\pi\frac{j_{i-1}}{k} + \delta c_{i-1}\right) \\
&= \frac{2\pi}{k}\underbrace{(j_{i-2} - d_i j_{i-1})}_{j_i \in \mathbb{N}} + \delta\underbrace{(c_{i-2} - d_i c_{i-1})}_{c_j \in \mathbb{Z}}
\end{aligned}$$

2. Note that $d_i \geq 1$, for all $2 \leq i \leq m$. Therefore $j_i < j_{i-2}$ and $j_m = 0$ for some $m \in \mathbb{N}$.

3. Let $m \in \mathbb{N}$ be the index where $j_m = 0$ and $j_i > 0$, for all $0 \leq i \leq m$. Then $\tilde{a}_m = \delta c_m$ and $\tilde{a}_m = s_m\tilde{a}_0 + t_m\tilde{a}_1$, where $s_m, t_m \in \mathbb{Z}$ can be computed by the extended Euclidean algorithm. It is always possible to find values s_m and t_m

with the properties $\tilde{a}_m = s_m\tilde{a}_0 + t_m\tilde{a}_1$, where $s_m, t_m \in \mathbb{Z}$ and $|t_m| \leq k$.

$$\begin{aligned}\delta c_m &= 2\pi s_m + t_m\left(\frac{j2\pi}{k} + \delta\right)\\ &= \frac{2\pi}{k}(ks_m + jt_m) + \delta t_m\end{aligned}$$

$|\delta t_m| \leq \frac{2\pi}{k}$, since $|t_m| \leq k$. Therefore, $(ks_m + jt_m) = j_m = 0$.
This implies

$$(s_m, t_m) = \begin{cases}(j, -k) & \delta < 0\\ (-j, k) & \delta \geq 0\end{cases}$$

and therefore $|c_m| = k$. □

In order to assure that the algorithm computes the correct order of the symmetry group, we need to bound the order of symmetry groups we consider. Assume that we only want to detect the symmetry groups $C_1, \ldots, C_{K-1}$, $K \in \mathbb{N}$. If $\delta < \frac{2\pi}{K^2}$, then $a_m < \frac{2\pi}{K}$. Thus testing $a_m < \frac{2\pi}{K}$ is a correct break condition for the algorithm computing the greatest common divisor of 2π and $\tilde{\alpha}$.

Lemma 2.3.9. *Let a number $K \in \mathbb{N}$ be given. For an angle $\tilde{\alpha}$ Algorithm 2.3.6 computes the smallest number $k < K$, so that the symmetry group C_k contains a rotation with rotation angle α, where $\tilde{\alpha} = \alpha + \delta$, $\delta \in \mathbb{R}$ and $0 \leq |\delta| \leq \frac{2\pi}{K^2}$, if such a k exists.*

Proof. The extended Euclidean algorithm computes the values $x, y, lastX, lastY$ (see Algorithm 2.3.6), so that $a = lastX2\pi + lastY\tilde{\alpha}$ and $b = x2\pi + y\tilde{\alpha}$ holds after each iteration step. Suppose $\alpha = \frac{j2\pi}{k}$, where j and k are coprime.
By Lemma 2.3.8 $k = |y|$, if $xk + yj = 0$.
The value $|y|$ is returned by Algorithm 2.3.6, if the condition $b > \frac{2\pi}{K}$ does not hold anymore. This is only possible if $xk + yj = 0$ since $(xk + yj) \geq 0$ and $y\delta \geq 0$ by Lemma 2.3.8 and therefore $b = (xk+yj)\frac{2\pi}{k} + y\delta \leq \frac{2\pi}{K}$ only holds if $(xk+yj) \leq 0$. □

Assume we consider an input set with symmetry group C_k. Let again $\tilde{\alpha} = \frac{j2\pi}{k} + \delta$ be the angle representing the cluster with the largest number of votes. If j and k are not coprime, then there exists a number $l \in \mathbb{N}$, so that $k = lk'$ and $j = lj'$ and k' and l' are coprime. Thus $\tilde{\alpha} = \frac{j'2\pi}{k'}$, where j' and k' are coprime and Algorithm 2.3.6 will return k' instead of k. It therefore computes the order of a subgroup of the symmetry group of the input set.

Since it is likely to happen that the computation of the order of the symmetry group returns the order of a subgroup of the symmetry group of the input figure, we repeat the procedure of verifying the symmetry group not only for the computed order of the symmetry group but also for a finite number of multiples. In the experimental results it turns out to be sufficient to check the computed order of the symmetry group and the multiples $2, 3$ and 4.

Algorithm 2.3.6 Algorithm for computing the order of the symmetry group given by an angle $\tilde{\alpha}$. A modified version of the extended Euclidean algorithm for computing the greatest common divisor of two real numbers is used.

OrderOfSymmetryGroup($\tilde{\alpha}$)
//Assume $\tilde{\alpha} = \frac{2\pi}{k} + \delta$, *where* $0 \leq |\delta| \leq \frac{2\pi}{K}$
$a = 2\pi$;
$b = \tilde{\alpha}$;
$x = 0$;
$y = 1$;
$lastX = 1$;
$lastY = 0$;
//For each iteration $a = lastX2\pi + lastY\tilde{\alpha}$ *and* $b = x2\pi + y\tilde{\alpha}$ *holds.*
while $b > \frac{2\pi}{K}$ **do**
$q = a \div b$;
$a' = b$;
$b = a \mod b$;
$a = a'$;
$x' = lastX - q \cdot x$;
$lastX = x$;
$x = x'$;
$c = a \mod b$;
$y' = lastY - q \cdot y$;
$lastY = y$;
$y = y'$;
end while
return $|y|$;

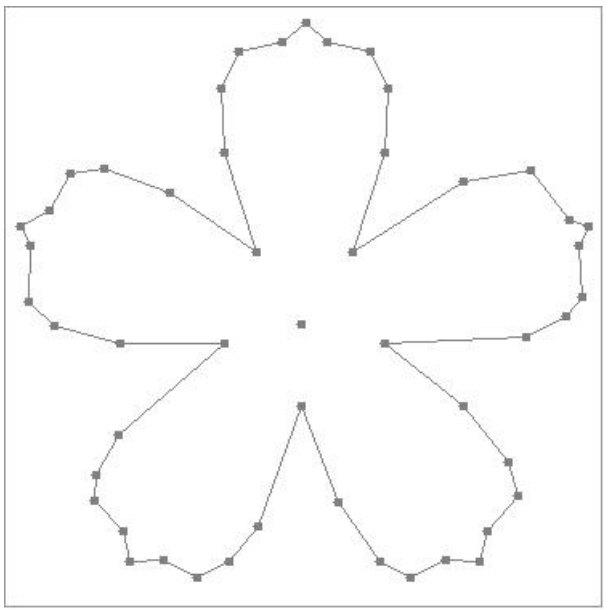
We will see in the next example how the algorithm explained above works on an input shape given as polygonal curve.

Example 2.3.10. The following polygonal curve is the input of the probabilistic symmetry detection algorithm for cyclic groups.

Choosing random pairs of points, and computing clusters of these votes gives the following array of rotation angles and number of votes:

Since the representation angle of the cluster with the largest number of votes is zero, Algorithm 2.3.6 takes the representation angle of the cluster with the second largest number of votes which in this case is $\alpha' = 73$. The algorithm proceeds as illustrated in Table 2.2:

When considering angle $\alpha' = 73$ we need to allow an error $\delta \leq 1$. Thus $K \leq 18$ needs to be true since $\delta < \frac{360}{K^2}$ must hold. When only considering symmetry groups with maximal order 18, the algorithm terminates after the third step since $b = 5 < \frac{360}{18} = 20$. It returns the value $|y| = 5$.



angle	number of votes	angle	number of votes
0	344	269	202
73	344	52	166
289	323	123	159
216	321	196	133
144	317	340	133
297	284	228	126
81	282		
154	260		
224	212		

Table 2.1: List of rotation angles and corresponding number of votes.

step	q	a	b	x	y	lastX	lastY
1		360	73	0	1	1	0
2	4	73	68	1	-4	0	1
3	1	68	5	-1	5	1	-4

Table 2.2: Illustration of the steps executed by OrderOfSymmetryGroup(73).

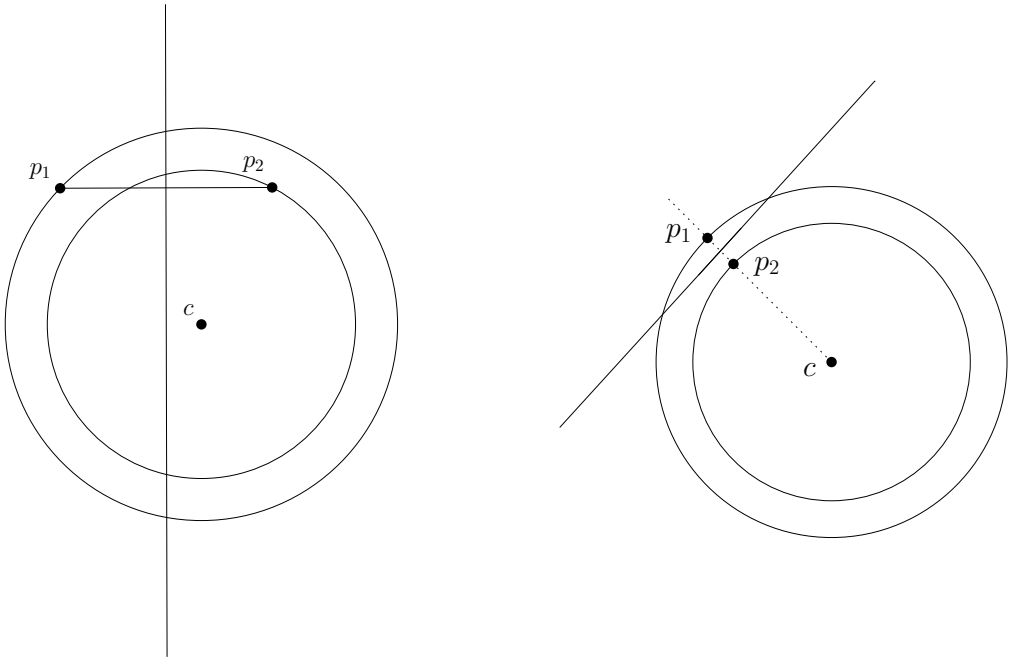
2.3.2 Reflectional Symmetry

As we saw in Section 1.1.2, an input figure has symmetry group D_k iff it has both rotational and reflectional symmetry. We explained how to detect rotational symmetry by using the probabilistic approach in the previous section. What remains to be done is to state how to detect reflectional symmetry with the probabilistic approach.

For two given points, the reflection line mapping the one point to the other is the well-defined perpendicular bisector of the line segment which connects the two points. Since the reflection lines of a D_k-symmetric figure all pass through the rotation center, we only need to consider those pairs of points whose reflection lines pass through the chosen rotation center, which is the center of mass in our case.

Observation 2.3.11. *Let two points p_1 and p_2 be given. The perpendicular bisector of the line $\overline{p_1p_2}$ passes through a point c, iff p_1 and p_2 lie on the boundary of a disk with center c.*

Since we consider input shapes that are not perfectly symmetric, but disturbed by noise, we need to make sure that the two randomly chosen points indeed have the same distance to the rotation center which is the center of mass in our algorithms. It depends on the way how we choose the two points whether the two randomly chosen points have equal distance to the rotation center. In the case where the input shape is represented by a finite set of parametrized curves, as for example straight line segments, and we compute the two random points using Algorithm 2.3.3, the two points have the same distance to the rotation center by construction. In the case where the input shape is represented by a finite set of points, it is unlikely to find two points with exactly the same distance to the rotation center. With the arguments given while we developed Algorithm 2.3.2 we consider two points whose distance to the rotation center only differ by some small value δ. Since the distances to the rotation center are allowed to differ by δ, the perpendicular bisector of the line segment $\overline{p_1p_2}$ will not pass through the rotation center in general. In the following we will explain what kind of problems may arise and how to deal with them.
Let again c denote the center of mass of the input point set P and let p_1 and p_2 be two points of P. Let $d(c, p_1) - d(c, p_2) = \varepsilon$. We need to define a line given by the two points p_1 and p_2 which passes through c and which tends to the reflection line defined by p_1 and p_2 as ε tends to zero. For $\varepsilon \neq 0$ the perpendicular bisector of the two points p_1 and p_2 is not a good choice since it does not pass through c in general, see 2.3.6(a). In the case where the three points p_1, p_2 and c lie on a line where c does not lie between p_1 and p_2, the angle between this line and the perpendicular bisector is $\frac{\pi}{2}$, see Figure 2.3.6(b). In the case where ε tends to 0, the two points p_1 and p_2 coincide, and we want the computed line to tend towards the line through $p_1 = p_2$ and c. When using the perpendicular bisector of the two randomly chosen points p_1 and p_2, this property is not achieved.

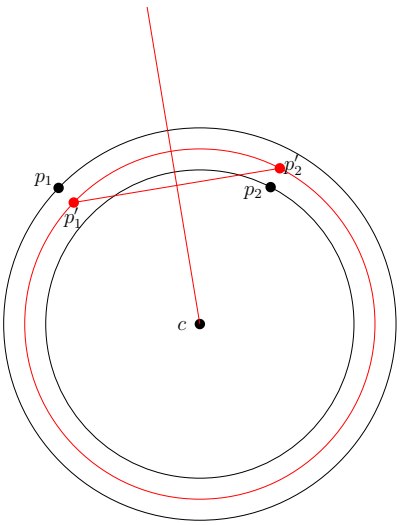


(a) The reflection line defined by p_1 and p_2 might not pass through the point c.

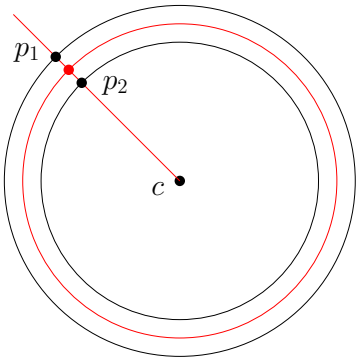
(b) The computed reflection line is perpendicular to the expected line which contains the three points p_1, p_2 and c.

Figure 2.3.6: Two reasons why the perpendicular bisector of the line segment $\overline{p_1p_2}$ is not a good choice.

A better way is to compute two points p_1' and p_2' which lie on the boundary of a disk with center c and radius $r = \frac{d(c,p_1)+d(c,p_2)}{2}$. The two points are defined to be the intersection of the line segments $\overline{cp_1}$ and $\overline{cp_2}$ with the boundary of that disk, respectively. The reflection line defined by the two points p_1' and p_2' fulfils the two required properties: It passes through the point c and it tends to the perpendicular bisector of the line segment $\overline{p_1p_2}$ if ε tends to zero. Figure 2.3.7 gives an illustration of these two properties.



(a) The reflection line defined by p'_1 and p'_2 passes through the point c.



(b) The computed reflection line equal to the expected line which contains the three points p_1, p_2 and c.

Figure 2.3.7: The perpendicular bisector of the line segment $\overline{p'_1 p'_2}$ is a good choice.

Observation 2.3.12. *Let two points p_1 and p_2 and a rotation center c be given. Let $r_1 = d(p_1, c)$ and $r_2 = d(p_2, c)$ be the distances between p_1, p_2 and c, respectively. Let $r = \frac{r1+r2}{2}$ be the average of the radii of the disks defined by p_1, p_2 and c, respectively. The intersection point of the line defined by p_1 or p_2 and c and the disk defined by c and r is the point $p'_1 = \frac{r}{r_1}p_1 + \left(1 - \frac{r}{r_1}\right)c$ or the point $p'_2 = \frac{r}{r_2}p_2 + \left(1 - \frac{r}{r_2}\right)c$, respectively.*

Algorithm 2.3.7 Computing the two averaged points p'_1 and p'_2 from the randomly chosen points p_1 and p_2

AveragePoints(p_1, p_2, c)
$r_1 = d(p_1, c)$;
$r_2 = d(p_2, c)$;
$r = \frac{r_1+r_2}{2}$
$p'_1 = \frac{r}{r_1}p_1 + \left(1 - \frac{r}{r_1}\right)c$
$p'_2 = \frac{r}{r_2}p_2 + \left(1 - \frac{r}{r_2}\right)c$
return p'_1 and p'_2

Representing the Reflection Line

From Section 1.1.2 we know that a D_k-symmetric shape has $2k$ reflection lines which all pass through the rotation center and that the angle between two reflection lines is a multiple of half the rotation angle. It is therefore convenient to represent a reflection line by the angle α between the line and the x-axis. In contrast to the angles computed for the rotational symmetry detection in the previous section, the computed angle α which represents the reflection line is in the interval $[0, \pi]$. This is due to the fact that the reflection line represented by α is the same as the one represented by $\alpha + \pi$.

The two points p'_1 and p'_2 computed by Algorithm 2.3.7 by using the randomly chosen points p_1 and p_2 lie on a circle with center c. The perpendicular bisector of $\overline{p'_1 p'_2}$ is the angle bisector of the lines $\overline{cp'_1}$ and $\overline{cp'_2}$. The angle between the perpendicular bisector of the line segment $\overline{p'_1 p'_2}$ and a line parallel to the x-axis and passing through c is given as follows:

Observation 2.3.13. *Let two points $p'_1 = (p'_{1x}, p'_{1y}), p'_2 = (p'_{2x}, p'_{2y}) \in \mathbb{R}^2$ with equal distance r to a rotation center c be given. Let l_x be the line parallel to the x-axis and passing through c, let l_1 be the line segment $\overline{cp'_1}$ and let l_2 be the line segment $\overline{cp'_2}$. Let α_1 be the angle between l_1 and l_x and let α_2 be the angle between l_2 and l_x. The angle α between the perpendicular bisector l of the line segment $\overline{p'_1 p'_2}$ and l_x is given by $\alpha = \frac{\alpha_1 + \alpha_2}{2}$, where*

$$\alpha_1 = \begin{cases} \arccos\left(\frac{p'_{1x} - c_x}{r}\right) & \text{if } p'_{1y} \geq 0 \\ \arccos\left(\frac{p'_{1x} - c_x}{r}\right) + \pi & \text{if } p'_{1y} < 0 \end{cases}$$

and

$$\alpha_2 = \begin{cases} \arccos\left(\frac{p'_{2x} - c_x}{r}\right) & \text{if } p'_{2y} \geq 0 \\ \arccos\left(\frac{p'_{2x} - c_x}{r}\right) + \pi & \text{if } p'_{2y} < 0 \end{cases}$$

Algorithm 2.3.8 states the procedure of calculating the angle representing the perpendicular bisector of the line segment defined by two points p'_1 and p'_2.

Clustering

Using this representation of the reflection lines, the transformation space again is a one-dimensional interval. Since a line represented by an angle $\alpha > \pi$ is the same line as the one represented by $\alpha - \pi$, we can restrict the transformation space for the reflectional symmetry detection to the interval $[0, \pi]$. We do this by subtracting π from each computed angle greater than π. Thus we can use the same clustering algorithm stated in Algorithm 2.3.5, as for the rotational symmetry detection algorithm the only difference is the upper bound of the interval.

Algorithm 2.3.8 Computing the angle representing the perpendicular bisector of the line segment $\overline{p_1 p_2}$.

ComputeReflectionLine(p_1, p_2, c)
$\alpha_1 = \arccos\left(\frac{p_{1x}-c_x}{r}\right)$;
if $p_{1y} < 0$ **then**
$\alpha_1 = \alpha_1 + \pi$;
end if
$\alpha_2 = \arccos\left(\frac{p_{2x}-c_x}{r}\right)$;
if $p_{2y} < 0$ **then**
$\alpha_2 = \alpha_2 + \pi$;
end if
$\alpha = \frac{\alpha_1+\alpha_2}{2}$;
return α;

Computing the Symmetry Group

Suppose we already computed the cyclic symmetry group C_k of the given input shape as described in Section 2.3.1. What remains to be done is to decide whether the input shape has reflectional symmetry w.r.t. some reflection line l. As stated in Chapter 1, if a shape has reflectional symmetry w.r.t. a line l, it also has reflectional symmetry w.r.t the lines $l_i, 1 \leq i \leq 2k-1$, where l_i passes through c, and the angle between l and l_i is $\frac{2\pi i}{2k}$, where k is the order of the cyclic symmetry group of the considered shape.
In contrast to the algorithm for detecting the rotational symmetry group, we are lucky that we already know the order k of the symmetry group. Thus we need to check if there are $2k$ angles representing reflection lines among the clusters with large number of votes. As for the detection of the rotational symmetry group we sort the clusters in a decreasing number of votes and consider the first $2k$ entries in this ordered list. We sort the $2k$ representing angles in this list in increasing order. After subtracting the minimum angle in this list, we test if the difference between angle α_i and $\frac{i360^\circ}{2k}$ is smaller than a given threshold. In our experiments 3° turned out to be a good threshold. If this is the case for all $2k$ angles, the symmetry group of the given input shape is the dihedral group D_k, otherwise it is the cyclic group C_k.

The algorithm for deciding whether the given input shape has symmetry group D_k is given in Algorithm 2.3.9.

Algorithm 2.3.9 Algorithm for deciding whether a given shape has symmetry group D_k for given set of clusters $\mathcal{C}$ of the transformation space and order k of the cyclic symmetry group.

ReflectionalSymmetry(Cluster $\mathcal{C}$, k)
  Sort the clusters in $\mathcal{C}$ by decreasing number of votes;
  for $i := 0$ to $2k$ **do**
    $\alpha_i := C_i$.Representative();
  end for
  Sort the angles in increasing order;
  //Use boolean variable to check if the distance between two successive angles is $\frac{360^\circ}{2k} \pm 3^\circ$.
  $b = TRUE$;
  for $i := 0$ to $2k - 1$ **do**
    $\alpha_i = \alpha_i - \alpha_0$
    $b = b \wedge \left(\left|\left|\alpha_i - \frac{i360^\circ}{2k}\right|\right| \leq 3^\circ\right)$;
  end for
  return b;

Number of Votes Needed

The algorithm for determining the symmetry group by using a probabilistic approach terminates after sufficiently many votes are generated.
Scharf [38] developed and proved bounds for the number of votes needed in order to determine a transformation which approximates w.r.t. a given value ε the optimal transformation with probability larger than a predefined probability. The optimization is done with respect to the size of the set of pairs of points, one of each set, which are mapped by the transformation to each other within a δ-neighborhood.
The sufficient number of votes is than given dependent on δ, ε and the predefined probability.
The theoretically determined bounds for the number of votes that suffice differ significantly from the number of votes that turned out to be sufficient in order to obtain good results in the experiments made by Scharf.
We therefore will not state theoretical results for the number of votes we need in our algorithm but rely on the results we got by experiments made. It turned out that 500 to 1000 votes suffice in order to detect the symmetry group of the given input if the distortion of the input is moderate.

The Algorithm

We are now ready to state the complete algorithm for detecting the symmetry group of a given input shape. The algorithm combines all techniques for detecting rotational and reflectional symmetry stated in this section in order to determine the symmetry group.

Algorithm 2.3.10 Probabilistic algorithm for symmetry detection

DetectSymmetryGroup(Shape $\mathcal{S}$)
c :=CenterOfMass($\mathcal{S}$);
repeat
Choose two points p_1 and p_2 randomly from the input shape $\mathcal{S}$ either by Algorithm 2.3.2 or by Algorithm 2.3.3.
//Compute the rotation angle defined by p_1 and p_2.
$\alpha_{\text{rot}} := \arccos\left(\frac{d(p_1,c)+d(p_2,c)-d(p_1,p_2)}{2d(p_1,c)d(p_2,c)}\right)$;
Vote for α_{rot} in transformation space T_{rot};
//Compute two points p_1' and p_2' with the same distance to c and positive y-coordinates by using Algorithm 2.3.7.
$p_1', p_2' = AveragePoints(p_1, p_2, c)$;
$\alpha_{\text{refl}} = ComputeReflectionLine(p_1', p_2')$;
Vote for α_{refl} and $(\alpha_{\text{refl}} + 180°)$ in transformation space T_{refl};
until enough samples taken*//See "Number of Votes Needed" Remark*
Cluster the votes in transformation space T_{rot} by using Algorithm 2.3.5 resulting in clusters $\mathcal{C}_{\text{rot}}$;
Cluster the votes in transformation space T_{refl} by using Algorithm 2.3.5 resulting in clusters $\mathcal{C}_{\text{refl}}$;
Compute the rotational symmetry group C_k from the clusters in $\mathcal{C}_{\text{rot}}$ by using a combination of Strategy 1 and Strategy 2;
//Decide whether $\mathcal{S}$ is reflectionally symmetric by using Algorithm 2.3.9;
if ReflectionalSymmetry($\mathcal{C}_{\text{refl}}$,$k$) **then**
return D_k;
else
return C_k;
end if

Chapter 3

ε-Symmetry Detection

In the last chapter we discussed how to determine the symmetry group of a given point set, even if the point set, due to noise or other perturbation, is not exactly symmetric. In this case, it is interesting to ask how symmetric (or asymmetric) the point set is, meaning how large the distance between this set and a symmetric point set is.

In this chapter, we will define the *ε-Symmetry Detection* problem, which is the problem of finding for a given input set P and a given symmetry group S an S-symmetric point set Q which is a close approximation of P. We will only consider finite symmetry groups, thus $S \in \{C_k, D_k | k \in \mathbb{N}^+\}$.

The ε-SD decision problem was already studied by Iwanowski [18]. He proved it to be NP-complete in general and to be in P if certain restrictions are added to the problem definition.

In this chapter, we will investigate the ε-SD decision problem as well as the ε-SD computational problem. We will reproduce some of the results already given by Iwanowski [18], but we will use different methods. In contrast to the proofs given by Iwanowski [18], our proofs are more geometrical and therefore we are able to produce algorithms which can be implemented without much effort.

In Section 3.1 we will start by defining the ε-SD problem. Afterwards, we will list several variations of the problem. In the remaining sections of this chapter we will state the algorithms and prove their complexity and running time for each variant of the ε-SD problem introduced in Section 3.1.

3.1 Complexity of the ε-Symmetry Detection Problem

Let us assume we are given a finite set of points P and a symmetry group S, where $S \in \{C_k, D_k | k \in \mathbb{N}^+\}$. We will now state the problem of finding an S-symmetric point set Q which is a close approximation of P. Prior to the definition of the ε-SD problem, we will give a formal definition of our notion of "close approximation", which we take from Iwanowski [18]:

Definition 3.1.1 (ε-Approximation)**.** Let two finite point sets $P \subset \mathbb{R}^2$ and $Q \subset \mathbb{R}^2$, both containing n points and a real number $\varepsilon \geq 0$, be given. We say that Q *ε- approximates* P, iff there exists a bijective function $f : P \to Q$, so that $|p - f(p)| \leq \varepsilon$, for all $p \in P$.

Using this definition, we can define the ε-SD decision problem as follows:

Problem 3.1.2 (The ε-SD decision problem)

Given: A set $P \subset \mathbb{R}^2$ of $n = mk$ points, a symmetry group S, where S is either C_k or D_k, $k \in \mathbb{N}^+$ and a real number $\varepsilon \geq 0$.

Question: Is there a set $Q \subset \mathbb{R}^2$ having symmetry group S, which ε- approximates P?

The corresponding computational problem is defined as follows:

Problem 3.1.3 (The ε-SD computational problem)

Given: A set $P \subset \mathbb{R}^2$ of $n = mk$ points and a symmetry group S, where S is either C_k or D_k, $k \in \mathbb{N}^+$.

Task: Compute the smallest $\varepsilon \geq 0$, so that there is a set Q having symmetry group S, which ε- approximates P.

Solving the ε-SD decision problem for a given point set P, a symmetry group S and an ε is at most as complicated as solving the ε-SD computational problem for P and S since the solution ε_{opt} of the computational problem gives an answer to the decision problem by simply checking if $\varepsilon_{opt} \leq \varepsilon$.
In figures illustrating examples for the ε-SD decision problem, the given value $\varepsilon \geq 0$ will be indicated by disks of radius ε centered at the points of the input set P. The solution of the ε-SD decision problem is then given by a symmetric point set Q, so that each point of Q lies in exactly one disk.

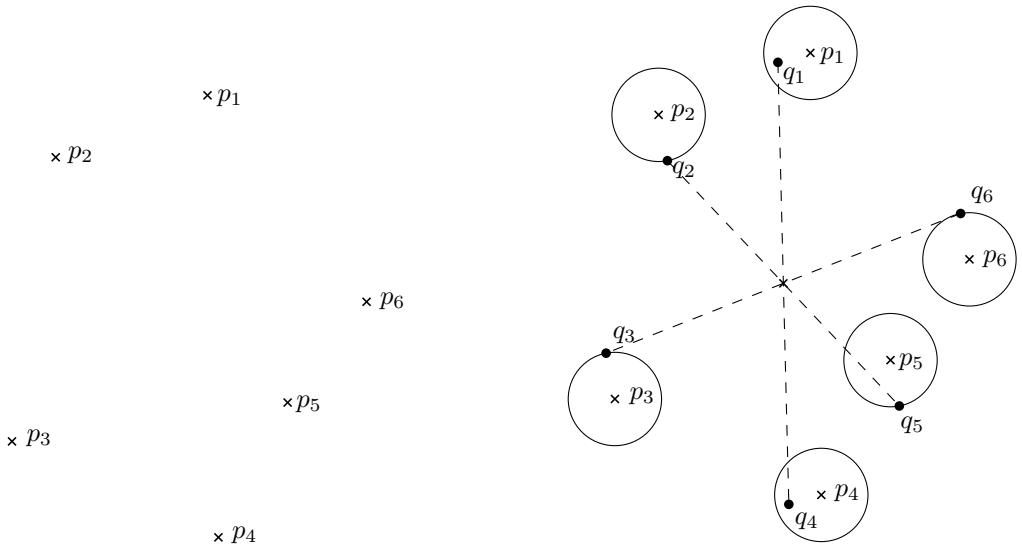
In the case where we consider the ε-SD computational problem, the input is given by a point set P and the solution is the smallest possible value $\varepsilon_{\text{opt}} \geq 0$, so that there is a point set Q where each point of Q lies in exactly one disk with radius ε_{opt} centered at a point of P.

The ε-SD decision problem is NP-complete in general, which was shown by Iwanowski [18]. However the ε-SD problem becomes easier and solvable in polynomial time if we either restrict it to certain symmetry groups or add more information about the point sets P and Q, respectively, to the problem definition.

In this section we will introduce and define some of those special cases of the ε-SD problem and state the complexity of each of them. Furthermore, we will use one section for each of these variants of the ε-SD problem in order to state an algorithm solving the problem considered, prove its correctness and analyze its running time.

The ε-SD Problem for the Symmetry Group C_2

The first restriction on the ε-SD problem we make is on the considered symmetry group. Iwanowski [18] proved the ε-SD *decision* problem to be in P if only the symmetry groups D_1 and C_2 are allowed. In Section 3.2 we will state a polynomial time algorithm solving the ε-SD *computational* problem for symmetry group C_2. As stated above, this also gives another proof for the ε-SD decision problem to be in P. In contrast to the proof of Iwanowski [18], our proof leads directly to a polynomial time algorithm which can be implemented quite easily.



(a) The input to the ε-SD computational problem for symmetry group C_2.

(b) The solution is given by the value of ε_{opt} and the C_2-symmetric point set Q.

Figure 3.1.1: Illustration of the definition of the ε-SD computational problem for symmetry group C_2.

Problem 3.1.4 (The ε-SD computational problem for symmetry group C_2)

Given: A set $P \subset \mathbb{R}^2$, $|P| = n = 2m, m \in \mathbb{N}$.

Task: Compute the smallest $\varepsilon \geq 0$, so that there is a point set Q having symmetry group C_2, which ε- approximates P.

The ε-SD Problem for the Symmetry Group $D_{|P|}$

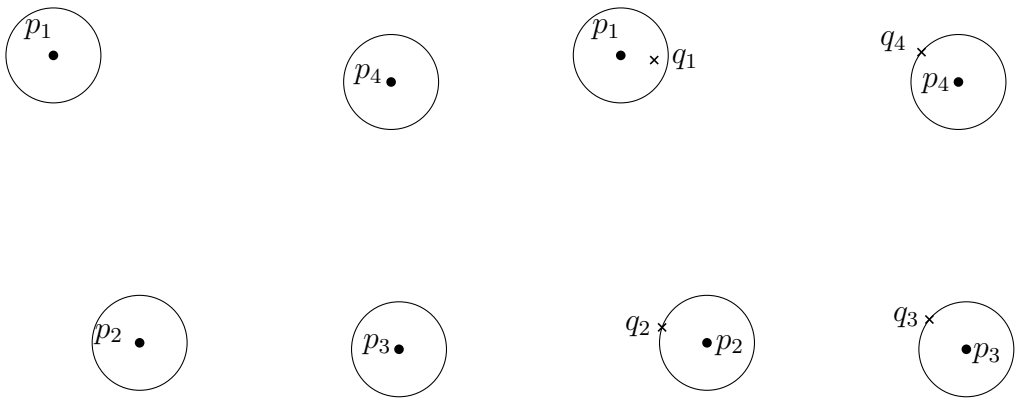
Another possibility is to restrict the symmetry group depending on the size of the point set P. For a point set of size $|P|$ we ask for a $D_{|P|}$-symmetric point set Q, $|Q| = |P|$ which approximates P. This variant of the ε-SD problem where the symmetry group is directly given by the number of points in P can be decided in polynomial time, as we will see in Section 3.3.

This variant of the ε-SD decision problem is defined as follows:

Problem 3.1.5 (The ε-SD decision problem for symmetry group $D_{|P|}$)

Given: A point set $P \subset \mathbb{R}^2$ and a value $\varepsilon \geq 0$.

Question: Is there a $D_{|P|}$-symmetric point set Q which ε- approximates P?



(a) The input to the ε-SD decision problem in this example is a set P of size $|P| = 4$ and a value ε.

(b) The solution is given by the C_4-symmetric point set Q.

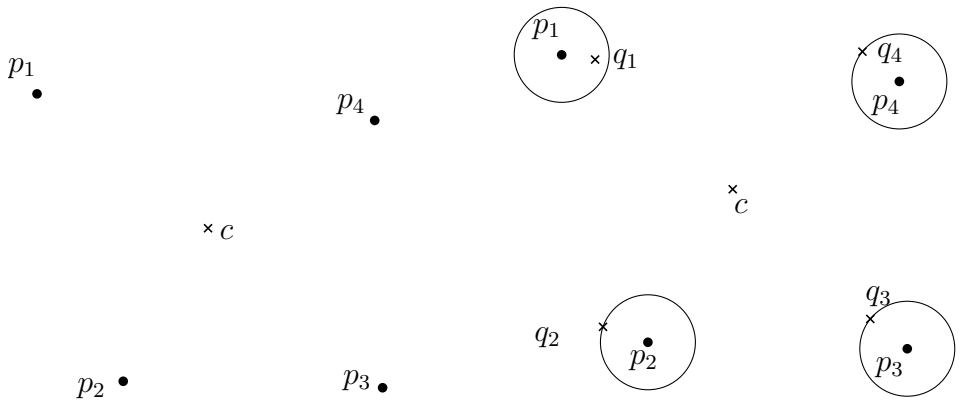
Figure 3.1.2: Illustration of the definition of the ε-SD decision problem for symmetry group $D_{|P|}$.

Solving this problem becomes easier if Q has to have a predefined rotation center which is part of the input.

Problem 3.1.6 (The ε-SD computational problem for symmetry group $D_{|P|}$ and rotation center c)

Given: A point set $P \subset \mathbb{R}^2$ and a rotation center c.

Task: Compute the smallest value $\varepsilon \geq 0$ so that there is a $D_{|P|}$-symmetric point set Q with rotation center c which ε- approximates P.



(a) The input to the ε-SD computational problem in this example is a set P of size $|P| = 4$ and a rotation center c.

(b) The solution is given by the value of ε_{opt} and the C_4-symmetric point set Q.

Figure 3.1.3: Illustration of the definition of the ε-SD computational problem for symmetry group $D_{|P|}$ and given rotation center.

Both problems can be solved in polynomial time. We will state the algorithms and the proofs of correctness and running time in Section 3.3. The definitions of Problem 3.1.5 and Problem 3.1.6 only differ in the fact that in Problem 3.1.6 the rotation center of Q is part of the input of the problem. Since more information is given, it is easier to solve this variant of the ε-SD problem. We are able to present a polynomial time algorithm for the ε-SD *computational* problem for symmetry group $D_{|P|}$ in the case where the rotation center of Q is given, and a polynomial time algorithm which decides the ε-SD *decision* problem for symmetry group $D_{|P|}$ if the rotation center of Q is not given. We will give an example of each variant of the ε-SD problem. Figure 3.1.2 gives an example of the case where the rotation center of Q is not given. Since it is a decision problem, the value of ε is part of the input which is indicated by the disks centered at the points of the input set P. Figure 3.1.3 shows an example of the case where the rotation center of Q is part of the input.

The ε-SD Problem for Given Partition of P.
If we do not want to restrict the considered symmetry group in order to be able to state polynomial time algorithms, we can add more knowledge about P to the input of the problem.
This leads to another restricted version of the ε-SD problem which we are able to solve in polynomial time.
We demand the input of the ε-SD problem for symmetry group C_k to be a partition of the point set P where all partition sets $P_1, \ldots, P_m$ have size k.
We then can state polynomial time algorithms solving the ε-SD problem for symmetry group C_k in polynomial time by applying the algorithms solving the ε-SD problem for symmetry group $D_{|P_i|}$ to each partition set separately and combining the results.
The problem is then defined as follows:

Problem 3.1.7 (The ε-SD computational problem for given partition of P and a given rotation center c.)

Given: A set $P \subset \mathbb{R}^2, |P| = mk$, a symmetry group C_k, a partition of P into m subsets $P_1, \ldots, P_m$ of size $|P_i| = k, 1 \leq i \leq m$ each and a rotation center c.

Task: Compute the smallest value $\varepsilon \geq 0$, so that there exists a C_k-symmetric point set Q and a partition of Q into m subsets $Q_1, \ldots, Q_m$ all of size k, all subsets having symmetry group D_k and rotation center c, so that Q_i ε-approximates P_i, for all $1 \leq i \leq m$.

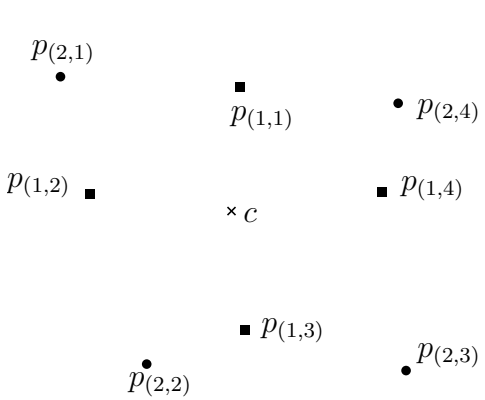
Without the knowledge of the rotation center c, we will consider the following decision problem:

Problem 3.1.8 (The ε-SD decision problem for given partition of P)

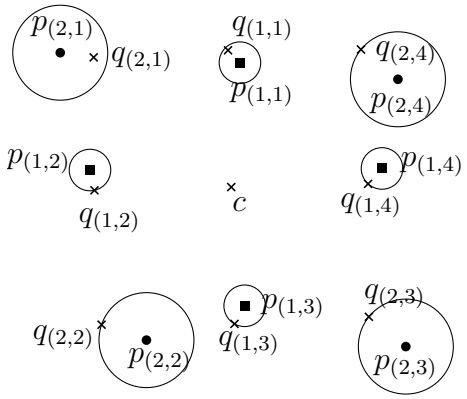
Given: A set $P \subset \mathbb{R}^2, |P| = mk$, a symmetry group C_k and a partition of P into m subsets $P_1, \ldots, P_m$ of size k each and a value $\varepsilon \geq 0$.

Question: Is there a C_k-symmetric point set Q and a partition of Q into subsets $Q_1, \ldots, Q_m$ all of size k, all having symmetry group D_k, and all having the same rotation center so that Q_i ε- approximates P_i for all $1 \leq i \leq m$?

Examples of the ε-SD problem where the partition of P into subsets of size k each is given are depicted in Figure 3.1.4 for the computational case where the rotation center is known and in Figure 3.1.5 for the decision problem where the rotation center is not known. The points in different partition sets are indicated by different shapes.

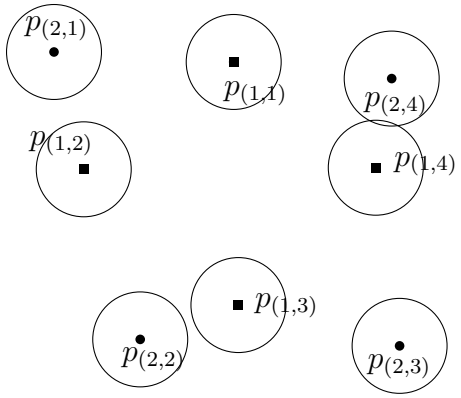


(a) The input to the ε-SD computational problem for symmetry group C_4 with given partition of P and a rotation center c.

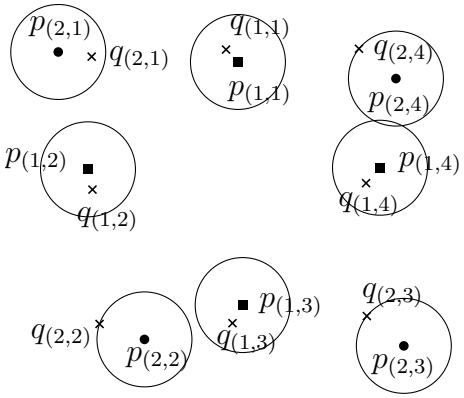


(b) The solution is given by the value of ε, indicated by the disks of radius ε and the C_4-symmetric point set Q.

Figure 3.1.4: Illustration of the definition of the ε-SD computational problem for given partition of P and given rotation center.



(a) The input to the ε-SD decision problem for symmetry group C_4 with given partition of P and value ε.



(b) The solution is given by the C_4-symmetric point set Q.

Figure 3.1.5: Illustration of the definition of the ε-SD decision problem for given partition of P and given ε.

We will present algorithms with polynomial running time for Problem 3.1.7 and Problem 3.1.8 and prove their correctness in Section 3.4. There we will see that we can reduce the ε-SD problem with given partition of P to solving the ε-SD computational problem for symmetry group $D_{|P_i|}$ on each partition set P_i. We will show how to combine the solutions of the partition sets in order to get a solution for the whole set P.

The ε-SD Problem for γ-Disjoint Point Sets

In Section 3.5 we will present a polynomial time algorithm deciding the ε-SD problem. Here we need no restriction on the symmetry group and no further input information like a partition of P or a rotation center of Q. The only assumption we make is that the input point set is well separated. This means that the points do not lie too close together with respect to the given ε. Prior to giving the definition of this variant of the ε-SD problem we define the meaning of "not too close together", which we take from Iwanowski [18]:

Definition 3.1.9 (γ-Disjoint Point Set)**.** Let $P = \{p_1, \ldots, p_n\}$ be a set of points in the plane. We call P *γ-disjoint*, iff $d(p_i, p_j) > \gamma$, for all $p_i, p_j \in P, p_i \neq p_j$, where $d(p_i, p_j)$ denotes the Euclidean distance between the points p_i and p_j.

We consider the following variant of the ε-SD problem for disjoint point sets:

Problem 3.1.10 (The ε-SD decision problem for $t\varepsilon$-disjoint point set P for some fixed constant t)

Given: A symmetry group C_k, a value $\varepsilon \geq 0$ and a $t\varepsilon$-disjoint point set $P \subset \mathbb{R}^2$, $|P| = n = mk$.

Question: Is there a C_k-symmetric point set Q which ε- approximates P?

Iwanowski [18] proved the ε-SD decision problem to be in P in the case where the input point set P is 8ε-disjoint. In Section 3.5 we will prove that the ε-SD decision problem for an input set P can be solved in polynomial time if P is $4t\varepsilon$-disjoint. The factor t results from a rotation center c where c can be computed dependent on P in polynomial time. Furthermore the solution of the ε-SD computational problem for P, restricted to the rotation center c has to be a t-approximation of the solution of the ε-SD computational problem for P. Using this general proof, we will give polynomial time algorithms for $t = 2$ and $t = \frac{2}{\sqrt{3}}$.

In the case where the rotation center c of the C_k-symmetric point set Q is given, we are able to decide the ε-SD decision problem in polynomial time if the input point set P is 4ε-disjoint. We will also prove that, for each value $\delta > 0$, the ε-SD decision problem for symmetry group C_k can be solved in time $O\left(\left(\frac{1}{\delta^2}\right)\left(n^3k + n^2k^3\right)\right)$ if the point set P is $4(1+\delta)\varepsilon$-disjoint.

3.2 The ε-SD Problem for Symmetry Group C_2

The ε-SD problem can be solved in polynomial time if it is restricted to the symmetry group C_2.
For the decision problem, Iwanowski [18] stated an algorithm which solves the ε-SD decision problem for the symmetry group C_2 in time $O(n^6)$ where n is the size of the point set.
In this section, we will state an algorithm solving the ε- SD computational problem for the symmetry group C_2 and an input point set $P, |P| = 2m$ in time $O(m^8)$. Since the algorithm is based on simple geometric calculations, it can be implemented without much effort.

We will consider the following problem, which we already defined in Section 3.1:

Problem 3.1.4 (The ε-SD computational problem for symmetry group C_2)

Given: A set $P \subset \mathbb{R}^2$, $|P| = n = 2m, m \in \mathbb{N}$.

Task: Compute the smallest value $\varepsilon \geq 0$, so that there is a point set Q having symmetry group C_2, which ε- approximates P.

A C_2-symmetric point set Q has the property that it is invariant under a rotation by π around a rotation center c. This implies that Q can be partitioned into $m = \frac{n}{2}$ subsets $Q_1, \dots Q_m$, each of size 2, with the property that $q_{(i,1)} = \rho_c^{\pi}(q_{(i,2)})$ and $q_{(i,2)} = \rho_c^{\pi}(q_{(i,1)})$, for all $Q_i = \{q_{(i,1)}, q_{(i,2)}\}$, $1 \leq i \leq m$.
For a given, not necessarily symmetric, point set P the crucial steps in order to solve the ε-SD problem for the symmetry group C_2, are

1. to find the partition of P into $m = \frac{n}{2}$ subsets, each of size 2, that results in the smallest possible ε.

2. to find the rotation center of the C_2-symmetric set Q.

Once the partition and the rotation center are known, the smallest value $\varepsilon \geq 0$, so that there exists a C_2-symmetric point set Q that ε-approximates P, can be computed easily.

3.2.1 Partition of P Known

We will start our investigation by concentrating on how to find the rotation center c for the C_2-symmetric point set Q, so that the value ε is minimized. As we already indicated in Section 3.1, the task to find c becomes easier if the partition of P into subsets of size 2 is given.
Since we consider the symmetry group C_2 in this section, we ask for a partition of P into m subsets of size 2 each.

Problem 3.2.1 (The ε-SD computational problem for symmetry group C_2 and a given partition of P.)

Given: A set $P \subset \mathbb{R}^2$ of n points, $n = 2m, m \in \mathbb{N}$, and a partition of P into subsets $P_1, \ldots, P_m$, $|P_i| = 2, 1 \leq i \leq m$.

Task: Find the smallest value ε, so that there is a C_2-symmetric point set Q which ε- approximates P, and Q can be partitioned into m subsets $Q_1, \ldots Q_m$, so that Q_i ε-approximates P_i, $1 \leq i \leq m$ and all Q_i have the same rotation center.

In order to solve the ε-SD computational problem for symmetry group C_2 and given partition of P, we need to consider the following two tasks:

The first task is to solve the problem locally by determining the smallest possible ε_i, so that Q_i ε_i-approximates P_i, for each $1 \leq i \leq m$ and all Q_i have the same rotation center c.

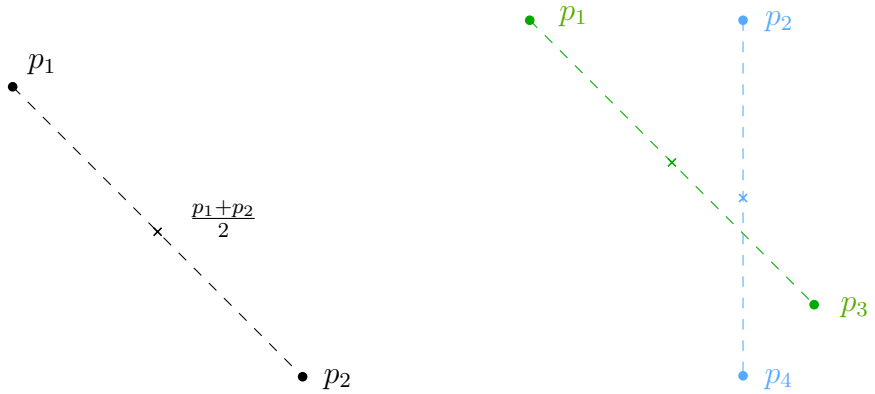
The second task is to determine a value ε so that the whole set Q ε-approximates the whole set P.

Suppose we already computed the optimal values of ε_i, for all $1 \leq i \leq m$ for the partition sets $P_1, \ldots, P_m$. We can then solve the ε-SD computational problem for the set P as follows:

Observation 3.2.2. *Let a point set P of size $|P| = 2m$, $m \in \mathbb{N}$, a partition of P into subsets $P_1, \ldots, P_m$ of size 2 and a rotation center $c \in \mathbb{R}^2$ be given. Let ε_i be the solution of the ε-SD computational problem for symmetry group C_2, input set P_i and rotation center c. Let Q_i be the C_2-symmetric point set which ε_i-approximates P_i. Then the point set $Q = \bigcup_{1 \leq i \leq m} Q_i$ is C_2-symmetric and Q ε-approximates P, where* $\varepsilon = \max\{\varepsilon_i | 1 \leq i \leq m\}$.

Remark 3.2.3. One partition set $P_i = \{p_{i_1}, p_{i_2}\}$ on its own is always C_2-symmetric, since the point $a_i = \frac{p_{i_1} + p_{i_2}}{2}$ is the rotation center of the C_2-symmetric point set P_i.

By Remark 3.2.3 we see that each subset for itself is C_2-symmetric. This is only true for the whole set P if the rotation centers $a_1, \ldots a_m$ of the subsets $P_1, \ldots, P_m$ coincide. In general, for an arbitrary point set, this is not true (see Figure 3.2.1). The following Lemma3.2.4 relates the optimal solution for the ε-SD computational problem for a given rotation center c and input set P_i to the distance between c and a_i.



(a) A point set containing two points is C_2-symmetric

(b) Two subsets have different rotation centers in general.

Figure 3.2.1: Illustration of Remark 3.2.3

Lemma 3.2.4. *Let $P_i = \{p_{(i,1)}, p_{(i,2)}\}$ be a subset of the partition of P and a_i be the rotation center defined by P_i. Let a rotation center c be given. Let ε_i be the solution of the ε-SD computational problem for given rotation center c and input set P_i. Then $\varepsilon_i = d(a_i, c)$.*

Proof. In order to prove the lemma, we need to show that there is a point set $Q_i = \{q_{(i,1)}, q_{(i,2)}\}$ where $q_{(i,1)} = \rho_c^\pi(q_{(i,2)})$ and $d(q_{(i,1)}, p_{(i,1)}) \leq d(a_i, c) = \varepsilon_i$ and $d(q_{(i,2)}, p_{(i,2)}) \leq d(a_i, c) = \varepsilon_i$ and ε_i is minimized.

$d(q_{(i,2)}, p_{(i,2)}) \leq d(a_i, c) = \varepsilon_i$ is equivalent to $d(q_{(i,1)}, \rho_c^\pi(p_{(i,2)})) \leq d(a_i, c) = \varepsilon_i$, since $q_{(i,1)} = \rho_c^\pi(q_{(i,2)})$.

Thus solving the ε-SD computational problem for symmetry group C_2 and input set P_i with respect to the rotation center c is equivalent to computing a point q_i, so that the distance between q_i to $p_{(i,1)}$ and $\rho_c^\pi(p_{(i,2)})$, respectively is minimized. This point is the center of the smallest enclosing disk of the two points $p_{(i,1)}$ and $\rho_c^\pi(p_{(i,2)})$ which in this case is the midpoint of the line segment defined by these two points. Thus, $q_{(i,1)} = \frac{1}{2}\left(p_{(i,1)} + \rho_c^\pi(p_{(i,2)})\right)$ and $q_{(i,2)} = \rho_c^\pi\left(q_{(i,1)}\right)$. What remains to be done is to show that $d(p_i, q_i) = d(a_i, c)$. Since a_i is the midpoint of $\overline{p_{(i,1)}p_{(i,2)}}$ and c is the midpoint of the line segment $\overline{p_{(i,2)}\rho_c^\pi(p_{(i,2)})}$ we can apply the intercept theorem and get $d(p_{(i,1)}, q_{(i,1)}) = d(a_i, c)$ □

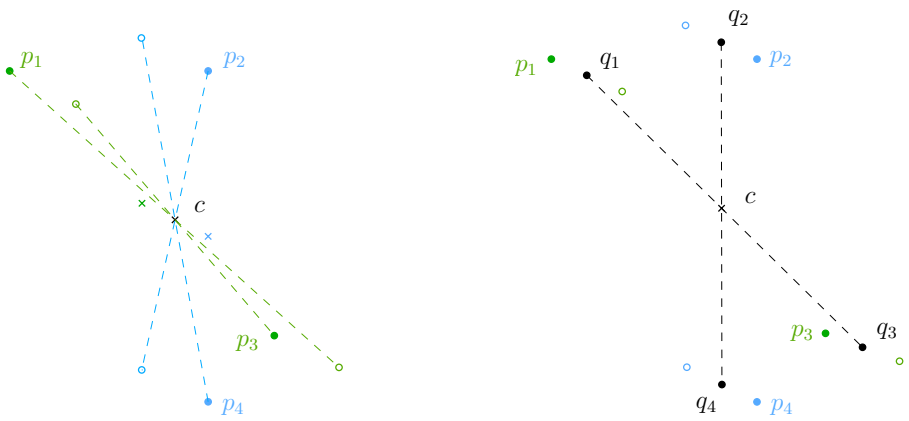
By Lemma 3.2.4 we can compute the solution for the ε-SD computational problem for symmetry group C_2, if the partition of P and the rotation center of Q are given:

Corollary 3.2.5. *Let a point set P, $|P| = 2m$, a partition of P into m subsets $P_1, \ldots, P_m$ of size 2 and a point c be given. Let $P_i = \{p_{(i,1)}, p_{(i,2)}\}$, $1 \leq i \leq m$. The solution of the ε-SD computational problem for symmetry group C_2 and known rotation center c for the input set P is then given by:*

$$\varepsilon = \max\{d(a_i, c) | 1 \leq i \leq m\}, \text{ where } a_i = \frac{p_{(i,1)} + p_{(i,2)}}{2}.$$

The value ε can be computed in time $O(m)$.

Proof. This follows directly from Lemma 3.2.4 and Observation 3.2.2. □



(a) Two partition sets and their rotational images from c.

(b) The optimal solution for P is given by the point set Q.

Figure 3.2.2: Example of the ε-SD computational problem for symmetry group C_2 with given partition and rotation center.

In the definition of the ε-SD computational problem for symmetry group C_2 and given partition of P, the rotation center is not given beforehand. In order to find a solution for the considered variation of the ε-SD problem we need to search for a point c, so that the maximum distance between this point c and all points a_i, where $1 \leq i \leq m$ is as small as possible. This follows directly from Corollary 3.2.5.

The following lemma characterizes the point c, so that $\max\{d(a_i, c) | 1 \leq i \leq m\}$ is minimized and therefore gives the solution for the ε-SD computational problem for symmetry group C_2 and given partition of P.

Lemma 3.2.6. *Let P be a set of $n = 2m$ points and $P_1, \ldots, P_m$ be a partition of P into m subsets each of size 2. Let $P_i = \{p_{(i,1)}, p_{(i,2)}\}$ and a_i be the midpoint of the segment $\overline{p_{(i,1)}p_{(i,2)}}$, $1 \leq i \leq m$. Furthermore, let $A = \{a_1, \ldots, a_m\}$ denote the set of midpoints of the partition sets. Let $D(c, r)$ be the smallest enclosing disk of A, where c denotes its center and r denotes its radius. Then r is the solution of the ε-SD computation problem for symmetry group C_2 and given partition of P. The center c of D is the rotation center of the C_2-symmetric point set Q which r-approximates P.*

Proof. Let $D(c, r_c)$ be the smallest enclosing disk of the set A from a fixed rotation center c. Then the radius of $D(c, r_c)$ is $r_c = \max\{d(c, a)|a \in A\}$ where $d(c, a)$ is the Euclidean distance between the two points c and a. The radius of the smallest enclosing disk $D(c_{opt}, r)$ of A is then $r = \min\{r_c|c \in \mathbb{R}^2\}$, and the center $c_{opt} \in \mathbb{R}^2$ of D is the point where the minimum radius is achieved.

For the partition of P into sets P_i, $1 \leq i \leq m$, let $\varepsilon_{c,i}$ be the solution for the ε-SD computational problem for symmetry group C_2 and given rotation center c for the input set P_i. As seen in Lemma 3.2.4, $\varepsilon_{c,i} = d(c, a_i)$. The solution ε_c for the whole set P is $\varepsilon_c = \max\{\varepsilon_{c,i}|1 \leq i \leq m\} = \max\{d(c, a_i)|a_i \in A\}$ as shown in Corollary 3.2.5. In order to find the minimum possible value ε for the set P, we need to find the point $c \in \mathbb{R}^2$ that minimizes the value ε_c. Therefore, $\varepsilon = \min\{\varepsilon_c|c \in \mathbb{R}^2\}$.

By setting $r_c = \varepsilon_c$ and $r = \varepsilon$ we see that the two problems, namely finding the smallest enclosing disk of A and finding the the solution of the ε-SD computational problem for symmetry group C_2 and given partition of P, are equivalent. We therefore get the solution for the ε-SD computational problem for symmetry group C_2 and given partition of P by computing the radius of the smallest enclosing disk of A. □

Algorithm 3.2.1 Algorithm for the ε-SD computational problem for symmetry group C_2 and given partition of P.

SDC2Partition($\mathcal{P} = \{P_1, \ldots, P_m\}$)
1 $A = \emptyset$;
2 *//Compute the set A of midpoints defined by $P_1 \ldots, P_m$.*
3 **for** $i = 1$ to m **do**
4 $\quad A = A \cup \{a_i = \frac{p_{(i,1)} + p_{(i,2)}}{2}\}$;
5 **end for**
6 *//Compute the smallest enclosing disk of A.*
7 $D = (c, r) =$**SmallestEnclosingDisk**(A);
8 **return** r;

Theorem 3.2.7. *Algorithm 3.2.1 solves the ε-SD computational problem for symmetry group C_2 and given partition of the input set P, $|P| = n$ in $O(n)$ time.*

Proof. The correctness of the algorithm follows directly from Lemma 3.2.6. What remains to be done is to analyze the running time.
Let n be the number of input points, so $n = 2m$. For each partition set P_i we need to compute the midpoint a_i, where $1 \leq i \leq m$. The computation of each midpoint takes constant time. Therefore, the computation of the set A of all midpoints can be done in $O(m)$ time.

The smallest enclosing disk of a set of m points can be computed in $O(m)$ time, as was shown by Megiddo [30].

Therefore Algorithm 3.2.1 has linear running time. □

3.2.2 Partition of P Not Known

In the previous section, we explained how to solve the ε-SD computational problem for symmetry group C_2 if a partition into $\frac{n}{2}$ subsets is given. Different partitions will result in different values of ε. One approach to solve the ε-SD computational problem for symmetry group C_2 would be to iterate over all possible partitions of P into subsets each of size 2, computing the value of ε for each partition and taking the minimum. This approach does not lead to a polynomial time algorithm, since the number of partitions of a set P containing n elements into subsets of size 2 is not polynomial in n.
In order to find a polynomial time algorithm for the ε-SD computational problem for symmetry group C_2, it is necessary to look at the relationship between two different partitions of the point set P more closely. As mentioned above, in order to compute the smallest possible ε for a given partition, we compute the smallest enclosing disk of the set A where A is the set of midpoints defined by the given partition.

Suppose we are given two different partitions of P into sets of size 2. This leads to two different sets of midpoints, A_1 and A_2. If $A_1 \cap A_2 \neq \emptyset$, then the two smallest enclosing disks, D_1 and D_2 intersect. It can also be the case, that D_1 and D_2 are the same although $A_1 \neq A_2$, as shown in Figure 3.2.3.

Thus different partitions may lead to the same smallest enclosing disk. Using the approach of examining all possible partitions would make us compute the same smallest enclosing disk again and again. We will therefore save a lot of time by considering only all possible smallest enclosing disks instead of examining all possible partitions of P.

In the case where the partition of P was given, we computed the smallest enclosing disk from the set of midpoints defined by that partition. We now will look at the problem the other way around and find the partition which defines the solution of the ε-SD computational problem for symmetry group C_2 from the set of all possible smallest enclosing disks.

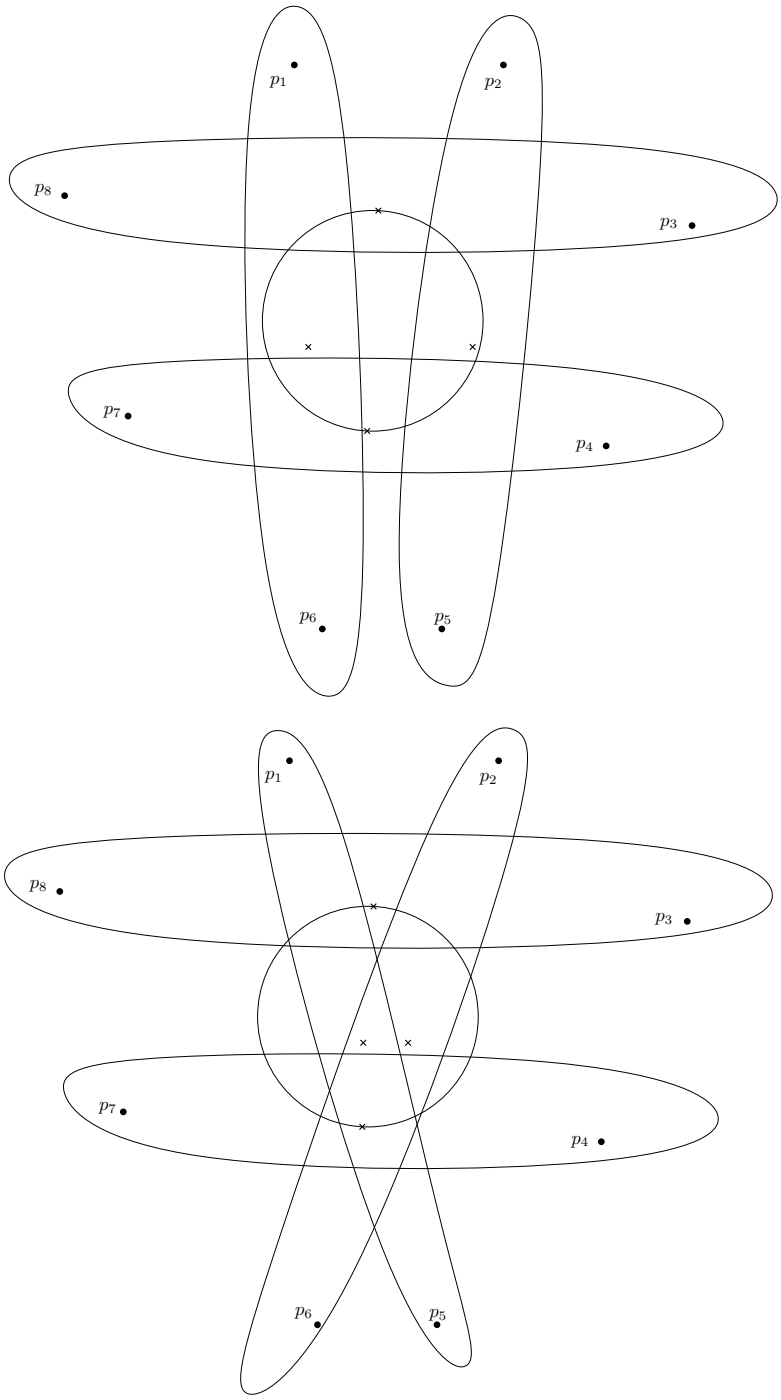


Figure 3.2.3: Two different partitions of P that result in the same smallest enclosing disk D since D is only defined by the midpoints of the two subsets $\{p_3, p_8\}$ and $\{p_4, p_7\}$

Lemma 3.2.8. *Let a point set P, $|P| = n$ be given. Let $\mathcal{P}$ be the set of all possible partitions of the point set P into subsets of size 2.*
Let $\mathcal{D}$ be the set of all smallest enclosing disks defined by the partitions in $\mathcal{P}$. Then $|\mathcal{D}| = O(n^6)$.

Proof. Let A^* be the set of the midpoints of all possible pairs of points in P. So $A^* = \{\frac{p_1+p_2}{2} | p_1, p_2 \in P \text{ and } p_1 \neq p_2\}$. Since P contains n points, A^* contains $\sum_{i=1}^{n-1}(n-i) = \frac{n(n+1)}{2} = \Theta(n^2)$ points. A smallest enclosing disk $D \in \mathcal{D}$ is defined by either two or three points in A^*. Thus there are $O(n^4)$ disks defined by two midpoints and $O(n^6)$ disks defined by three midpoints. Altogether we need to consider $O(n^6)$ different disks. □

Each disk which is a smallest enclosing disk of midpoints defined by some partition of P contains a set $A \subseteq A^*$ of $m = \frac{n}{2}$ midpoints corresponding to that partition. Additionally, it may contain some other midpoints as can be seen in Figure 3.2.4.

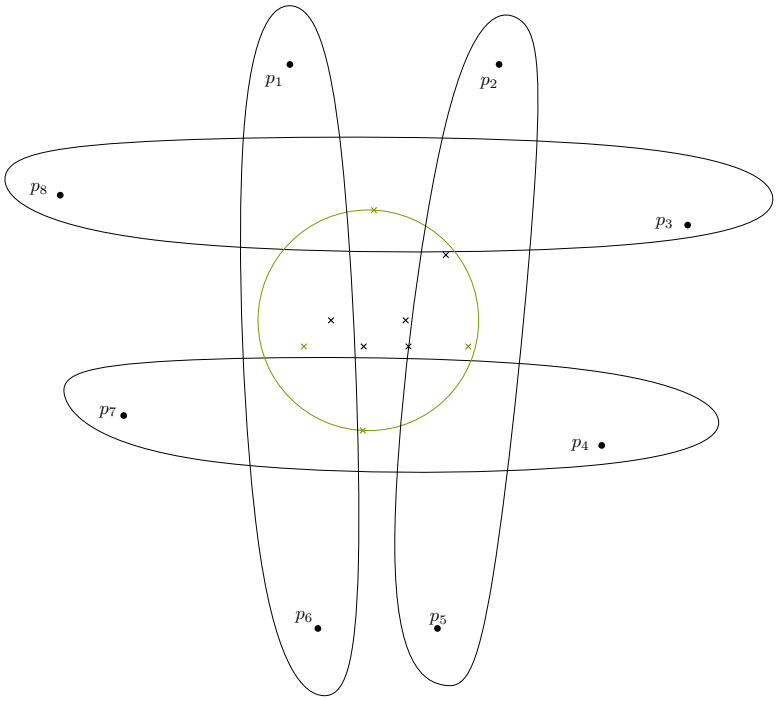


Figure 3.2.4: The smallest enclosing disk contains the midpoints of the depicted partition (green) and additional midpoints.

Since each disk D containing a subset $A \subseteq A^*$ is defined by either two or three points of A, we can compute all disks containing a subset of A^* by enumerating all pairs and triples of points of A^*. Let $\mathcal{D}^*$ be the set of all possible disks for the set of midpoints A^*. For each disk $D \in \mathcal{D}^*$ we need to check if it corresponds to a smallest enclosing disk of a set of midpoints $A \subseteq A^*$ which corresponds to a partition of P. Let $\mathcal{D} \subseteq \mathcal{D}^*$ be this set of disks. The disk $D \in \mathcal{D}$ having the smallest radius therefore corresponds to the partition of P which induces the solution of the ε-SD computational problem for symmetry group C_2.

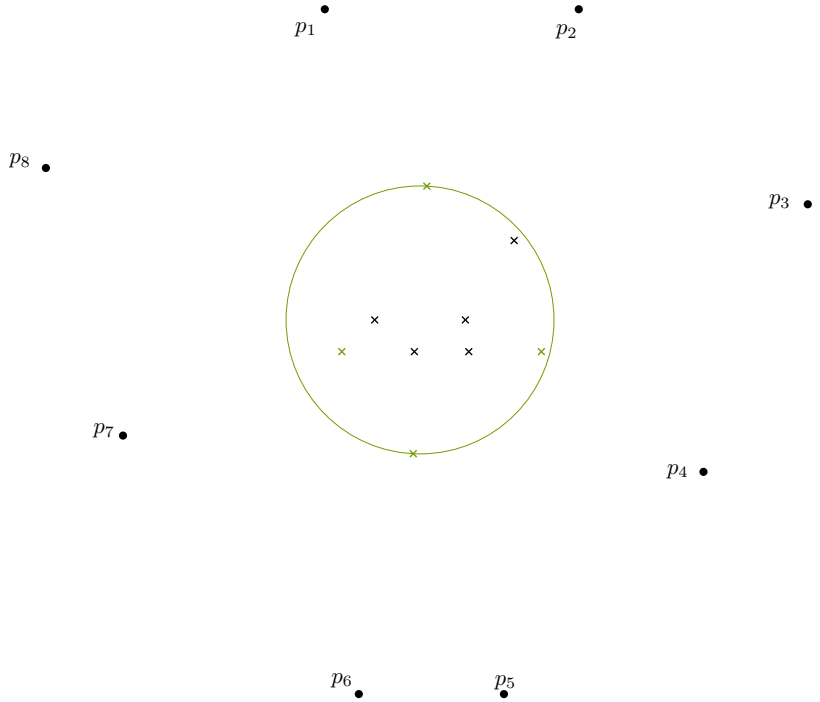
The crucial point is to determine if a disk $D \in \mathcal{D}^*$ contains a set of midpoints corresponding to a partition of P into subsets of size 2. Each set of midpoints $A = \{a_1, \ldots, a_m\}$ that corresponds to such a partition has the property that no point of P contributes to more than one midpoint of A. This means that $a_1 \in A$ and $a_2 \in A$ is true only if $a_1 = \frac{p_1+p_2}{2}$ and $a_2 = \frac{p_3+p_4}{2}$ for some $p_1, \ldots, p_4$ which are distinct.

We can regard the point set P as the vertex set V of an undirected graph $G = (V, E)$. Each midpoint defined by a pair of points of $p_1, p_2 \in P$ defines an edge $\{p_1, p_2\}$ of the edge set E. Let $D \in \mathcal{D}^*$ be a disk defined by two or three points of A^*. Let $A^D = D \cap A^*$ be the set of midpoints that are contained in the disk D. The disk D induces a graph $G^D = (V, E^D)$ on the point set P, where $V = P$ and $E^D = \{\{p_1, p_2\} | p_1, p_2 \in P \text{ and } a = \frac{p_1+p_2}{2} \in A^D\}$, see Figure 3.2.5.

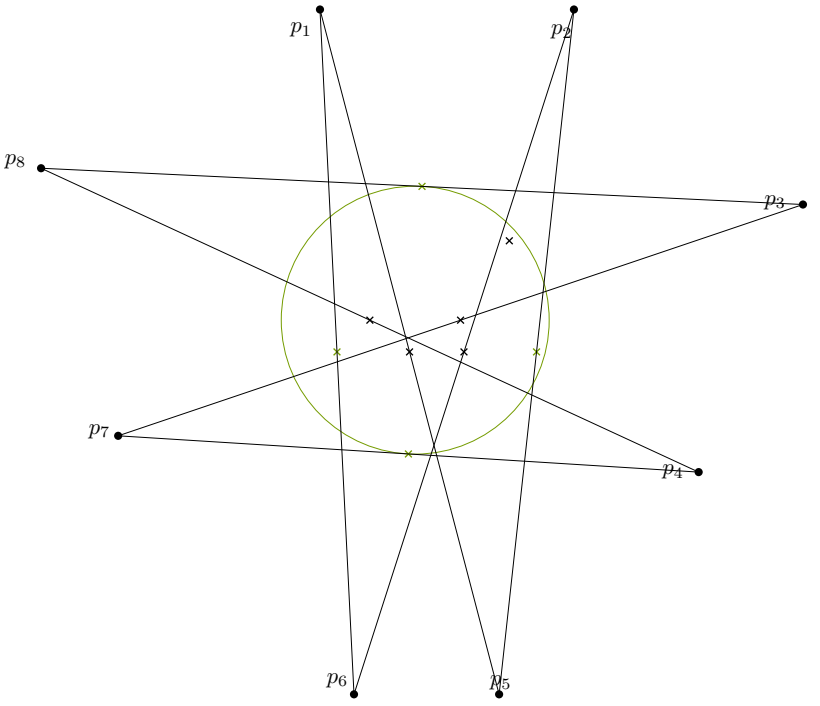
We can use the graph G^D in order to determine if the disk D contains a subset $A \subseteq A^*$ which corresponds to a partition of P as the following lemma shows:

Lemma 3.2.9. *Let P be a point set and let A^* be the set of midpoints of all pairs of points in P. Let D be a disk defined by two or three points in A^*. Let $A^D = D \cap A^*$. Let $G^D = (P, E^D)$, where $E^D = \{\{p_1, p_2\} | p_1, p_2 \in P \text{ and } a = \frac{p_1+p_2}{2} \in A^D\}$. Then D contains a disk $D^{\mathcal{P}}$ which is the smallest enclosing disk of the set of midpoints $A^{\mathcal{P}}$ corresponding to a partition $\mathcal{P}$ of P iff G^D contains a perfect matching.*

Proof. G^D contains a perfect matching
$\Leftrightarrow$
there exists a subset $E^{\mathcal{P}} \subseteq E^D$, so that each vertex in P is contained in exactly one edge of $E^{\mathcal{P}}$.
$\Leftrightarrow$
$P = \bigcup_{e \in E^{\mathcal{P}}} (e)$ and $e_i \cap e_j = \emptyset$, f.a. $e_i, e_j \in E^{\mathcal{P}}$ and $e_i \neq e_j$.
$\Leftrightarrow$
$E^{\mathcal{P}}$ defines a partition into subsets of size 2 on P and the subset $E^{\mathcal{P}} \subseteq E^D$ corresponds to a subset $A^{\mathcal{P}}$ of the midpoints A^D contained in disk D.
$\Leftrightarrow$
D contains a disk $D^{\mathcal{P}}$ which is the smallest enclosing disk of $A^{\mathcal{P}}$. □

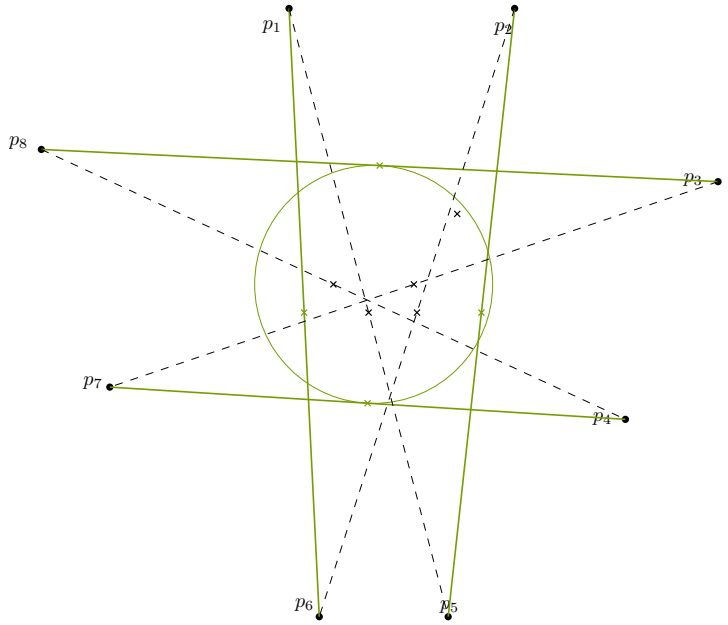


(a) Point set P and a disk D defining a set of midpoints $A^D \subseteq A^*$.

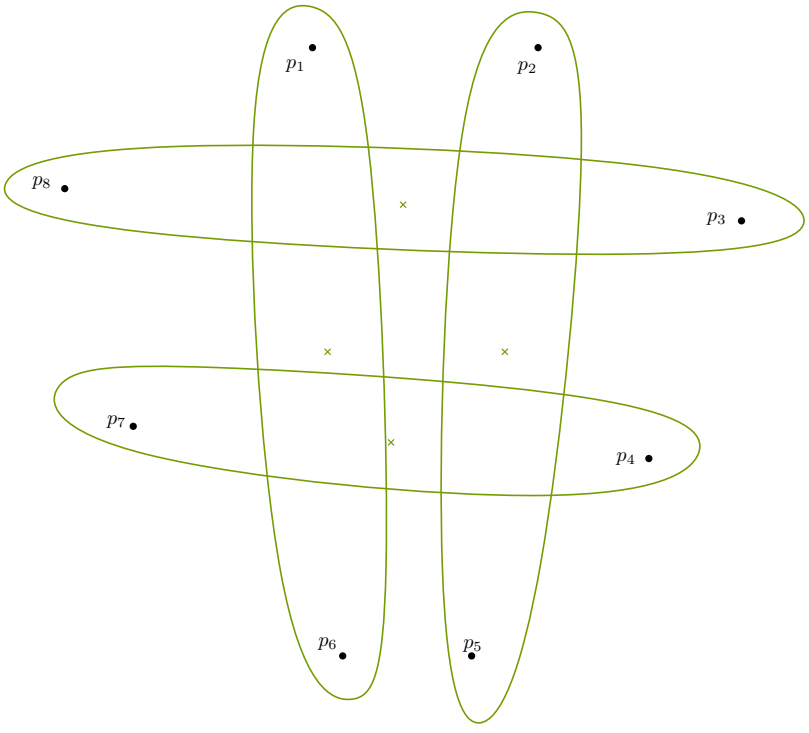


(b) Graph G induced by P and A.

Figure 3.2.5: Illustration of the construction of the graph D^G from P and D.



(a) The perfect matching in G.



(b) The partition of P into subsets of size 2.

Figure 3.2.6: Illustration of Lemma 3.2.9.

We will complete this section by stating the algorithm which solves the ε-SD computational problem for symmetry group C_2, proving its correctness and determining its running time.

Algorithm 3.2.2 A polynomial time algorithm for the ε-SD computational problem for symmetry group C_2.

SymmetryDetection$(P = \{p_1, \ldots, p_n\})$
1 *//Compute the midpoints of all pairs of points*
2 **for** $i = 1$ to n **do**
3 **for** $j = i + 1$ to n **do**
4 $a_{i,j} = \frac{p_i + p_j}{2}$; *//$a_{i,j}$ is the midpoint of p_i and p_j*
5 $A^* = A^* \cup \{a_{i,j}\}$; *//$A^*$ is the set of midpoints*
6 **end for**
7 **end for**
8 *//Compute all possible disks defined by two midpoints*
9 **for** $i = 1$ to $|A^*|$ **do**
10 **for** $j = i + 1$ to $|A^*|$ **do**
11 $D_{i,j} =$ **Disk**(a_i, a_j);
12 $\mathcal{D}^* = \mathcal{D}^* \cup \{D_{i,j}\}$; *//$\mathcal{D}$ is the set of disks*
13 **end for**
14 **end for**
15 *//Compute all possible disks defined by three midpoints*
16 **for** $i = 1$ to $|A^*|$ **do**
17 **for** $j = i + 1$ to $|A^*|$ **do**
18 **for** $k = j + 1$ to $|A^*|$ **do**
19 $D_{i,j,k} =$ **Disk**(a_i, a_j, a_k);
20 $\mathcal{D}^* = \mathcal{D}^* \cup \{D_{i,j,k}\}$;
21 **end for**
22 **end for**
23 **end for**
24 *//For each disk, check if it defines a partition $\mathcal{P}$ of P into subsets of size 2 each*
25 **for** $i = 1$ to $|\mathcal{D}^*|$ **do**
26 $G^{D_i} =$ **ConstructGraph**(P, D_i, A^*);*//Constructs a graph from P and D_i*
27 **if** (**HasPerfectMatching**$(G^{D_i}) \wedge D_i$.**Radius()**$< \varepsilon$) **then**
28 $\varepsilon = D_i$.**Radius()**;
29 **end if**
30 **end for**
31 **return** ε;

The two subroutines used in Algorithm 3.2.2 are stated in Algorithm 3.2.3 and Algorithm 3.2.4.

Algorithm 3.2.3 Algorithm to construct a graph from the point set P and a disk D.

ConstructGraph(P,D,A^*)
1 $V = P$;
2 $E = \emptyset$;
3 **for all** $a_{i,j} \in A^*$ **do**
4 *//Add an edge connecting vertices p_i and p_j if the midpoint of p_i and p_j is contained in D*
5 **if** D.**CONTAINS**($a_{i,j}$) **then**
6 $E = E \cup \{p_i, p_j\}$;
7 **end if**
8 **end for**

Algorithm 3.2.4 Algorithm to decide, if a graph has a perfect matching.

HasPerfectMatching($G = (V, E)$)
1 Compute maximum matching E_M for the graph G using a matching algorithm;
2 **return** $|E_M| == \frac{|V|}{2}$;

Theorem 3.2.10. *Algorithm 3.2.2 solves the ε-SD computation problem for symmetry group C_2 in $O(n^8\sqrt{n})$ time.*

Proof. In lines 27 and 28 we update the solution value ε if the considered disk contains a set of midpoints corresponding to a partition of P and its radius is smaller than the optimum determined so far. The radius of that disk is the correct solution for the corresponding partition, as we showed in Section 3.2.1. Since we consider all possible disks, we compute the smallest value of ε, so that ε is the radius of the smallest enclosing disk of a set of midpoints corresponding to a partition of P. What remains to be analyzed is the running time. The number of disks that need to be checked is in $O(n^6)$ as proven in Lemma 3.2.8. Constructing a graph for each disk takes $O(n^2)$ time. Computing a maximum matching in a graph $G = (V, E)$ can be done in $O(\sqrt{|V|}|E|)$ time. This was shown by Micali and Vazirani [32]. The total running time of our algorithm is $O(n^6(n^2 + \sqrt{n}n^2)) = O(n^8\sqrt{n})$ □

Dynamic Matching

The running time of Algorithm 3.2.2 can be improved by using a dynamic graph matching algorithm. This is due to the fact that the graphs computed from the $O(n^6)$ disks are not independent but overlap in many edges. Moreover, it is possible

to sort the constructed graphs in such a way that two neighboring graphs differ by at most two edges. When computing the maximum matching for each graph from scratch, too much work is done. A more appropriate way is to use a matching algorithm on dynamic graphs. The idea of dynamic graphs is that a graph is not a static data structure but can be modified by inserting or deleting an edge. A dynamic graph matching algorithm updates the maximum matching after each insert or delete operation. The algorithm for computing a maximum matching in a general graph by Micali and Vazirani [32] can easily be used for dynamic graph matching (see [1]) since it works in phases. In each phase the size of a valid matching is increased if possible.
Consider a dynamic graph $G = (V, E)$ and let M be a maximum matching for G. In the case where an edge is inserted to the graph M remains a valid matching but need not to be maximum anymore. However, the maximum matching in the modified graph can only contain one edge more than M. After performing one phase of the algorithm by Micali and Vazirani [32], the maximum matching for the modified graph is obtained. In the other case where an edge is deleted from G, it also is deleted from M if it was a matching edge. Either M is the maximum matching in the modified graph or before the deletion of the edge there was an other matching which also was maximum and did not contain the deleted edge. M still is a valid matching. Since at most one edge is deleted from M, $|M| \geq |M'| - 1$, where M' is a maximum matching in the modified graph. Thus by performing one phase of the algorithm of Micali and Vazirani [32] a maximum matching for the modified graph can be computed.
In the algorithm of Micali and Vazirani [32] one phase takes $O(|E|)$ time. Maintaining a maximum matching after each insert or delete operation in a dynamic graph therefore takes $O(|E_t|)$ time per update operation where $|E_t|$ is the number of edges in the graph at time t when the operation is executed. Iterating over the $O(n^6)$ disks and using dynamic matching reduces the running time of Algorithm 3.2.2 to $O(n^8)$ since there are $O(n^2)$ edges in the graph at any time. This running time can be assured if the disks are processed in an order which guarantees the edge sets of two graphs defined by two successive disks to differ by at most two edges.
We will construct a global graph which we will traverse in order to obtain a sequence of disks so that the above required property holds. The vertex set of the global graph is the set of centers of the $O(n^6)$ disks we need to examine. The edges are the perpendicular bisectors of the point set over which the disks are defined. This point set is the set of midpoints of pairs of points in P.

Definition 3.2.11. Let a point set $P \subset \mathbb{R}^2$, $|P| = n$ be given.
Let $S = \{\frac{p+p'}{2} | p, p' \in P, p \neq p'\}$ be the set of midpoints defined by pairs of points in P.
We define a graph $\hat{G} = \left(\hat{V}, \hat{E}\right)$.
The vertex set is given by $\hat{V} = \hat{V}_2 \cup \hat{V}_3$, where
$\hat{V}_2 = \{\hat{v}_{\{i,j\}} | s_i, s_j \in S \text{ which are distinct}\}$ and

$\hat{V_3} = \{\hat{v}_{\{h,i,j\}} | s_h, s_i, s_j \in S,$ which are distinct$\}$.
The vertices $\hat{v}_{\{i,j\}}$ and $\hat{v}_{\{h,i,j\}}$ are the centers of the disks defined by s_i, s_j and s_h, s_i, s_j, respectively.
Consider a vertex $\hat{v}_{i,j} \in \hat{V_2}$ and all vertices $\{\hat{v}_{h,i,j} | \hat{v}_{h,i,j} \in \hat{V_3}\}$. All these vertices lie on the perpendicular bisector of s_i and s_j by definition.
The edges of $\hat{G}$ are given by the set $\hat{E}$ which is defined as follows:
$(\hat{v}_{i,j}, \hat{v}_{h,i,j}) \in \hat{E}$ iff $\hat{v}_{i,j}$ and $\hat{v}_{h,i,j}$ are neighbors on the perpendicular bisector defined by s_i and s_j.
$(\hat{v}_{h,i,j}, \hat{v}_{h',i,j}) \in \hat{E}$ iff $\hat{v}_{h,i,j}$ and $\hat{v}_{h',i,j}$ are neighbors on the perpendicular bisector defined by s_i and s_j.

An illustration of $\hat{G}$ is given in Figure 3.2.7

Definition 3.2.12. We call s_h, s_i, s_j the *defining points* of $\hat{v}_{\{h,i,j\}}$ and s_i, s_j the *defining points* of $\hat{v}_{\{i,j\}}$.

Observation 3.2.13. *Let $\hat{e} \in \hat{E}$ be an edge in $\hat{G}$. If $\hat{e} = (\hat{v}, \hat{w})$, then the set of defining points of $\hat{v}$ and $\hat{w}$ differ by exactly one point.*

Lemma 3.2.14. *Let a point set $P \subset \mathbb{R}^2$, $|P| = n$ be given. Consider the global graph as defined in Definition 3.2.11. Let $\hat{v}, \hat{w} \in \hat{V}$ be two adjacent vertices. Let $D_{\hat{v}}$ and $D_{\hat{w}}$ be the two disks defined by $\hat{v}$ and $\hat{w}$, respectively. Let $S_{D_{\hat{v}}} = S \cap D_{\hat{v}}$ and $S_{D_{\hat{w}}} = S \cap D_{\hat{w}}$. Then $|S_{D_{\hat{v}}} \triangle S_{D_{\hat{w}}}| \leq 2$.*

Proof. We first show that the points of $S_{D_{\hat{v}}} \cup S_{D_{\hat{w}}}$ which are not in $S_{D_{\hat{v}}} \cap S_{D_{\hat{w}}}$ are either on the boundary of $D_{\hat{v}}$ or on the boundary of $D_{\hat{w}}$.
Let $s \in S$ be a point in $S_{D_{\hat{v}}}$ and let $\hat{v}$ be defined by three points $s_h, s_i, s_j \in S$ all not equal to s. So s lies not on the boundary of $D_{\hat{v}}$. The perpendicular bisectors of the line segments $\overline{ss_h}, \overline{ss_i}, \overline{ss_j}$ each partition the plane into two half-planes. Since s lies in $S_{D_{\hat{v}}}$, it lies in the same half-plane as $\hat{v}$ for each perpendicular bisector.
Suppose s lies in $S_{D_{\hat{v}}}$ but not in $S_{D_{\hat{w}}}$. Suppose w.l.o.g. that the edge $(\hat{v}, \hat{w})$ is the perpendicular bisector of the line segment $\overline{s_i s_j}$. Thus s_i and s_j both lie on the boundary of $D_{\hat{v}}$ as well as on the boundary of $D_{\hat{w}}$. If $s \notin P_{D_{\hat{w}}}$, the perpendicular bisectors of the two line segments $\overline{ss_i}$ and $\overline{ss_j}$ must be crossed while traversing from $\hat{v}$ to $\hat{w}$ on the perpendicular bisector of $\overline{s_i s_j}$. Since $\hat{w}$ is adjacent to $\hat{v}$, $\hat{w}$ needs to be the intersection of perpendicular bisectors of the three line segments $\overline{ss_i}, \overline{ss_j}$ and $\overline{s_i s_j}$. This implies that s is on the boundary of $D_{\hat{w}}$ and therefore $s \in S_{D_{\hat{w}}}$. This is a contradiction to our assumption. Thus s is on the boundary of $D_{\hat{v}}$.
By the definition of $\hat{G}$ the set of points which define two adjacent vertices differ by at most two points. Thus the difference between the points of S contained in $D_{\hat{v}}$ and $D_{\hat{w}}$ is at most two. The case where $\hat{v}$ is defined by two points in S and $\hat{w}$ is defined by three points in S follows analogously. □

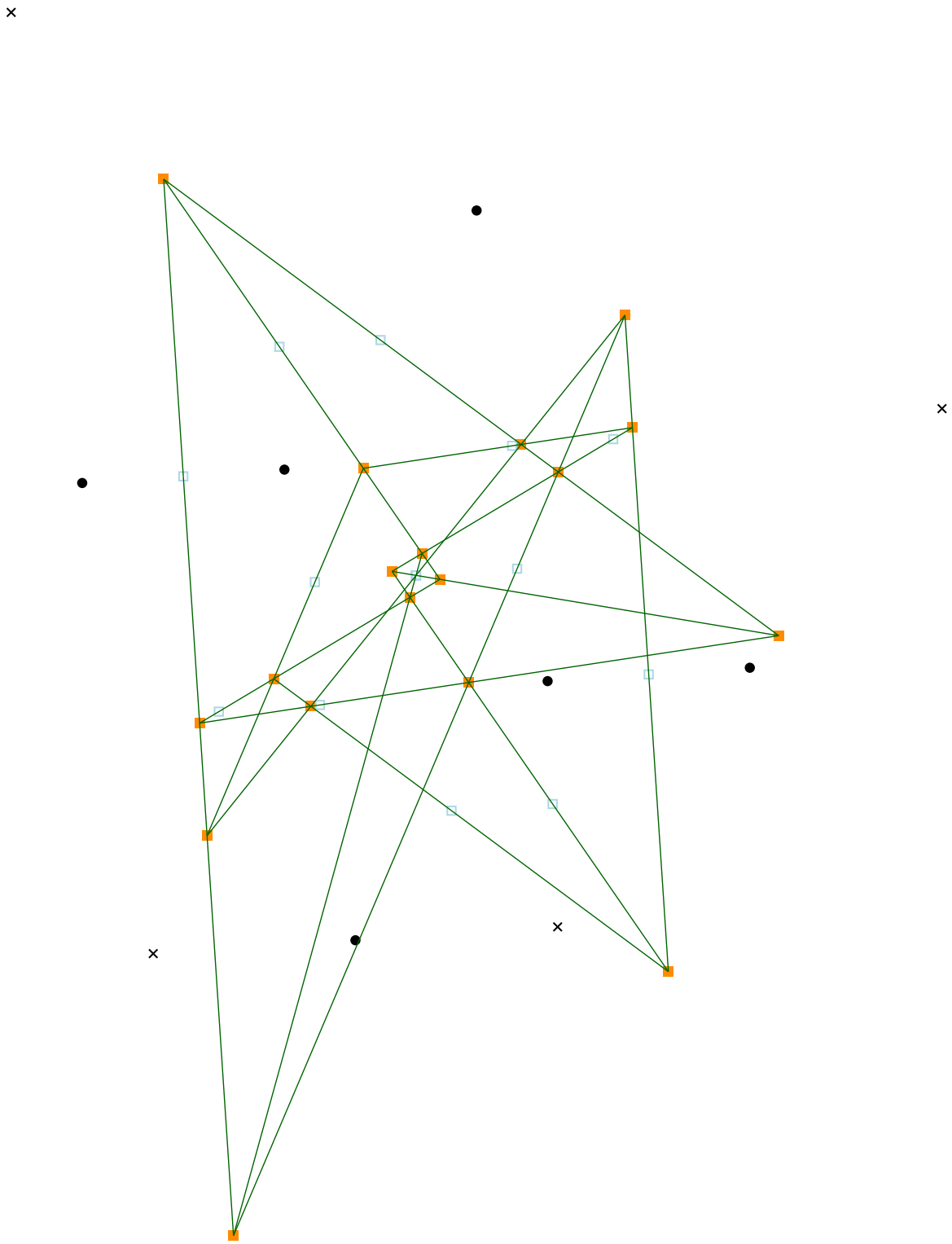


Figure 3.2.7: An illustration of a global graph. The point set P is illustrated by crosses, the point set S is illustrated by circles, the vertex set $\hat{V}_2$ is illustrated by blue squares and the vertex set $\hat{V}_3$ is illustrated by orange boxes.

Theorem 3.2.15. *Let a point set $P \subset \mathbb{R}^2$ be given. The solution of the ε-SD computational problem for symmetry group C_2 can be determined in $O(n^8)$ time.*

Proof. The point set S can be computed in $O(n^2)$ time. The global graph $\hat{G}$ consists of the $O(n^6)$ vertices which can be computed from the point set S in $O(n^6)$ time. The edges are given by the $O(n^4)$ perpendicular bisectors of the line segments defined by pairs of points in S. For each vertex its location on the perpendicular bisectors given by the defining points of the vertex can be computed in constant time and all vertices on one perpendicular bisector need to be sorted according to their location in order to obtain the edge set of $\hat{G}$. Thus $\hat{G}$ can be constructed in time $O(n^6 \log n)$. Each vertex represents a disk. The maximum matching of the graph defined by the points of S contained in that disk is stored in the vertex. By traversing $\hat{G}$ with breadth-first-search the matching is updated by performing at most two phases of the algorithm of Micali and Vazirani [32]. By Lemma 3.2.14 it is guaranteed, that at most two phases are sufficient since the set of edges of two graphs defined by two neighboring disks in the global graph differ by at most two edges. For each vertex in the global graph at most two phases of the matching algorithm are performed. Thus the overall running time is $O(n^8)$. □

3.3 The ε-SD Problem for Symmetry Group $D_{|P|}$

In this section we will investigate the variation of the ε-SD problem where the symmetry group is dependent on the number of input points. We subdivide this section into two subsections. In the first subsection we will consider the ε-SD computation problem for symmetry group $D_{|P|}$ and additional knowledge of the rotation center c. In the second subsection we consider the ε-SD decision problem for symmetry group $D_{|P|}$ with no further restrictions. In each subsection we will develop the algorithms solving the problems considered and give the proofs on the correctness and running time of the algorithms. For the rest of this section, let $n = |P|$ be the size of the input set P.
The results of this section are joint work with Lena Schlipf and are part of the work published in [5].

3.3.1 Rotation Center Known

We start by considering the ε-SD computational problem for symmetry group D_n in the case where the rotation center c of the D_n-symmetric point set Q is given beforehand.

We define the problem as in Section 3.1:

Problem 3.1.6 (The ε-SD computational problem for symmetry group D_n and rotation center c.)

Given: A point set $P \subset \mathbb{R}^2$, $|P| = n$ and a rotation center c.

Task: Compute the smallest value $\varepsilon \geq 0$ so that there is a D_n-symmetric point set Q with rotation center c which ε-approximates P.

Since we ask for a D_n-symmetric point set Q in this section, $|Q| = n$ and therefore Q is a regular n-gon.
We can restrict the problem further by adding an order on the input points. By doing so, we force the bijection to map a point in P to a certain vertex of the regular n-gon. We extend Definition 3.1.1 to the case where an order on P is given:

Definition 3.3.1. Let a point set P, an order $(p_1, \ldots, p_n)$ on P, a symmetry group D_n and a value $\varepsilon \geq 0$ be given. A D_n-symmetric point set Q *ε-approximates P with respect to the given order* iff there is a bijection $f : P \to Q$, so that $d(p, f(p)) \leq \varepsilon$ and $f(p_i) = \rho_c^{\frac{2\pi}{k}}\left(f(p_{(i-1)})\right)$ for all $1 \leq i \leq n$, where we define $p_0 := p_n$.

Adding the information of the order on P leads to the following definition of the ε-SD problem:

Problem 3.3.2 (The ε-SD computational problem for symmetry group D_n with given order and rotation center c)

Given: A point set $P \subset \mathbb{R}^2$, $|P| = n$, an order $(p_1, \ldots, p_n)$ on P and a rotation center c.

Task: Compute the smallest value $\varepsilon \geq 0$ so that there is a D_n-symmetric point set Q with rotation center c which ε-approximates P with respect to the order of P.

We start by solving the ε-SD computational problem for symmetry group D_n with given rotation center and order on P. We will use this algorithm later on in order to solve the ε-SD for symmetry group D_n and given rotation center where the order on P is not known.

Lemma 3.3.3. *Let a point set P, an order $(p_1, \ldots, p_n)$ on P and a rotation center c be given.*
Let $\alpha = \frac{2\pi}{n}$ be the rotation angle defined by the symmetry group D_n.
Let $r_i = \rho_c^{(i-1)\alpha}(p_i)$, $1 \leq i \leq n$ be the point we get when rotating p_i around c by $(i-1)\alpha$ in counterclockwise direction.
Let q_1 be the center of the smallest enclosing disk D of the point set $\{r_i | 1 \leq i \leq n\}$ and $q_i = \rho_c^{-(i-1)\alpha}(q_1)$, $2 \leq i \leq n$.
Let ε be the radius of D.
Then ε is the solution of the ε-SD computational problem for symmetry group D_n with given rotation center c and given rotational order on the input set P. The point set $Q = \{q_1, \ldots, q_n\}$ is the corresponding D_n-symmetric point set ε-approximating P with respect to the rotational order on P.

Proof. For a given point set $P = \{p_1, \ldots, p_n\}$ and a D_n-symmetric point set $Q = \{q_1, \ldots, q_n\}$ with rotation center c, the smallest possible value ε, so that Q ε-approximates P, is $\varepsilon = \max\{d(p_i, q_i) | 1 \leq i \leq n\}$ by definition. Rotating the point p_i around c counterclockwise by the angle $(i-1)\alpha$ yields the point r_i as defined above. The corresponding point q_i will be mapped to the point q_1 when applying the same rotation to it as for p_i. Therefore minimizing the distance between p_i and q_i corresponds to minimizing the distance between r_i and q_1.

Thus, computing a D_n-symmetric point set Q which minimizes ε corresponds to computing a single point q_1 minimizing $\max\{d(r_i, q_1) | 1 \leq i \leq n\}$.

For n given points, the point minimizing the maximal distance to all n points is the center of the smallest enclosing disk.

Let ε be the radius of the smallest enclosing disk, then

$$\begin{aligned} d(r_i, q_1) \leq \varepsilon &\Leftrightarrow d(\rho_c^{-(i-1)\alpha}(r_i), \rho_c^{-(i-1)\alpha}(q_1)) \leq \varepsilon \\ &\Leftrightarrow d(p_i, q_i) \leq \varepsilon \end{aligned}$$

Therefore ε is the solution of the ε-SD computational problem for symmetry group D_n with given rotation center c and order on P. $Q = \{q_1, \ldots, q_n\}$ is the corresponding D_n-symmetric point set which ε-approximates P with respect to the order on P. □

For a visualization of this proof see Figure 3.3.1

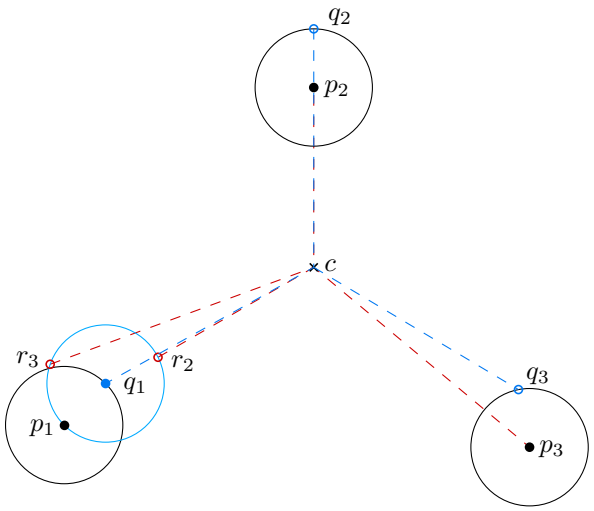


Figure 3.3.1: Illustration of the proof of Lemma 3.3.3 for symmetry group D_3: The red points are the points of $P = \{p_1, p_2, p_3\}$ rotated corresponding to the rotational order (p_1, p_2, p_3) on P. The smallest enclosing disk D of the points p_1, r_2, r_3 is colored blue. The point q_1 of the C_3-symmetric point set Q is the blue center of D. The remaining blue points are the points of Q defined by q_1 and c.

Theorem 3.3.4. *Algorithm 3.3.1 solves the ε-SD computational problem for symmetry group D_n with given rotation center c and order $(p_1, \ldots, p_n)$ on P in linear time.*

Proof. The correctness follows directly from Lemma 3.3.3. What remains to be done is to analyze the running time. Computing the sets R and Q takes time $O(n)$ since n points are computed for each set and the computation of each point can be done in constant time. The smallest enclosing disk of n points can be computed in linear time using an algorithm given by Megiddo [30]. This results in an $O(n)$ overall running time. □

Algorithm 3.3.1 Algorithm for the ε-SD computational problem for symmetry group D_n with given rotation center and given order on P.

SDD$_\mathbf{n}$OrderAndCenter($P = (p_1, \ldots, p_n)$, c)
  //Compute the rotation angle α for symmetry group D_n.
  $\alpha = \frac{2\pi}{n}$;
  //Compute the set of rotated points of P according to the order on P.
  $R = \emptyset$;
  for $i = 1 \ldots n$ **do**
    $r_i = \rho_c^{(i-1)\alpha}(p_i)$;
    $R = R \cup \{r_i\}$;
  end for
  $D =$ **SmallestEnclosingDisk**(R);
  $q_1 = D$.**Center()**;
  //Compute the set Q from q_1 and c.
  $Q = \emptyset$;
  for $i = 1 \ldots n$ **do**
    $q_i = \rho_c^{-(i-1)\alpha}(q_1)$;
    $Q = Q \cup \{q_i\}$;
  end for
  //The radius of D is the solution of the ε-SD problem.
  $\varepsilon = D$.**Radius()**;
  return ε;

One reason why we are able to solve the ε-SD computational problem for symmetry group D_n in linear time is the knowledge of the rotational order on P. As stated above, this order gives us the mapping between P and the D_n-symmetric point set Q. We therefore know which angle to apply to each point $p_i \in P$ in order to compute the set R. Different orders lead to different D_n-symmetric point sets Q and different values ε. If no order on P is given, the question is to find the order on P which results in the smallest value of ε.

Problem 3.1.6 (The ε-SD computational problem for symmetry group D_n and rotation center c)

Given: A point set $P \subset \mathbb{R}^2$ and a rotation center c.

Task: Compute the smallest value $\varepsilon \geq 0$ so that there is a D_n-symmetric point set Q with rotation center c which ε- approximates P.

The polynomial time algorithm to solve this problem is more or less the same as the one where the rotational order on the points is given. However, we need to find a way to determine the order on P which results in the smallest value of ε.

One simple way would be to consider all possible rotational orders on each subset. Unfortunately, this is equivalent to the number of permutations of n numbers since we need to compute all possibilities to map the n rotation angles in $\Lambda = \{i\alpha | \alpha = \frac{2\pi}{n}, 0 \leq i \leq n-1\}$ to the points in P. The number of permutations of n numbers, however, is $n!$ which is not polynomial in n. Thus, this approach is not helpful in order to find a polynomial time algorithm.

In Algorithm 3.3.1 we computed the smallest enclosing disk of the rotated points in order to determine the optimal value of ε. We generated these points by rotating each point in P by an angle defined by the rotational order on P.

Since we do not know the rotational order of the points, we do not know which angle in Λ to apply to which point in P. We therefore compute $n(n-1) = \Theta(n^2)$ points by rotating each of the n points in P by each of the n angles in Λ. By $r_{(i,l)} = \rho_c^{l\alpha}(p_i)$ we denote the point generated by rotating p_i around c with angle $l\alpha \in \Lambda$.

Let $\mathcal{R} = \{r_{i,l} | 1 \leq i \leq n, 0 \leq l \leq n-1\}$ be the set of all rotational images of points in P with respect to angles in Λ. Rotated points corresponding to one special order on P are a set $R \subset \mathcal{R}$, so that each point in P is represented by exactly one point in R, thus $r_{(i,l)}, r_{(i',l')} \in R \Leftrightarrow i \neq i' \wedge l \neq l'$ and $|R| = |P|$. Each set R with these properties defines a rotational order on P. For this order we can compute the value of ε by taking the radius of the smallest enclosing disk of R as stated in Algorithm 3.3.1.

Since a smallest enclosing disk is defined by either two or three points, we compute $\Theta(n^4)$ disks defined by two points and $\Theta(n^6)$ disks defined by three points from the $\Theta(n^2)$ points in $\mathcal{R}$.

Each disk D we consider contains a subset $\tilde{R} \subseteq \mathcal{R}$ of rotated points. We need to check if there is a set $R \subseteq \tilde{R}$, $|R| = n$, so that the labels define a bijection between the points in P and the angles in Λ. We use a matching algorithm on bipartite graphs to solve this problem:

Lemma 3.3.5. *Let a point set P, $|P| = n$ and a rotation center c be given. Let $\alpha = \frac{2\pi}{n}$ and $\mathcal{R} = \{r_{(i,l)} = \rho_c^{l\alpha}(p_i) \,|\, 1 \leq i \leq n, 0 \leq l \leq n-1\}$ be the set of rotated points of P. Let D be a disk defined by two or three points in $\mathcal{R}$ and let $\tilde{R} \subset \mathcal{R}$ be the points of $\mathcal{R}$ contained in D. We can decide in time $O(M(n))$ if $\tilde{R}$ contains a subset of n points, so that the labels of these points describe a bijection between points in P and angles in Λ, and therefore an order on P. $M(n)$ is the time needed for the computation of a perfect matching in a bipartite graph with $O(n)$ vertices.*

Proof. We reduce the problem of finding a subset of n points in $\tilde{R}$, so that the labels of these points define a bijection between points in P and angles in Λ to the problem of finding a perfect matching in a bipartite graph. Since the labels represent the points in P on the one hand and the angles in Λ on the other hand, the two vertex sets of the bipartite graph are the sets P and Λ. For each point $r_{(i,l)} \in \tilde{R}$ we add an edge between the vertices p_i and $\alpha_l = l\alpha$. This graph has a perfect matching

iff exactly n edges exist, so that each vertex is incident to exactly one edge. This is equivalent to a bijection between points in P and angles in Λ. We therefore can solve the problem of deciding if there exists a set $R \subseteq \tilde{R}$, so that the labels define a bijective mapping between P and Λ in $O(M(n))$ time. If such a perfect matching exists, we get the rotational order on the points from the edges in the graph since they model the correspondence between points and angles. □

Figure 3.3.2 gives an illustration of the proof. The different shapes illustrate the points whereas the different colors are used to distinguish between the rotation angles. All rotational images of points with respect to the rotation angle π are colored blue. The labels of the points are as described above. In the example in Figure 3.3.2 we consider the symmetry group D_4, and therefore a point set containing four points is depicted. Also, one of the possible $\Theta(n^6)$ disks is shown. Below the point set we see on the left hand side the graph with all edges corresponding to the points in the circle. On the right hand side the matching edges are highlighted.
Using this matching algorithm we can adapt Algorithm 3.3.1 to the situation where the rotational order is not known and get Algorithm 3.3.2.

Algorithm 3.3.2 Algorithm for the ε-SD computational problem for symmetry group D_n with given rotation center c.

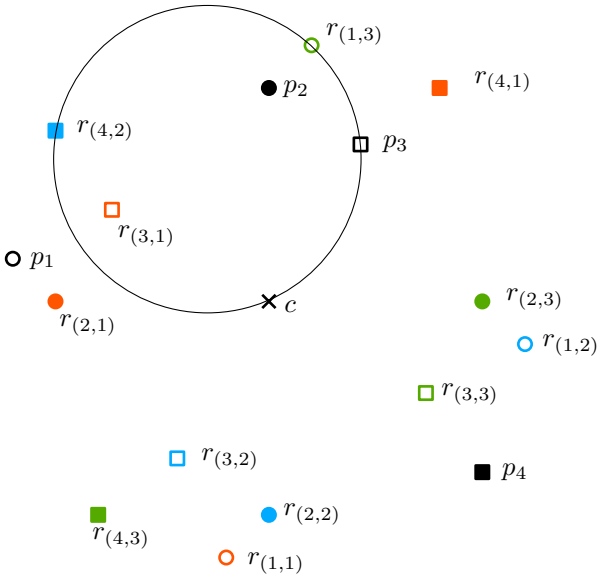
SDD$_\mathbf{n}$Center($P = \{p_1, \ldots, p_n\}$, c)

1 Compute the set $\mathcal{R}$ as defined above.
2 Compute all disks defined by two or three points in $\mathcal{R}$.
3 For each disk D compute the set $\tilde{R} = \{r_{(i,l)} \in \mathcal{R} | r_{(i,l)}$ is contained in $D\}$.
4 Construct bipartite graph $G = (P \cup \Lambda, E)$, where $E = \{\{p_i, \alpha_l\} | r_{(i,l)} \in \tilde{R}\}$.
5 Use a matching algorithm for bipartite graphs to check, if G contains a perfect matching.
6 If this is the case, compare the radius r of D to the current value of ε. If $r < \varepsilon$, set $\varepsilon = r$.
7 After computing and checking all the disks, ε is the solution of the ε-SD computational problem for symmetry group D_n with given rotation center c.

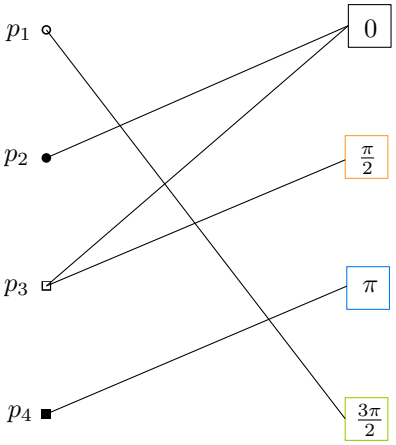
We complete this section by proving the correctness and running time of Algorithm 3.3.2.

Lemma 3.3.6. *Algorithm 3.3.2 solves the ε-SD computational problem for symmetry group D_n and with given rotation center c in $O(n^6 \cdot M(n))$, where $M(n)$ is the time needed to decide if a bipartite graph with n vertices has a perfect matching.*

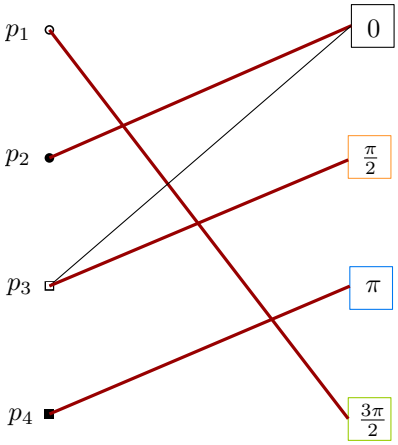
Proof. The correctness follows from Lemma 3.3.5. What remains to be done is to analyze the running time of Algorithm 3.3.2. The set $\mathcal{R}$ contains $\Theta(n^2)$ points. Since each disk considered is defined by at most three points of $\mathcal{R}$, we need to consider $\Theta(n^6)$ disks in total. For each disk we need to construct a bipartite graph which takes $O(n^2)$ time. □



(a) A disk D containing a subset of the rotated points of P.



(b) The bipartite graph G_D corresponding to the points in D.



(c) A perfect matching in G_D.

Figure 3.3.2: Illustration of the proof of Lemma 3.3.5.

Computing a maximum matching in bipartite graphs can be done in $O\left(\sqrt{|V_1|+|V_2|}|E|\right)$ time by an algorithm given by Hopcroft and Karp [15] or in $O\left(\sqrt{\frac{|V|^3|E|}{\log|V|}}\right)$ time by an algorithm given by Alt et al. [3].
Another possibility to decide if the center of the computed disk D is the vertex of the optimal regular n-gon is to use the bottleneck distance. For two point sets A and B in the plane, the bottleneck distance minimizes the maximum distance between the points in A and B, where the mapping between A and B is a bijection.

The bottleneck distance can be computed in time $O(\sqrt{n}n \log n)$ using an algorithm by Efrat et al. [13].

We can replace the matching algorithm in Algorithm 3.3.2 by using the bottleneck distance. For each disk $D(r, q)$ defined by two or three points of the set $\mathcal{R}$ we can check if the regular n-gon defined by the given rotation center c and q r-approximates P by computing the bottleneck distance between P and the vertex set of the regular n-gon and checking if it is smaller or equal to r.

Thus we can modify Algorithm 3.3.2 as follows:

Algorithm 3.3.3 Algorithm for the ε-SD computational problem for symmetry group D_n with given rotation center c.

SDD$_\mathbf{n}$Center($P = \{p_1, \ldots, p_n\}$, c)
1 Compute the set $\mathcal{R}$ as defined above.
2 Compute all disks defined by two or three points in $\mathcal{R}$.
3 For each disk $D = (r, q)$ compute the regular n-gon Q_D defined by c and q.
4 b_D=**BottleneckDistance**(P, Q_D);
5 **if** $b_D \leq r$ and $r \leq \varepsilon$ **then**
6 $\varepsilon = r$;
7 **end if**

Lemma 3.3.7. *Algorithm 3.3.3 solves the ε-SD computational problem for symmetry group D_n and with given rotation center c in $O(n^7\sqrt{n} \log n)$ time.*

Proof. The correctness arguments still hold and the running time of Algorithm 3.3.3 is then $O(n^6\sqrt{n}n \log n) = O(n^7\sqrt{n} \log n)$. □

Remark 3.3.8. The running time of Algorithm 3.3.3 is better than the running time of Algorithm 3.3.2, since it is always possible that the bipartite graph contains n^2 edges. Nevertheless we state both results because of the following reasons:

1. Not all graphs constructed by Algorithm 3.3.2 contain n^2 edges, a lot of those graphs will only contain $O(n)$ edges, thus a more careful analysis might result in a better worst-case running time for Algorithm 3.3.2.

2. In Algorithm 3.3.2, the decision whether the center of a considered disk is a vertex of the regular n-gon which ε-approximates the point set P is made only on P. We need neither to compute the vertex nor the whole point set Q as it is done by Algorithm 3.3.3. One could also think of improving the running time by using a dynamic approach as for the algorithm considering symmetry group C_2. For the matching approach, dynamic matching can be used, since the graph only changes in a constant number of edges, when passing from one disk to the next. For the bottleneck distance approach, for each disk a completely different point set Q needs to be computed which makes a dynamic approach almost impossible.

In the following algorithms we will also explain and analyze both approaches, due to the above given arguments.

Theorem 3.3.9. *Let a point set $P \subset \mathbb{R}^2, |P| = n$ and a rotation center $c \in \mathbb{R}^2$ be given. The ε-SD computational problem for symmetry group D_n with respect to c can be solved in $O(n^7\sqrt{n}\log n)$ time by using the bottleneck distance and in $O(n^6 \cdot M(n))$ time, where $M(n)$ is the time needed to decide if a bipartite graph with n vertices has a perfect matching, by using matching in bipartite graphs.*

Proof. The theorem follows from Lemma 3.3.7 and Lemma 3.3.6. □

3.3.2 Rotation Center c Not Known

In this section we consider the ε-SD decision problem for symmetry group D_n. In contrast to the last section, we are not given the rotation center of the D_n-symmetric point set Q which ε-approximates P. The problem we investigate in this section is defined as follows:

Problem 3.1.5 (The ε-SD decision problem for symmetry group D_n)

Given: A point set $P \subset \mathbb{R}^2$ and a value $\varepsilon \geq 0$.

Question: Is there a D_n-symmetric point set Q which ε- approximates P?

Since we do not know the rotation center of the D_n-symmetric point set Q, we need to compute it from the given point set P.

Again it is easier to consider the case where the rotational order on P is given:

Problem 3.3.10 (The ε-SD decision problem for symmetry group D_n with given rotational order)

Given: A point set $P \subset \mathbb{R}^2$, a rotational order $(p_1, \ldots, p_n)$ on P and a value $\varepsilon \geq 0$.

Question: Is there a D_n-symmetric point set Q and an order $(q_1, \ldots, q_n)$ so that $d(p_i, q_i) \leq \varepsilon$, for all $1 \leq i \leq n$?

Problem 3.3.10 is a decision problem, we can therefore think of it as of finding a regular n-gon $Q = (q_1, \ldots, q_n)$, so that q_i lies in the disk D_i with center p_i and radius ε, for all $i \in \{1, \ldots, n\}$. Suppose such a regular n-gon exists, then the following holds:

Lemma 3.3.11. *Let a point set $P = (p_1, \ldots, p_n)$, a rotational order on P and a value $\varepsilon \geq 0$ be given. Let $Q = (q_1, \ldots, q_n)$ be a regular n-gon with rotation center c, so that $d(p_j, q_j) \leq \varepsilon$. Let furthermore D_j be the disk with center p_j and radius ε f.a $1 \leq j \leq n$. Then $D_1 \cap \rho_c^{(j-1)\alpha}(D_j) \neq \emptyset$ for all $1 \leq j \leq n$, where $\alpha = \frac{2\pi}{n}$.*

Proof. Let $r_j = \rho_c^{(j-1)\alpha}(p_j)$ be the rotational image of p_j with respect to the rotation center c and angle $(j-1)\alpha$. D_1 and $\rho_c^{(j-1)\alpha}(D_j)$ intersect, iff $d(p_1, r_j) \leq 2\varepsilon$.

$$\begin{aligned} d(p_1, r_j) &\leq d(p_1, q_1) + d(q_1, r_j) \\ &= d(p_1, q_1) + d(\rho_c^{-(j-1)\alpha}(q_1), \rho_c^{-(j-1)\alpha}(r_j)) \\ &= d(p_1, q_1) + d(q_j, p_j) \\ &\leq \varepsilon + \varepsilon \\ &= 2\varepsilon \end{aligned}$$

□

Lemma 3.3.11 gives us a hint where to find the unknown rotation center c. For each pair (p_1, p_j) of ordered points we need to find the region of points so that rotating the disk D_j around one point in that region by $(j-1)\alpha$ results in a disk which intersects D_1. We call that region the *apex region* of D_1 and D_j. We will see in the oncoming algorithms and proofs that it does not suffice to consider the apex regions of two disks but that we need to consider the apex regions defined by a point and a disk.
We will use the following definitions:

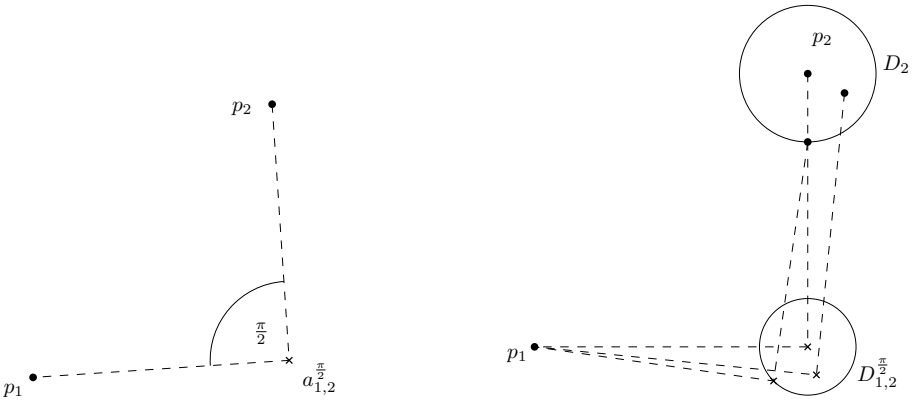
Definition 3.3.12 (Apex Point and Apex Disk)**.**

1. For two points p_i and p_j and an angle α the *apex point* of p_i and p_j with respect to α is the point $a_{i,j}^{\alpha}$, so that $p_i = \rho_{a_{i,j}^{\alpha}}^{\alpha}(p_j)$.

2. For a point p_i and a disk D_j the *apex disk* of p_i and D_j with respect to α is the set $D_{i,j}^{\alpha} = \{a_{i,j}^{\alpha} | a_{i,j}^{\alpha}$ is the apex point of p_i and $p_j \in D_j\}$.

For illustration see Figure 3.3.3.

In Lemma 3.3.13 we will show how to compute the apex point of two points with respect to an angle α. In Lemma 3.3.15 we will prove the apex region defined by a point, a disk and an angle α to be a disk and show how to compute its center and radius.



(a) Apex point defined by p_1, p_2 and $\alpha = \frac{\pi}{2}$.

(b) Apex disk defined by p_1, D_2 and $\alpha = \frac{\pi}{2}$.

Figure 3.3.3: Illustration of Definition 3.3.12

Lemma 3.3.13. *The apex point of two points p_i and p_j is the unique point* $a^{\alpha}_{i,j} = \left(\frac{(p^y_j - p^y_i)}{2\tan\left(\frac{\alpha}{2}\right)} + \frac{1}{2}(p^x_i + p^x_j), \frac{(p^x_i - p^x_j)}{2\tan\left(\frac{\alpha}{2}\right)} + \frac{1}{2}(p^y_i + p^y_j) \right)$

Proof. In the following computations we will denote the apex point by a.
By the definition of the apex point, rotating p_j around a yields p_i. So

$$\begin{pmatrix} p^x_i \\ p^y_i \end{pmatrix} = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \begin{pmatrix} p^x_j - a^x \\ p^y_j - a^y \end{pmatrix} + \begin{pmatrix} a^x \\ a^y \end{pmatrix}$$

from this definition for the rotation around a, we get the following system of equations:

$$\begin{aligned} p^x_i &= (p^x_j - a^x)\cos\alpha - (p^y_j - a^y)\sin\alpha + a^x && \text{(I)} \\ p^y_i &= (p^x_j - a^x)\sin\alpha + (p^y_j - a^y)\cos\alpha + a^y && \text{(II)} \end{aligned}$$

Sorting both equations, so that terms containing parts of the apex point a are on the right side and all others are on the left side leads to the following system of equations:

$$\begin{aligned} p^x_i - p^x_j\cos\alpha + p^y_j\sin\alpha &= a^x(1-\cos\alpha) + a^y\sin\alpha && \text{(III)} \\ p^y_i - p^x_j\sin\alpha - p^y_j\cos\alpha &= a^y(1-\cos\alpha) - a^x\sin\alpha && \text{(IV)} \end{aligned}$$

We eliminate a^x by computing (V) $= (\sin\alpha)$(III) $+ (1-\cos\alpha)$(IV):

$$\begin{aligned} & p_i^x \sin\alpha - p_j^x(\sin\alpha\cos\alpha + \sin\alpha(1-\cos\alpha)) \\ & +p_i^y(1-\cos\alpha) + p_j^y(\sin^2\alpha - (1-\cos\alpha)\cos\alpha) \\ & = \\ & a^y(\sin^2\alpha + (1-\cos\alpha)^2) \end{aligned} \quad \text{(V)}$$

Which can be reformed to:

$$p_i^x \sin\alpha + (1-\cos\alpha)p_i^y - p_j^x \sin\alpha + p_j^y(1-\cos\alpha) = 2a^y(1-\cos\alpha) \quad \text{(VI)}$$

Therefore,

$$a^y = \frac{(p_i^x - p_j^x)\sin\alpha}{2(1-\cos\alpha)} + \frac{p_i^y + p_j^y}{2}$$

Applying the value of a^y to (II) :

$$\begin{aligned} a^x \sin\alpha &= p_j^x \sin\alpha + p_j^y \cos\alpha - p_i^y + a^y(1-\cos\alpha) \\ &= p_j^x \sin\alpha + p_j^y \cos\alpha - p_i^y + \frac{1}{2}((p_i^x - p_j^x)\sin\alpha + (p_i^y + p_j^y)(1-\cos\alpha)) \\ &= \frac{1}{2}(\sin\alpha(p_i^x + p_j^x) + (1+\cos\alpha)(p_j^y - p_i^y)) \end{aligned}$$

and therefore,

$$a^x = \frac{(1+\cos\alpha)(p_j^y - p_i^y)}{2\sin\alpha} + \frac{(p_i^x + p_j^x)}{2}$$

What remains to be shown is $\frac{\sin\alpha}{(1-\cos\alpha)} = \frac{1}{\tan\left(\frac{\alpha}{2}\right)}$ and $\frac{(1+\cos\alpha)}{\sin\alpha} = \frac{1}{\tan\left(\frac{\alpha}{2}\right)}$

$$\begin{aligned} \frac{\sin\alpha}{(1-\cos\alpha)} &= \frac{2\sin\left(\frac{\alpha}{2}\right)\cos\left(\frac{\alpha}{2}\right)}{\cos^2\left(\frac{\alpha}{2}\right) + \sin^2\left(\frac{\alpha}{2}\right) - \cos^2\left(\frac{\alpha}{2}\right) + \sin^2\left(\frac{\alpha}{2}\right)} \\ &= \frac{2\sin\left(\frac{\alpha}{2}\right)\cos\left(\frac{\alpha}{2}\right)}{2\sin^2\left(\frac{\alpha}{2}\right)} \\ &= \frac{\cos\left(\frac{\alpha}{2}\right)}{\sin\left(\frac{\alpha}{2}\right)} \\ &= \frac{1}{\tan\left(\frac{\alpha}{2}\right)} \end{aligned}$$

$$\begin{aligned}\frac{(1+\cos\alpha)}{\sin\alpha} &= \frac{\cos^2\left(\frac{\alpha}{2}\right)+\sin^2\left(\frac{\alpha}{2}\right)+\cos^2\left(\frac{\alpha}{2}\right)-\sin^2\left(\frac{\alpha}{2}\right)}{2\sin\left(\frac{\alpha}{2}\right)\cos\left(\frac{\alpha}{2}\right)}\\ &= \frac{2\cos^2\left(\frac{\alpha}{2}\right)}{2\sin\left(\frac{\alpha}{2}\right)\cos\left(\frac{\alpha}{2}\right)}\\ &= \frac{\cos\left(\frac{\alpha}{2}\right)}{\sin\left(\frac{\alpha}{2}\right)}\\ &= \frac{1}{\tan\left(\frac{\alpha}{2}\right)}\end{aligned}$$

□

The distance between two apex points can be related to the distance between the points defining those apex points.

Lemma 3.3.14. *Let three points p_1, p_2, p_3 and an angle α be given. Let $a^\alpha_{1,2}$ be the apex point defined by p_1, p_2 and α. Let $a^\alpha_{1,3}$ be the apex point defined by p_1, p_3 and α. Let $d = d(a^\alpha_{1,2}, a^\alpha_{1,3})$ denote the Euclidean distance between $a^\alpha_{1,2}$ and $a^\alpha_{1,3}$.*
Then $d(p_2, p_3) = d\sqrt{2(1-\cos\alpha)} = 2d\sin\left(\frac{\alpha}{2}\right)$

Proof. Iwanowski [18] proved in Lemma 2.4.6 of his thesis
$d(p_2, p_3) = d\sqrt{2(1-\cos\alpha)}$.
That implies

$$\begin{aligned}d(p_2,p_3) &= d\sqrt{2(1-\cos\alpha)}\\ &= d\sqrt{2\left(\cos^2\left(\frac{\alpha}{2}\right)+\sin^2\left(\frac{\alpha}{2}\right)-\cos^2\left(\frac{\alpha}{2}\right)+\sin^2\left(\frac{\alpha}{2}\right)\right)}\\ &= d\sqrt{2\left(2\sin^2\left(\frac{\alpha}{2}\right)\right)}\\ &= 2d\sin\left(\frac{\alpha}{2}\right)\end{aligned}$$

□

Using Lemma 3.3.14 we can prove the apex region defined by a point p, a disk D and an angle α to be a disk.

Lemma 3.3.15. *Let a point $p_1 \in \mathbb{R}^2$ a disk $D_2 \subset \mathbb{R}^2$ and an angle α be given. Let p_2 be the center of D_2 and let r be its radius. Then the apex region defined by p_1, D_2 and α is a disk $D^\alpha_{1,2}$ with center $a^\alpha_{1,2}$ and radius $\frac{r}{2\sin\left(\frac{\alpha}{2}\right)}$, where $a^\alpha_{1,2}$ is the apex point defined by p_1, p_2 and α.*

Proof. Let $\tilde{p}$ be a point in the disk D_2, so $d(p_2, \tilde{p}) \leq r$. Let a^α denote the apex point defined by $p_1, \tilde{p}$ and α. Then $d(a^\alpha_{1,2}, a^\alpha) = \frac{d(p_2,\tilde{p})}{2\sin\left(\frac{\alpha}{2}\right)} \leq \frac{r}{2\sin\left(\frac{\alpha}{2}\right)}$ by Lemma 3.3.14.

On the other hand, let a^α be a point in $D^\alpha_{1,2}$. Then there is a unique point $\tilde{p} \in \mathbb{R}^2$ so that a^α is the apex point of $\tilde{p}$, p_1 and α, by Lemma 3.3.13. What remains to be shown is $\tilde{p} \in D_2$. By Lemma 3.3.14 $d(\tilde{p}, p_2) = d(a^\alpha_{1,2}, a^\alpha) 2\sin\left(\frac{\alpha}{2}\right) \leq 2\sin\left(\frac{\alpha}{2}\right) \frac{r}{2\sin\left(\frac{\alpha}{2}\right)} = r$. □

By the above Lemma 3.3.15 it is clear that for an angle α and a point $c \in D^\alpha_{1,2}$ the rotational image of D_2 with respect to the rotation center c and angle α contains the point p_1. So points in the apex region seem to be good candidates for the rotation center of Q.

The answer of the ε-SD decision problem for symmetry group D_n is simpler to find if one point of the regular n-gon is known beforehand. This is a similar restriction as in the previous section. There the rotation center of Q was given, but not the location of the vertices. Now we suppose we are given one vertex of Q, but do not know the rotation center.

Problem 3.3.16 (The ε-SD decision problem for symmetry group D_n with given rotational order and given vertex of Q)

Given: A point set P with rotational order $(p_1, \ldots, p_n)$ and a value $\varepsilon \geq 0$.

Question: Is there a D_n-symmetric point set $Q = \{p_1, q_2, \ldots, q_n\}$ with order $(p_1, q_2, \ldots, q_n)$, so that $d(p_i, q_i) \leq \varepsilon$, for all $2 \leq i \leq n$?

The rotation center of Q lies in the intersection of the $n-1$ apex disks defined by p_1 and the disks $D_2, \ldots, D_n$ respectively. Here D_j is the disk with center p_j and radius ε, for all $2 \leq j \leq n$. We will prove this in the following lemma:

Lemma 3.3.17. *Let a point $p_1 \in \mathbb{R}^2$ and $n-1$ disks $D_2, \ldots, D_n$ be given. Let $p_2, \ldots, p_n$ be the centers of the disks and let each disk have radius ε. Let $D^{(j-1)\alpha}_{1,j}$ denote the apex disk defined by p_1, D_j and angle $(j-1)\alpha$, where $\alpha = \frac{2\pi}{n}$. Then there is a regular n-gon $Q = (p_1, q_2, \ldots, q_n)$, so that $d(q_j, p_j) \leq \varepsilon$, for all $2 \leq j \leq n$ iff $\bigcap_{2 \leq j \leq n} \left(D^{(j-1)\alpha}_{1,j}\right) \neq \emptyset$.*

Proof. "⇒" Suppose the intersection of the apex disks is not empty. Let c be a point in this intersection. By Lemma 3.3.15 the radius of $D^{(j-1)\alpha}_{1,j}$ is $\frac{\varepsilon}{2\sin\left(\frac{(j-1)\alpha}{2}\right)}$. Thus,

$$d(c, a^{(j-1)\alpha}_{1,j}) \leq \frac{\varepsilon}{2\sin\left(\frac{(j-1)\alpha}{2}\right)},$$

where $a^{(j-1)\alpha}_{1,j}$ is the apex point defined by p_1, p_j and $(j-1)\alpha$. Let $q_j = \rho_c^{-(j-1)\alpha}(p_1)$. Then $(p_1, q_2, \ldots, q_n)$ is a regular n-gon with rotation center c. It remains to show that $d(q_j, p_j) \leq \varepsilon$. By Lemma 3.3.14

$$d(q_j, p_j) \leq 2\sin\left(\frac{(j-1)\alpha}{2}\right) d(c, a^{(j-1)\alpha}_{1,j}) = 2\sin\left(\frac{(j-1)\alpha}{2}\right) \frac{\varepsilon}{2\sin\left(\frac{(j-1)\alpha}{2}\right)} = \varepsilon$$

"$\Leftarrow$" Suppose there exists a regular n-gon $Q = (p_1, q_2, \ldots, q_n)$ so that $d(p_j, q_j) \leq \varepsilon$. Let c be its rotation center. We need to show that $c \in D_{1,j}^{(j-1)}$ for all $2 \leq j \leq n$.

$$d(c, a_{1,j}^{(j-1)\alpha}) = \frac{d(q_j, p_j)}{2\sin\left(\frac{(j-1)\alpha}{2}\right)} \leq \frac{\varepsilon}{2\sin\left(\frac{(j-1)\alpha}{2}\right)}$$

where the first equality holds by Lemma 3.3.14.
Thus c is in the intersection of the apex disks and it is therefore not empty. This completes the proof. □

Using Lemma 3.3.17 we can state Algorithm 3.3.4 which solves the ε-SD decision problem with given rotational order and vertex of Q in linear time.

Algorithm 3.3.4 Algorithm for the ε-SD decision problem for symmetry group D_n with given rotational order and vertex of Q.

SDD$_n$PartitionAndVertex$((p_1, p_2, \ldots, p_n), \varepsilon)$
  $\alpha = \frac{2\pi}{n}$;
  for $j = 2$ to n **do**
    Compute the apex disk $D_{1,j}^{(j-1)\alpha}$ defined by p_1, p_j and $(j-1)\alpha$ and ε;
  end for
  //***disksIntersect*** *is a procedure which decides if the intersection of the given disks is not empty.*
  return disksIntersect$(D_{1,2}^{\alpha}, D_{1,3}^{(2)\alpha}, \ldots, D_{1,n}^{(n-1)\alpha})$;

Theorem 3.3.18. *Algorithm 3.3.4 solves the ε-SD decision problem for symmetry group D_n with given order and vertex of Q in $O(n)$ time.*

Proof. The correctness follows from Lemma 3.3.17. The computation of the $O(n)$ apex disks needs constant time per disk and can therefore be done in linear time. We can also test in $O(n)$ if $O(n)$ disks intersect by an algorithm of Reichling [35]. Therefore, the overall running time is $O(n)$. □

Again, as in Section 3.3.1, we can do without the knowledge of the rotational order of the input points by using a matching algorithm on bipartite graphs. In Section 3.3.1 we rotated each input point by each possible rotation angle. Afterwards we labeled the resulting points by the number of the original point and the number of the rotation angle. We computed all possible disks defined by rotated point and tested if the labels of the points contained in that disk contain a matching in a bipartite graph defined by points, angles and labels.

We can apply the same arguments to the case where one vertex of Q is given but no rotational order on the input points is known. Since we do not know the rotational order one parameter for the computation of the apex disk is missing. For each point $p_j \in \{p_2, \ldots, p_n\}$ we will therefore compute all apex disks defined by p_1, p_j and an

angle $\alpha_i \in \Lambda = \{i\alpha | 1 \leq i \leq n-1\}$. As a result we will get $\Theta(n^2)$ apex disks which we denote by $D_{1,j}^{i\alpha}$. Analogous to Algorithm 3.3.2 we will test for all intersections of the apex disks if the labels of the intersecting disks define a perfect matching in a bipartite graph. Again we will construct one bipartite graph for each intersection. The vertex sets are the point set $\{p_2, \ldots, p_n\}$ on one side and the set Λ on the other side. We will add an edge between point p_j and angle α_i iff the intersection of the apex disk $D_{1,j}^{i\alpha}$ and the considered intersection region is not empty.

Algorithm 3.3.5 Algorithm for the ε-SD decision problem for symmetry group D_n with given vertex of Q.

SDD$_\mathbf{n}$ $(P = (p_1, p_2, \ldots, p_n), \varepsilon)$
  answer $= NO$
  $\alpha = \frac{2\pi}{n}$
  for $j = 2$ to n **do**
    for $i = 1$ to $n-1$ **do**
      Compute the apex disk $D_{1,j}^{i\alpha}$ defined by p_1, p_j and $i\alpha$
    end for
  end for
  Compute the arrangement of all disks in $D\{_{1,j}^{i\alpha} | 2 \leq j \leq n, 1 \leq i \leq n-1\}$;
  //Traverse the arrangement by depth first search.
  for all regions D in the arrangement **do**
    Build bipartite graph $G = (\{p_2 \ldots, p_n\} \cup \{\alpha_1, \ldots, \alpha_{n-1}\}, E)$, where $(p_j, \alpha_i) \in E$ iff $D \cap D_{1,j}^{i\alpha} \neq \emptyset$
    answer $=$ (answer) $\vee$ (G contains perfect matching);
  end for
  return answer;

Lemma 3.3.19. *Algorithm 3.3.5 solves the ε-SD decision problem for symmetry group D_n with given vertex of Q in time $O(n^4 M(n))$, where $M(n)$ is the time required to test if a perfect matching in a bipartite graph exists.*

Proof. The correctness follows from the correctness of Algorithm 3.3.4 and Lemma 3.3.6. What remains to be done is to analyze the running time. Since we construct $O(n^2)$ apex disks, we will get an arrangement of disks with $O(n^4)$ cells. This arrangement can be computed in $O(n^4)$ time by using an algorithm stated by Amato et al. [4]. For each cell we need to compute the bipartite graph and test if a perfect matching exists. Thus the running time is $O(n^4 M(n))$, where $M(n)$ is the time needed to test if a bipartite graph with $O(n)$ vertices contains a perfect matching. □

Dynamic Matching

We can reduce the running time by using a dynamic matching algorithm, see Section 3.2.2. We construct a graph G which represents the arrangement of the $O(n^2)$ apex

disks. Each intersection of two disks defines a vertex of G and two vertices are connected by an edge if they are neighbors on the boundary of a disk. Thus there are $O(n^4)$ vertices and $O(n^4)$ edges in G. G is a planar graph and it and its faces can be computed in $O(n^4 \log n)$ time by using an naive algorithm. We can then traverse the arrangement of the apex disk via the edges of this graph by computing the dual graph $\hat{G}$ in $O(n^2)$ and traverse it.
When traversing the arrangement of disks via the circular arcs, the number of apex disks which contain the considered intersection region increases or decreases by one. Thus we need to perform at most one insert edge and one delete edge operation to the dynamic graph while traversing from one intersection region to the next. After each step, the time needed to recompute the maximum matching of the dynamic graph is $O(n^2)$ since the graph contains $O(n^2)$ edges at any time. Thus the running time can be reduced to $O(n^4 n^2) = O(n^6)$. This improves the running time of Algorithm 3.3.5:

Lemma 3.3.20. *The ε-SD decision problem for symmetry group D_n with given vertex of Q in can be decided in $O(n^6)$ time by using dynamic matching.*

Bottleneck Distance

As in Algorithm 3.3.3 the bottleneck distance can be used in order to decide for each cell in the arrangement of disks if it gives a solution to the problem. The running time is then $O(n^4 n\sqrt{n} \log n) = O(n^5 \sqrt{n} \log n)$ since the bottleneck distance is applied to a regular n-gon defined by a point from each cell and the known vertex q.

Lemma 3.3.21. *The ε-SD decision problem for symmetry group D_n with given vertex of Q in can be decided in $O(n^5 \sqrt{n} \log n)$ time by using the bottleneck distance.*

Theorem 3.3.22. *Let a point set $P \subset \mathbb{R}^2, |P| = n$ and a point $v \in \mathbb{R}^2$ be given. The ε-SD decision problem for symmetry group D_n where v is a vertex of Q can be decided in time $O(n^6)$ by using dynamic matching and in time $O(n^5 \sqrt{n} \log n)$ by using the bottleneck distance.*

Proof. The theorem follows from Lemma 3.3.20 and Lemma 3.3.21. □

We will now generalize the above algorithm to the case where neither the rotation center nor a vertex of the regular n-gon is known. Let us again consider the input as a set of disks $D_1, \ldots, D_n$, where D_i has center p_i and radius ε, for all $1 \leq i \leq n$. Algorithm 3.3.5 decides the ε-SD decision problem for symmetry group D_n if p_1 is forced to be a vertex of the regular n-gon. There, one point in the intersection of the apex disks together with the known vertex of Q uniquely defines the regular n-gon.

For the general version of the ε-SD decision problem for symmetry group D_n, which we will solve now, neither the rotation center nor a vertex of Q is known. At least one vertex of Q can be forced to lie on the boundary of a disk with radius ε centered at a point $p \in P$.

Lemma 3.3.23. *Let a point set P, $|P| = n$ and a value $\varepsilon \geq 0$ be given. If there exists a D_n-symmetric point set Q which ε-approximates P then there exists a D_n-symmetric point set $\tilde{Q}$ which ε-approximates P and $d(p, \tilde{f}(p)) = \varepsilon$ for at least one point $p \in P$.*

Proof. Let Q be the regular n-gon which ε-approximates P and let c be its rotation center. Suppose no vertex of Q lies on the boundary of the ε-disk around its corresponding point in P. Then the radius of the regular n-gon Q can be increased until one vertex lies on the boundary of the disk with radius ε around its corresponding point in P. See Figure 3.3.4 for illustration. □

The vertex of the regular n-gon Q is therefore defined by one parameter, namely the angle β which denotes the vertex' location on the boundary of the disk centered at one point of P. We assume this point to be $p_1 \in P$ for a moment.

We therefore need to define apex regions $\mathcal{A}^{\alpha}_{1,j}$ for D_1, D_j and α so that the intersection of the regions $\mathcal{A}^{\alpha}_{1,2}, \ldots, \mathcal{A}^{\alpha}_{1,n}$ defines the rotation center of Q as well as one vertex and therefore Q itself.

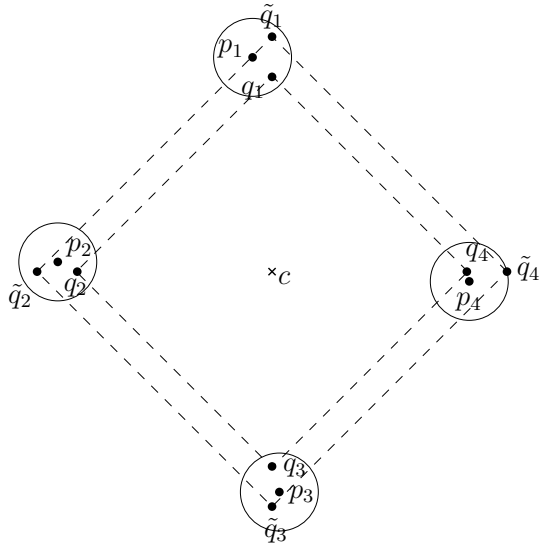


Figure 3.3.4: Illustration of Lemma 3.3.23

Definition 3.3.24 (Apex Region)**.** Let two disks D_1 and D_j with centers p_1 and p_j, resp. and radius ε and an angle α be given. The *apex region* of D_1 and D_j with respect to α is defined as:

$$\mathcal{A}^{\alpha}_{1,j} = \{(\beta, a) | a \in D^{\alpha}_{1,j}(\beta), 0 \leq \beta \leq 2\pi\}.$$

Here $D^{\alpha}_{1,j}(\beta)$ is the apex disk defined by p_j, $q(\beta) = (\varepsilon \cos\beta, \varepsilon \sin\beta) + p_1$ and α.

The apex region of p_1 and p_j is a three-dimensional object. The three dimensions are given by the x- and y-coordinates of the point $a \in D^{\alpha}_{1,j}(\beta)$ and the value of β.

For each fixed β the region is the apex disk defined by D_j and $q(\beta)$. When β changes, the apex disk rotates around the apex point defined by p_1, p_j and α. Thus the apex region $A^{\alpha}_{1,j}$ is a helix in $\mathbb{R}^3$. An illustration is given in Figure 3.3.5

We can state an analogous theorem considering the apex regions $\mathcal{A}^{\alpha}_{1,2}, \ldots, \mathcal{A}^{(n-1)\alpha}_{1,n}$ as Lemma 3.3.17. Again, for simplicity, we will assume that a rotational order on P is given.

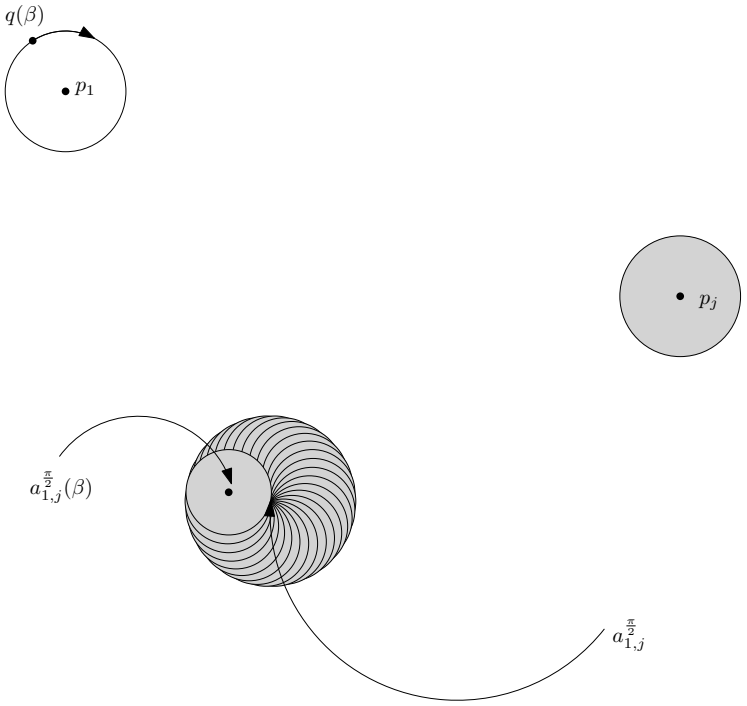


Figure 3.3.5: Illustration of the apex region defined by two disks.

Lemma 3.3.25. *Let a point set $P = (p_1, \ldots, p_n)$, a rotational order on P and a value $\varepsilon \geq 0$ be given. Furthermore, let D_j denote the disk with center p_j and radius ε, for all $1 \leq j \leq n$. Let $\mathcal{A}^{(j-1)\alpha}_{1,j}$ be the apex region defined by D_1, D_j and $(j-1)\alpha$, where $\alpha = \frac{2\pi}{n}$. There is a regular n-gon $Q = (q_1, \ldots, q_n)$ so that $q_j \in D_j$, for all $2 \leq j \leq n$ and q_1 is on the boundary of D_1 iff $\bigcap_{1 \leq j \leq n} \left(\mathcal{A}^{(j-1)\alpha}_{1,j}\right) \neq \emptyset$.*

Proof. "$\Rightarrow$" Suppose there is a regular n-gon which ε-approximates P and q_1 lies on the boundary of D_1 and let $q_1 = (\varepsilon \cos\beta, \varepsilon \sin\beta) + p_1$. Then the apex disks $D^{\alpha_j}_{1,j}(\beta), 2 \leq j \leq n$ all intersect by Lemma 3.3.17. Thus $\bigcap_{1 \leq j \leq n} \left(\mathcal{A}^{(j-1)\alpha}_{1,j}\right) \neq \emptyset$ holds.
"$\Leftarrow$" Suppose $\bigcap_{1 \leq j \leq n} \left(\mathcal{A}^{(j-1)\alpha}_{1,j}\right) \neq \emptyset$. Let (β, c) be a point in this intersection region. Then $c \in D^{(j-1)}_{1,j}(\beta)$ for all $2 \leq j \leq n$. Thus by Lemma 3.3.17 there exists a regular n-gon Q which ε-approximates P. Q is defined by c and $q(\beta)$, therefore the vertex $q(\beta)$ of Q lies on the boundary of D_1 by construction. $\square$

Using Lemma 3.3.25 we can state a polynomial time algorithm solving the ε-SD decision problem for symmetry group D_n with given order on P. In the case where we assume one vertex of Q to lie on D_1 we need to test, if all apex regions $\mathcal{A}_{1,j}^{(j-1)\alpha}, 2 \leq j \leq n$ intersect. We do not know which vertex of Q lies on the boundary of the disk centered at the corresponding point of P. Thus we need to iterate the procedure over all points in P.

Algorithm 3.3.6 Algorithm solving the ε-SD decision problem for symmetry group D_n with given order.

SDD$_\mathbf{n}$OrderKnown$(P = (p_1, \ldots, p_n), \varepsilon)$
  result = FALSE;
  $\alpha = \frac{2\pi}{n}$;
  for $i = 1$ to n **do**
    for $j = 1$ to n, $i \neq j$ **do**
      Compute the apex region $\mathcal{A}_{i,j}^{((j-i) \mod n)\alpha}$ defined by D_i, D_j
      and $((j-i) \mod n)\alpha$ and ε;
    end for
    result = result $\vee \left(\bigcap_{1 \leq j \leq n} \left(\mathcal{A}_{i,j}^{((j-i) \mod n)\alpha} \right) \neq \emptyset \right)$;
  end for
  return result;

Theorem 3.3.26. *Algorithm 3.3.6 solves the ε-SD decision problem for symmetry group D_n with given rotational order in time $O(n^6)$ time.*

Proof. The correctness of Algorithm 3.3.6 follows from Lemma 3.3.25. What remains to be done is to analyze the running time. For each point p_i, $1 \leq i \leq n$ we need to compute the arrangement of $n-1$ apex regions and test if they intersect. Basu et al. [7] stated an algorithm for computing a point from each cell of the arrangement of a set of n hypersurfaces in $\mathbb{R}^d$ each of degree at most b in time $O(n^{d+1}b^{O(d)})$ (see also the book of Sack and Urrutia [37]). Using this result, one can compute a point in each cell of the arrangement of apex regions in time $O(n^4)$ and for each point we need to test if it is contained in all apex regions. We iterate over all points in P. The overall running time is $O(n^6)$. □

In order to finally state an algorithm solving the ε-SD decision problem for symmetry group D_n we will need to decide if there is a rotational order on P, so that applying Algorithm 3.3.6 results in the answer YES. We will again use the same technique as for the case where a vertex of Q was given. For all $i \in \{1, \ldots, n\}$ we will compute all apex regions $\mathcal{A}_{i,j}^{l\alpha}$, $2 \leq j \leq n$ and $1 \leq l \leq n-1$. We then will compute all intersections and apply the matching argument as in Algorithm 3.3.5.

Lemma 3.3.27. *Algorithm 3.3.7 solves the ε-SD decision problem for symmetry group D_n in time $O((n^9 M(n))$.*

Proof. The correctness follows from the correctness of Algorithm 3.3.6 and the matching argument.
What remains to be done is to analyze the running time. For each point $p \in P$ we compute $O(n^2)$ apex regions in $\mathbb{R}^3$. We can compute a point in each cell of the arrangement in time $O(n^8)$ by using the result of Basu et al. [7] (see proof of Theorem 3.3.26). For each cell we need to construct the bipartite graph and test if a perfect matching exists. This takes time $O(M(n))$ per cell where $M(n)$ is the time needed to compute a maximum matching. We iterate over all points in P. Thus the overall running time is $O(n^8 M(n)n) = O(n^9 M(n))$. □

Algorithm 3.3.7 Algorithm for the ε-SD decision problem for symmetry group D_n.

SDD$_\mathbf{n}$ $(P = \{p_1, \ldots, p_n\}, \varepsilon)$
  answer = NO
  $\alpha = \frac{2\pi}{n}$
  for $i = 1$ to n **do**
    for $j = 1$ to n, $i \neq j$ **do**
      for $l = 1$ to $n - 1$ **do**
        Compute the apex disk $\mathcal{A}_{i,j}^{l\alpha}$ defined by p_i, p_j and $l\alpha$
      end for
    end for
    for Intersection regions $\mathcal{A}$ **do**
      Build bipartite graphs $G_i = (\{p_2 \ldots, p_n\} \cup \{\alpha_1, \ldots, \alpha_{n-1}\}, E)$, where $(p_j, \alpha_l) \in E$ iff $\mathcal{A} \cap \mathcal{A}_{i,j}^{l\alpha} \neq \emptyset$
      answer = (answer) $\vee$ (G_i contains perfect matching);
    end for
  end for
  return answer;

Bottleneck Distance

The bottleneck distance can be used in order to decide for each of the $O(n^8)$ cells in the arrangement of apex regions if it gives a solution to the problem. Applying the bottleneck distance takes $O(n\sqrt{n}\log n)$ time. We need to iterate over n points, thus the running time is $O(n^8 nn\sqrt{n}\log n) = O(n^{10}\sqrt{n}\log n)$.

Lemma 3.3.28. *The ε-SD decision problem for symmetry group D_n with given vertex of Q in can be decided in $O(n^{10}\sqrt{n}\log n)$ time by using the bottleneck distance.*

Theorem 3.3.29. *Let a point set $P \subset \mathbb{R}^2, |P| = n$ and a value $\varepsilon > 0$ be given. The ε-SD decision problem can be solved in time $O(n^9 M(n))$ by using matching in bipartite graphs or in time $O(n^{10}\sqrt{n}\log n)$ by using bottleneck distance.*

3.4 The ε-SD Problem for Symmetry Group C_k

In this Section we will state polynomial time algorithms solving the ε-SD problem for symmetry group C_k where a partition of the input set P of size $|P| = km$ into m subset of size k each is given. In contrast to Section 3.3, the symmetry group order is not dependent on the size of the input set. Again, we consider two cases. In the first case the rotation center c of the C_k-symmetric point set Q is given. Here we are able to solve the computational problem in polynomial time. The second case we consider is the one where the rotation center is not known beforehand. For this case we present a polynomial time algorithm for the decision problem.
For both variations of the ε-SD problem we can use the results of Section 3.3. Each partition set P_i, $1 \leq i \leq m$ has size k, which is exactly the number of the symmetry group we consider. The following lemma shows that we can use the partition sets in order to compute the C_k-symmetric point set Q for the input set P.

Lemma 3.4.1. *Let a point set P, $|P| = mk$ and a partition of P into m subsets $P_1 \ldots, P_m$ of size $|P_i| = k$, $1 \leq i \leq m$ each be given. Let Q be a C_k-symmetric point set which ε-approximates P. Then there is a partition of Q into m subsets $Q_1 \ldots, Q_m$ of size $|Q_i| = k$ each so that Q_i ε-approximates P_i for all $1 \leq i \leq m$.*

Proof. Let Q be the C_k-symmetric point set which ε-approximates P. Thus there is a bijection $f : P \to Q$, so that $d(p, f(p)) \leq \varepsilon$ for all $p \in P$. Let $Q_i = \{f(p) | p \in P_i\}$ for all $1 \leq i \leq m$. Then Q_i contains k elements. Since f is a bijection, $Q_i \cap Q_j = \emptyset$, for $i \neq j$. For all $1 \leq i \leq m$, Q_i ε-approximates P_i by the definition of Q_i. □

The basic idea of the two algorithms, the one for the case where the rotation center is known and the other for the case where the rotation center is not known, is to solve the ε-SD problem for the partition sets independently and use the results in order to solve the ε-SD problem for the whole input set P. For the case where the rotation center of the C_k-symmetric point set Q is given beforehand, the following lemma explains how to combine the results for the partition sets $P_1, \ldots, P_m$ to a result for the whole point set P.

Lemma 3.4.2. *Let a point set P, a partition of P into subsets of size k each, a symmetry group C_k and a rotation center c be given. Let furthermore $Q_1, \ldots, Q_m$ be D_k-symmetric point sets of size k all having c as their rotation center. Let $Q_i \cap Q_j = \emptyset$ for $i \neq j$. Suppose Q_i ε_i-approximates P_i, for $1 \leq i \leq m$. Then $Q = \bigcup_{1 \leq i \leq m} (Q_i)$ is a C_k-symmetric point set which ε-approximates P and $\varepsilon = \max\{\varepsilon_i | 1 \leq i \leq m\}$.*

Proof. Since Q_i ε_i-approximates P_i, there is a bijection $f_i : P_i \to Q_i$ so that $d(p, f_i(p)) \leq \varepsilon_i$ for all $1 \leq i \leq m$. We can use these bijections in order to define a bijection between P and Q. So $f : P \to Q$ and $f(p) = f_i(p)$, for $p \in P_i$. Since the sets Q_i do not intersect, f is well defined and a bijection. It remains to show

that Q ε-approximates P.

$$\begin{aligned} d(p, f(p)) &= d(p, f_i(p)), p \in P_i \\ &\leq \varepsilon_i \\ &\leq \varepsilon \end{aligned}$$

since $\varepsilon = \max\{\varepsilon_i | 1 \leq i \leq m\}$.
Thus, Q ε-approximates P. Q is C_k-symmetric since all subsets $Q_1, \ldots, Q_m$ are D_k-symmetric and they all have the same rotation center c. Thus

$$\begin{aligned} \rho_c^{\left(\frac{2\pi}{k}\right)}(Q) &= \rho_c^{\left(\frac{2\pi}{k}\right)}\left(\bigcup_{1\leq i\leq m}(Q_i)\right) \\ &= \bigcup_{1\leq i\leq m}\left(\rho_c^{\left(\frac{2\pi}{k}\right)}(Q_i)\right) \\ &= \bigcup_{1\leq i\leq m}(Q_i) \\ &= Q \end{aligned}$$

□

Note that although the point sets $Q_i, 1 \leq i \leq m$ are D_k-symmetric, this is not the case for the point set Q in general (see Figure 3.4.1). Q is only D_k symmetric in the case where all partition sets $Q_i, 1 \leq i \leq m$ are symmetric with respect to the same reflection line or if the union of $t < m$ partition sets is a D_{tk}-symmetric point set and is symmetric with respect to the same reflection line as the remaining partition sets (see Figure 3.4.2).

3.4.1 Rotation Center Known

We start by investigating the variation of the ε-SD decision problem where the rotation center as well as a partition of P into m subsets of size k each is given.

Problem 3.1.7 (The ε-SD computational problem for given partition of P and a given rotation center c.)

Given: A point set $P \subset \mathbb{R}^2, |P| = mk$, a symmetry group C_k, a partition of P into m subsets $P_1, \ldots, P_m$ of size $|P_i| = k, 1 \leq i \leq m$ each and a rotation center c.

Task: Compute the smallest value $\varepsilon \geq 0$, so that there exists a C_k-symmetric point set Q and a partition of Q into m subsets $Q_1, \ldots, Q_m$ all of size k, all subsets having symmetry group D_k and rotation center c, so that Q_i ε-approximates P_i, for all $1 \leq i \leq m$.

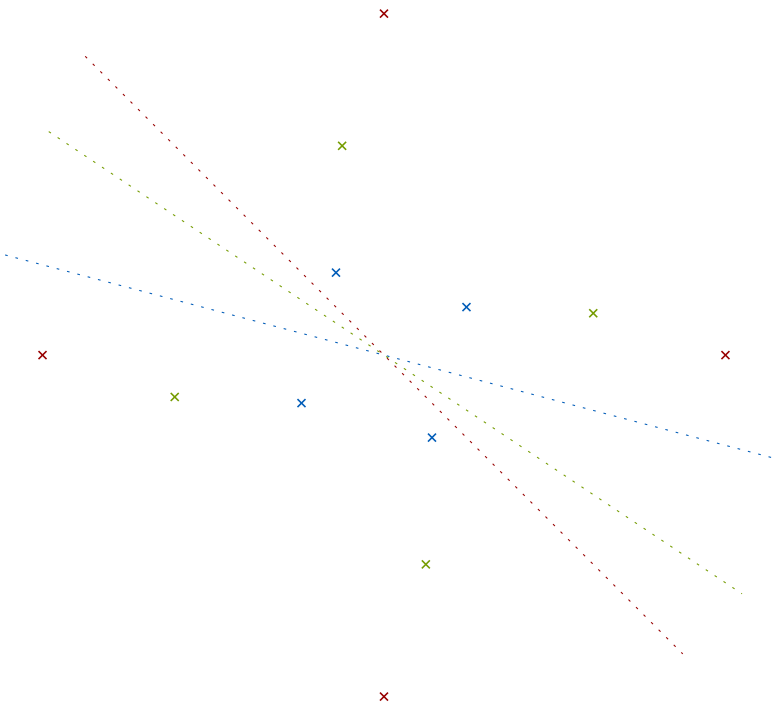
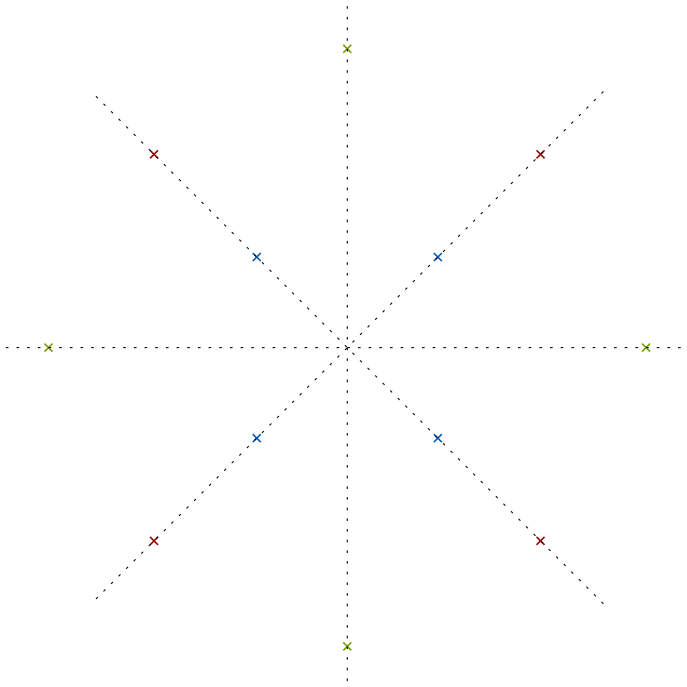
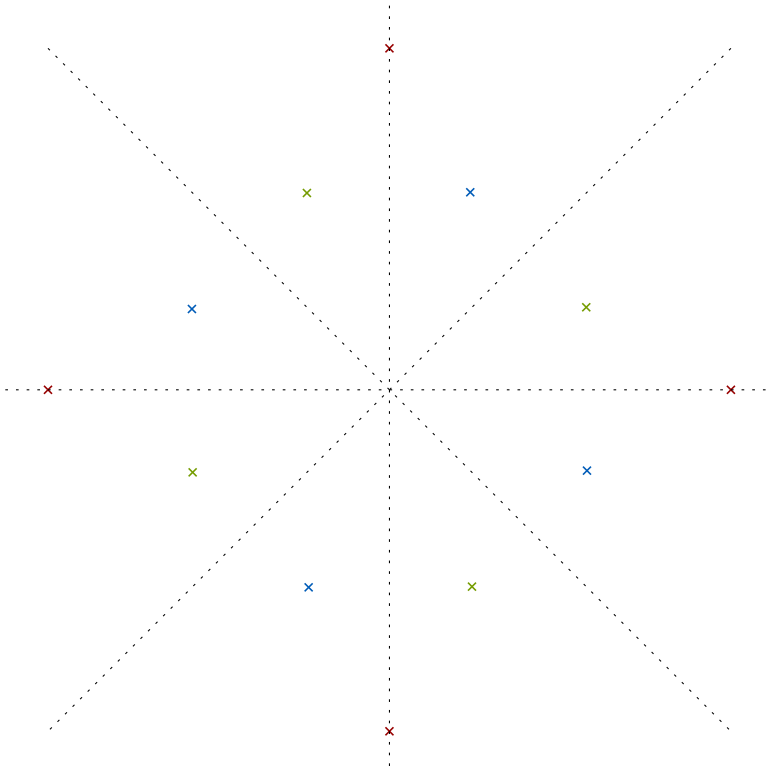


Figure 3.4.1: In general, the reflection lines of the D_k-symmetric partition sets do not coincide, and therefore the union of these sets is not D_k-symmetric but C_k-symmetric. For each partition set one reflection line is depicted.



(a) All partition sets are symmetric with respect to the same reflection lines (dotted).



(b) The blue and green subsets are not symmetric with respect to the reflection lines of the red subset, but their union is D_8-symmetric with respect to the same reflection line as the red point set.

Figure 3.4.2: Examples of a D_4-symmetric point set. The different partition sets are indicated by different colors. Each partition set on its own is D_4-symmetric.

In the previous section, we solved the ε-SD computational problem for point sets P, symmetry group $D_{|P|}$ and a given rotation center c. Since in this section, $|P_i| = k$ and we consider the symmetry group C_k, we can apply the algorithm developed in the previous section for each partition set P_i, symmetry group $D_{|P_i|}$ and rotation center c separately. For each partition set P_i the result is a value $\varepsilon_i \geq 0$. By Lemma 3.4.2 we know that if for each partition set P_i there exists a D_k-symmetric point set Q_i with rotation center c which ε-approximates P_i, then there is a C_k-symmetric point set Q with rotation center c which ε-approximates the whole point set P. By taking $\varepsilon = \max\{\varepsilon_i | 1 \leq i \leq m\}$ we get the solution of the ε-SD computational problem for point set P, a partition of P into m subsets of size k, a given rotation center c and symmetry group C_k.

An algorithm solving the ε-SD computational problem for a point set P, a given partition of P into m subsets for size k each, a given rotation center c and symmetry group C_k with polynomial running time is given in Algorithm 3.4.1.

Algorithm 3.4.1 Algorithm for the ε-SD computational problem for symmetry group C_k with given partition of P and rotation center c.

$\mathbf{SDC_k Partition}$ $(\{P_1, \ldots, P_m\}, c)$
  $\varepsilon = 0$;
  for $i = 1$ to m **do**
    $\varepsilon_{\text{temp}} =$ **$\mathbf{SDD_{|P_i|}Center}$**$(P_i, c)$;
    if $\varepsilon_{\text{temp}} > \varepsilon$ **then**
      $\varepsilon = \varepsilon_{\text{temp}}$;
    end if
  end for
  return ε.

We complete this section by proving the correctness and analyzing the running time of Algorithm 3.4.1.

Theorem 3.4.3. *Algorithm 3.4.1 solves the ε-SD computational problem for symmetry group C_k with given partition of P and rotation center c in time $O(nk^5M(k))$. Here $M(k)$ denotes the time for computing a perfect matching in bipartite graphs with k vertices.*

Proof. The correctness follows from Lemma 3.4.2 and Lemma 3.4.1. The algorithm for **$\mathbf{SDC_{|P|}Center()}$** is given in Algorithm 3.3.2. Since each partition set P_i has size k, the time needed to compute **$\mathbf{SDD_{|P|}Center}$**(P_i) is $O(k^6M(k))$. Since we execute Algorithm 3.3.2 m times, the running time of Algorithm 3.4.1 is $O(mk^6M(k)) = O(\frac{n}{k}k^6M(k)) = O(nk^5M(k))$. □

Remark 3.4.4. For some inputs it might be the case, that $Q_i = Q_j$ for $i \neq j$, where Q_i and Q_j ε-approximate P_i and P_j, respectively. In this case we allow Q to be a multiset.

Remark 3.4.5. The complexity of $M(k)$ can be taken from the proof of Lemma 3.3.6. Note that the running time of Algorithm 3.4.1 is linear if the symmetry group order is bounded by a constant number. This is the case in most of the realistic inputs we have considered in Chapter 2. Here we consider symmetry groups C_k where $k \leq 12$ in most of the cases. The input sets may contain hundreds of points and are partitioned into subsets of more or less constant size. Thus we apply a constant time algorithm to $m = O(n)$ subsets.

With the same arguments we can state a polynomial time algorithm for the ε-SD computational problem for symmetry group C_k where the rotation center c, the partition of P and additionally a rotational order on each partition set is given. In Algorithm 3.4.1 we do not need the knowledge of the rotational order to give a polynomial time algorithm. However, if the rotational order of each partition set is known, the running time decreases significantly. In Section 3.5 we will have the case, where we know the rotation center and can compute (under certain assumptions) the partition of P as well as the rotational order on the partition sets in polynomial time from the input set P. Thus, we state Algorithm 3.4.2 which we will use in Section 3.5:

Algorithm 3.4.2 Algorithm for the ε-SD computational problem for symmetry group C_k with given partition of P, rotational order on the partition sets and rotation center c.

SDC$_\mathbf{k}$PartitionOrder$(\{(p_{(1,1)}, \ldots, p_{(1,k)}), \ldots, (p_{(m,1)}, \ldots, p_{(m,k)})\}, c)$
  $\varepsilon = 0$;
  for $i = 1$ to m **do**
    $\varepsilon_{\text{temp}}$ = **SDD$_{|\mathbf{P_i}|}$OrderAndCenter(**$(p_{(i,1)}, \ldots, p_{(i,k)})$**,** c**)**;
    if $\varepsilon_{\text{temp}} > \varepsilon$ **then**
      $\varepsilon = \varepsilon_{\text{temp}}$;
    end if
  end for
  return ε.

We prove the correctness and analyze the running time of Algorithm 3.4.2 in the following theorem.

Theorem 3.4.6. *Algorithm 3.4.2 solves the ε-SD computational problem for symmetry group C_k with given partition of P, rotational order on the partition sets and rotation center c in time $O(n)$.*

Proof. The correctness follows from the correctness of Algorithm 3.3.1 and from Lemma 3.4.2 and Lemma 3.4.1. Algorithm 3.3.1 has running time $O(k)$ and is called

m times, thus the overall running time is $O(mk) = O(n)$. □

3.4.2 Rotation Center Not Known

The first problem we will investigate in this section is the ε-SD decision problem with given partition of P into m subsets of size k each and given rotational order on each of these partition sets. Afterwards, we will consider the case where only a partition of P into m subsets of size k each is given but no rotational order is known.

Problem 3.1.8 (The ε-SD decision problem for given partition of P and given rotational orders on each partition set.)

Given: A set $P \subset \mathbb{R}^2, |P| = mk$, a symmetry group C_k, a value $\varepsilon \geq 0$, a partition of P into m subsets $P_1, \ldots, P_m$ of size k each and a rotational order on each partition set.

Question: Is there a C_k-symmetric point set Q and a partition of Q into subsets $Q_1, \ldots, Q_m$, all of size k and all having symmetry group D_k, so that Q_i ε-approximates P_i with respect to the rotational order on P_i for all $1 \leq i \leq m$?

This problem was already investigated and solved in polynomial time by Iwanowski [18]. We will use the same ideas for our algorithm. We will state them by using the above defined apex points. The analysis of the running time given by Iwanowski [18] is based on the complexity of the considered arrangement but does not take into account the time needed to compute the arrangement. We will present a detailed analysis of the running time of the algorithms.

The crucial point in the algorithm solving Problem 3.1.8 is to determine for each partition set P_i, $1 \leq i \leq m$ the set of points which serve as a rotation center of a regular k-gon Q_i which ε-approximates P_i and to test if all these m point sets have a common intersection.

As we already saw in Section 3.3, a rotation center c is the center of a regular k-gon Q_i which ε-approximates P_i, iff the radius of the smallest enclosing disk of the points of P_i rotated around c with respect to their given rotational order is smaller or equal to ε.

The radius of a smallest enclosing disk of a set R_i of rotated points defined by points in P_i and the rotational order defined on P_i is smaller or equal to ε iff the radius of each enclosing disks of subsets of three points in R_i is smaller or equal to ε (see Iwanowski [18]).

Consider a triple of points p_1, p_2, p_3 of a partition set and let $\alpha_1, \alpha_2, \alpha_3$ be the angles defining the rotational order of these three points within their set. Let r be the radius of the smallest enclosing disk of $\rho_c^{\alpha_1}(p_1), \rho_c^{\alpha_2}(p_2), \rho_c^{\alpha_3}(p_3)$. Then $r \leq \varepsilon$ if and only if one of the following conditions holds:

1. The radius of the disk with $\rho_c^{\alpha_1}(p_1), \rho_c^{\alpha_2}(p_2), \rho_c^{\alpha_3}(p_3)$ on the boundary is smaller or equal to ε.

2. $d(\rho_c^{\alpha_1}(p_1), \rho_c^{\alpha_2}(p_2)) \leq 2\varepsilon$ and
$d(\rho_c^{\alpha_1}(p_1), \rho_c^{\alpha_2}(p_2))^2 \geq d(\rho_c^{\alpha_1}(p_1), \rho_c^{\alpha_3}(p_3))^2 + d(\rho_c^{\alpha_2}(p_2), \rho_c^{\alpha_3}(p_3))^2$

3. $d(\rho_c^{\alpha_1}(p_1), \rho_c^{\alpha_3}(p_3)) \leq 2\varepsilon$ and
$d(\rho_c^{\alpha_1}(p_1), \rho_c^{\alpha_3}(p_3))^2 \geq d(\rho_c^{\alpha_1}(p_1), \rho_c^{\alpha_2}(p_2))^2 + d(\rho_c^{\alpha_2}(p_2), \rho_c^{\alpha_3}(p_3))^2$

4. $d(\rho_c^{\alpha_2}(p_2), \rho_c^{\alpha_3}(p_3)) \leq 2\varepsilon$ and
$d(\rho_c^{\alpha_2}(p_2), \rho_c^{\alpha_3}(p_3))^2 \geq d(\rho_c^{\alpha_1}(p_1), \rho_c^{\alpha_2}(p_2))^2 + d(\rho_c^{\alpha_1}(p_1), \rho_c^{\alpha_3}(p_3))^2$

The first condition is true in the case where the smallest enclosing disk with radius smaller or equal to ε is defined by all three points, where the other three conditions describe the case where the radius of the smallest enclosing disk with radius smaller or equal to ε is defined by two of the three points.

For two points p_i and p_j with associated angles α_i and α_j let $a_{i,j}$ be the apex point defined by p_i, p_j, and the angle $|\alpha_i - \alpha_j|$. Then by Lemma 3.3.14 $d(\rho_c^{\alpha_i}(p_i), \rho_c^{\alpha_j}(p_j)) = 2\sin\left(\frac{|\alpha_i-\alpha_j|}{2}\right) d(c, a_{i,j})$.

We can therefore rewrite the conditions in terms of apex points:

1. The radius of the disk defined by the vertices of a triangle with side lengths $2\sin\left(\frac{\alpha_1-\alpha_2}{2}\right) d(a_{1,2}, c)$, $2\sin\left(\frac{\alpha_1-\alpha_3}{2}\right) d(a_{1,3}, c)$, $2\sin\left(\frac{\alpha_2-\alpha_3}{2}\right) d(a_{2,3}, c)$ is smaller or equal to ε.

2. $2\sin\left(\frac{\alpha_1-\alpha_2}{2}\right) d(a_{1,2}, c)) \leq 2\varepsilon$ and
$\left(2\sin\left(\frac{\alpha_1-\alpha_2}{2}\right) d(a_{1,2}, c)\right)^2 \geq \left(2\sin\left(\frac{\alpha_1-\alpha_3}{2}\right) d(a_{1,3}, c)\right)^2 + \left(2\sin\left(\frac{\alpha_2-\alpha_3}{2}\right) d(a_{2,3}, c)\right)^2$

3. $2\sin\left(\frac{\alpha_1-\alpha_3}{2}\right) d(a_{1,3}, c)) \leq 2\varepsilon$ and
$\left(2\sin\left(\frac{\alpha_1-\alpha_3}{2}\right) d(a_{1,3}, c)\right)^2 \geq \left(2\sin\left(\frac{\alpha_1-\alpha_2}{2}\right) d(a_{1,2}, c)\right)^2 + \left(2\sin\left(\frac{\alpha_2-\alpha_3}{2}\right) d(a_{2,3}, c)\right)^2$

4. $2\sin\left(\frac{\alpha_2-\alpha_3}{2}\right) d(a_{2,3}, c)) \leq 2\varepsilon$ and
$\left(2\sin\left(\frac{\alpha_2-\alpha_3}{2}\right) d(a_{2,3}, c)\right)^2 \geq \left(2\sin\left(\frac{\alpha_1-\alpha_3}{2}\right) d(a_{1,3}, c)\right)^2 + \left(2\sin\left(\frac{\alpha_1-\alpha_2}{2}\right) d(a_{1,2}, c)\right)^2$

Remark 3.4.7. The radius r of the disk defined by the three side length a, b and c of a triangle is given by $r = \frac{a}{\sin(\alpha)}$, where α is the angle opposite to the side with length a. Also, $\cos(\alpha) = \frac{a^2-b^2-c^2}{-2bc}$ and therefore

$$r \leq \varepsilon \Leftrightarrow r^2 = \frac{a^2}{4\left(1 - \frac{a^2-b^2-c^2}{-2bc}\right)} \leq \varepsilon^2$$

Using these conditions we can test for the existence of a regular k-gon which ε-approximates P_i:
There is a regular k-gon Q_i which ε-approximates P_i iff there is a point c so that for

all triple of points of P_i one of these four conditions holds. This was already proven by Iwanowski [18].

Each of these conditions is given by a polynomial which defines a suitable region in $\mathbb{R}^2$. By using an algorithm of Basu et al. [7] we can compute a point of each cell of the arrangement of these regions. If and only if there is a regular k-gon which approximates P_i then there is one whose rotation center is one of the points computed by the algorithm of Basu et al. [7]. This can be seen as follows:
In the case where there is a regular k-gon which ε-approximates P_i, its rotation center c fulfills one of the conditions defined above for each triple of P_i. Thus the corresponding regions of all these triples intersect. On the other hand, if for each triple of points one of the conditions are fulfilled for the same point c, then it is a rotation center with the property that all smallest enclosing disks have radius smaller or equal to ε. Thus there is a regular k-gon with rotation center c which approximates P_i.
For one partition set we need to compute the arrangement of $7k^3$ regions and test for each point representing a cell of the arrangement if it is the rotation center of a regular k-gon ε-approximating P_i.

A C_k-symmetric point set Q which ε-approximates P is the union of m regular k-gons $Q_1, \ldots, Q_m$ all having the same rotation center and Q_i ε-approximates P_i, $1 \leq i \leq m$. For each partition set we compute the regions for each triple of points given by the above conditions. We then consider the arrangement of all these regions of all partition sets. We ask for a point c with the property, that at least one condition is fulfilled for each triple of points within the partition sets. Again we can apply the algorithm of Basu et al. [7] in order to compute a set of points representing each cell of the arrangement. For each point we check, if it is the rotation center of a C_k-symmetric point set which ε-approximates P.

Theorem 3.4.8. *Let a point set P, $|P| = mk = n$, a partition of P into m subsets of size k each and a rotational order on each subset be given. Algorithm 3.4.3 decides the ε-SD decision problem for symmetry C_k and point set P in time $O\left(n^4k^6\right)$*

Proof. The correctness follows from the above observations and was proven by Iwanowski [18]. What remains do be done is to analyze the running time. Computing a point of each cell of the arrangement of $O\left(mk^3\right)$ regions in $\mathbb{R}^2$ takes time $O\left((mk^3)^3\right) = O\left(m^3k^9\right)$ by using the algorithm of Basu et al. [7]. There are $O\left(m^3k^9\right)$ cells and thus $O\left(m^3k^9\right)$ rotation center candidates need to be tested. For each rotation center candidate it takes $O(n)$ time to test if it is the center of a C_k-symmetric point set which ε-approximates P. Thus the overall running time is $O\left(m^3k^9n\right) = O\left(n^4k^6\right)$. □

We can generalize the above ideas in order to state a polynomial time algorithm deciding the ε-SD decision problem in the case where the rotational orders of the partition sets are not part of the input. This is a variant of the ε-SD problem which was not considered so far.

Algorithm 3.4.3 Algorithm deciding the ε-SD decision problem for symmetry group C_k, given partition of P and given rotational order.

SDC$_k$PartitionOrder$(\{(p_{(1,1)},\ldots,p_{(1,k)}),\ldots,(p_{(m,1)},\ldots,p_{(m,k)})\},\varepsilon)$
  for $i=1$ **to** m **do**
    for each triple of points in P_i **do**
      Define polynomials representing the conditions as above.
    end for
  end for
  Compute set of points C representing each cell of the arrangement given by the polynomials, use algorithm of Basu et al. [7].
  for each $c \in C$ **do**
    if **SDC$_k$PartitionOrder**$\{(p_{(1,1)},\ldots,p_{(1,k)}),\ldots,(p_{(m,1)},\ldots,p_{(m,k)})\},c) \leq \varepsilon$ **then**
      return TRUE;
    end if
  end for
  return FALSE;

The procedure **SDC$_k$PartitionOrder**$\{(p_{(1,1)},\ldots,p_{(1,k)}),\ldots,(p_{(m,1)},\ldots,p_{(m,k)})\},c)$ is stated in Algorithm 3.4.2.

Problem 3.1.8 (The ε-SD decision problem for given partition of P.)

Given: A set $P \subset \mathbb{R}^2, |P| = mk$, a symmetry group C_k and a partition of P into m subsets $P_1,\ldots,P_m$ of size k each and a value $\varepsilon \geq 0$.

Question: Is there a C_k-symmetric point set Q and a partition of Q into subsets $Q_1,\ldots,Q_m$ all of size k and all having symmetry group D_k so that Q_i ε-approximates P_i for all $1 \leq i \leq m$?

The conditions for a point c having the property that a set of rotated points given by a triple of points rotated around c are defined by the three points and the three angles which define the rotated points. In the case where the rotational order of each partition set was given, the mapping between the points in P_i and the rotation angles was given by the rotational order. Thus we needed to consider 7 conditions for each triple of points.

In the case where the rotational order is not given, we need to consider 7 conditions for each triple point-angle-pair and compute the arrangement of the regions defined by those conditions. For each cell of the arrangement we compute a point c representing this cell and test if c is the rotation center of a C_k-symmetric point set which ε-approximates P with respect to the partition of P. There exist C_k-symmetric point sets which ε-approximates P with respect to the partition of P iff

the center of one of them represents one of the cells of the arrangement. This can be seen as follows:
Let Q be the C_k-symmetric point set which ε-approximates P with respect to the partition of P and let c be its rotation center. Then there is an order on the points of each partition set P_i such that the radius of the smallest enclosing points of the rotated points defined by the points in the partition sets, the rotational order and the center c is smaller or equal to ε. Therefore the regions defined by these conditions intersect and c is a point in this intersection.

Algorithm 3.4.4 Algorithm deciding the ε-SD decision problem for symmetry group C_k, given partition of P.

SDC$_\mathbf{k}$Partition$(\{P_1, \ldots, P_m\}, \varepsilon)$
  for $t = 1$ **to** m **do**
    for each triple of point-angle-pairs in P_i **do**
      Define polynomials representing the conditions as above.
      Compute set of points C representing each cell of the arrangement given by the polynomials, use algorithm of Basu et al. [7].
    end for
  end for
  for each $c \in C$ **do**
    if SDC$_\mathbf{k}$Partition$(\{P_1, \ldots, P_m\}, c) \leq \varepsilon$ **then**
      return TRUE;
    end if
  end for
  return FALSE;

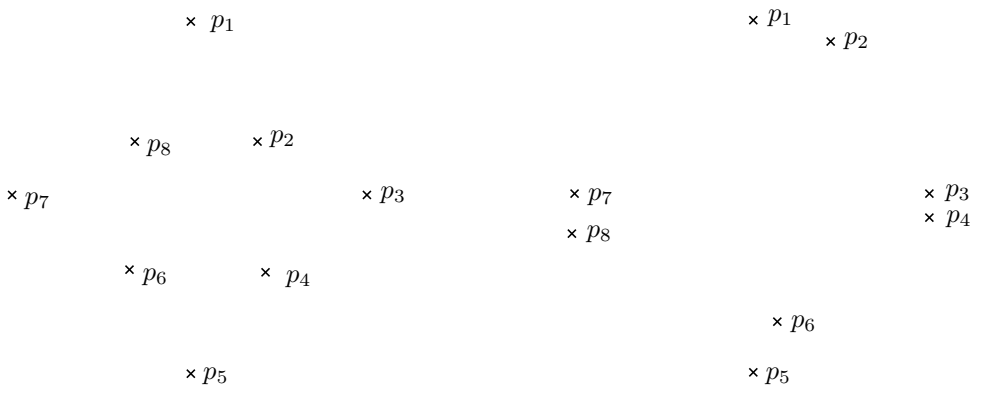
The procedure **SDC$_\mathbf{k}$Partition**$(\{P_1, \ldots, P_m\}, c)$ is stated in Algorithm 3.4.1.

Theorem 3.4.9. *Let a point set P, $|P| = km = n$ a partition of P into m subsets of size k each and a value ε be given. Algorithm 3.4.4 decides the ε-SD decision problem for symmetry group C_k and P in time $O(n^4k^{20}M(k))$.*

Proof. The correctness follows from the above observations. What remains to be done is to analyze the running time. For each point-angle triple of the points in the partition sets we need to consider 7 regions defined by the above conditions. We need to compute a set of points representing each cell of an arrangement of $7mk^6$ regions. Using the results of Basu et al. [7] this takes time $O\left((mk^6)^3\right)$ and the size of the set of representing points is $O\left((mk^6)^3\right)$. We apply Algorithm 3.4.1 for each of those points. Algorithm 3.4.1 has running time $O\left(nk^5M(k)\right)$, where $M(k)$ is the time needed to compute a maximum matching in a bipartite graph with k vertices. Thus the overall running time is $O(m^3k^{18}nk^5M(k)) = O(n^4k^{20}M(k))$. □

3.5 The ε-SD Problem for γ-Disjoint Point Sets

In the last three sections, we proved the ε-SD decision problem to be in P if we consider the symmetry group C_2, as shown in Section 3.2, if the symmetry group is dependent on the number of input points, as shown in Section 3.3, or if a partition of the input set of subsets of size k each is given. Thus the complicated part of solving the ε-SD problem is to find the partition of the input set P. The task of finding a partition of P becomes easier if the points are well separated.



(a) A good choice for the partition would be $\{p_1, p_3, p_5, p_7\}$ and $\{p_2, p_4, p_6, p_8\}$.

(b) It is not obvious which partition would lead to the optimal solution.

Figure 3.5.1: Considering symmetry group C_4, the optimal partition in (a) is easier to find than the optimal partition in (b).

We use a definition given by Iwanowski [18] to formalize this assumption:

Definition 3.5.1. Let $P = \{p_1, \ldots, p_n\}$ be a set of points in the plane. We call P γ-disjoint, iff $d(p_i, p_j) > \gamma$, for all $p_i, p_j \in P, p_i \neq p_j$, where $d(p_i, p_j)$ denotes the Euclidean distance between the points p_i and p_j.

In his thesis, Iwanowski [18] presents a polynomial time algorithm for the ε-SD decision problem in the case where the input set is 8ε-disjoint. By generalizing this approach, we are able to give a polynomial time algorithm for the ε-SD decision problem for 4ε-disjoint point sets if the rotation center c of the C_k-symmetric point set Q which ε-approximates P is given. Note that this is not the same problem as considered in Section 3.4 since we are not given a partition of P into subsets of size k each anymore.
We will also give a polynomial time algorithm for the ε-SD decision problem where we neither need to know the rotation center of Q nor a partition of P. We only ask the point set to be $4t\varepsilon$-disjoint and we demand the existence of a point $\tilde{c}$ which can be computed in polynomial time from P and has the property that the solution of the ε-SD computational problem for symmetry group C_k and rotation center $\tilde{c}$ is a

t-approximation of the solution of the ε-SD computational problem for symmetry group C_k. A formal definition of t-approximation will be given in Definition 3.5.7 later on in this section.

3.5.1 The ε-SD Problem for γ-Disjoint Point Sets with Given Rotation Center

We will start by explaining how to solve the ε-SD decision problem for a point set P and a given rotation center c if P is 4ε-disjoint in polynomial time. Thus the problem we will consider first is the following:

Problem 3.5.2 (The ε-SD decision problem for 4ε-disjoint point set P and given rotation center.)

Given: A symmetry group C_k, a value $\varepsilon \geq 0$, a 4ε-disjoint point set $P \subset \mathbb{R}^2$ where $|P| = mk$ and a rotation center $c \in \mathbb{R}^2$.

Question: Is there a C_k-symmetric point set Q with rotation center c which ε-approximates P?

Theorem 3.5.3. *Algorithm 3.5.1 solves the ε-SD decision problem for symmetry group C_k and 4ε-disjoint point sets with given rotation center c in time $O(n \log n)$, where n is the size of the input set P.*

Proof. We will first show the correctness of the algorithm.
First suppose that there exists a C_k-symmetric point set Q which ε-approximates P. By Lemma 3.4.1 there is a partition of Q into m C_k-symmetric subsets $Q_1, \ldots, Q_m$, $|Q_i| = k$, $1 \leq i \leq m$, all having rotation center c and a corresponding partition of P into subsets $P_1, \ldots, P_m$, $|P_i| = k$, so that Q_i ε-approximates P_i, for all $1 \leq i \leq m$.

Consider a subset P_i and its corresponding C_k-symmetric subset Q_i. By Lemma 3.3.11 the existence of a point set Q_i which ε-approximates P_i implies that there is an order $(p_{(i,2)}, \ldots, p_{(i,k)})$ on P_i, so that
$d(p_{(i,1)}, \rho_c^{(j-1)\alpha}(p_{(i,j)})) \leq 2\varepsilon$ in especially $d(\rho_c^{-(j-1)\alpha}(p_{(i,1)}), p_{(i,j)}) \leq 2\varepsilon$, for all $2 \leq j \leq k$.
This is equal to the condition checked in line 6 since $\rho_c^{-(j-1)\alpha}(p_{(i,1)})$ denotes the j^{th} vertex of the regular k-gon defined by c and $p_{(i,1)}$.
Therefore we are able to compute a partition of P into m subsets of size k in line 6. The answer to the ε-SD decision problem for P and the computed partition given by Algorithm 3.4.2 will be YES. It is not possible that a vertex Q_i is mapped to more than one point of P due to the fact that P is 4ε-disjoint.

Suppose there are two points $p_{(i,j)}, \tilde{p}_{(i,j)} \in P$, so that $d(\rho_c^{-(j-1)\alpha}(p_{(i,1)}), p_{(i,j)}) \leq 2\varepsilon$ and $d(\rho_c^{-(j-1)\alpha}(p_{(i,1)}), \tilde{p}_{(i,j)}) \leq 2\varepsilon$. Then

$$\begin{aligned} d(p_{(i,j)}, \tilde{p}_{(i,j)}) &\leq d(\rho_c^{-(j-1)\alpha}\left(p_{(i,1)}\right), p_{(i,j)}) + d(\rho_c^{-(j-1)\alpha}\left(p_{(i,1)}\right), \tilde{p}_{(i,j)}) \\ &\leq 2\varepsilon + 2\varepsilon \\ &= 4\varepsilon \end{aligned}$$

This is a contradiction to the assumption that P is 4ε-disjoint.
Therefore, if there exists a C_k-symmetric set Q which ε-approximates P, then Algorithm 3.5.1 computes the correct partition of P into m subsets of size k each in line 6 and the answer of Algorithm 3.4.2 will be YES.

On the other hand, suppose that no point set Q exists which ε-approximates P. In the case where we computed a partition, the answer of Algorithm 3.4.2 will be NO. In the case where we are not able to compute a partition, the answer is NO in line 8. Therefore the answer of Algorithm 3.5.1 will be NO if no C_k-symmetric point set Q exists which ε-approximates P.

What remains to be done is to analyze the running time. We choose at most m points from P, one for each partition set. Computing the regular k-gon defined by a point and a rotation center takes $O(k)$ time per regular k-gon, so in total $O(n)$ time. For each vertex of the regular k-gons we need to find the nearest point in P and check if the distance to that point is at most 2ε. This can be done by computing the Voronoi diagram in $O(n \log n)$ time of the point set P and then locate the Voronoi cell of each vertex in $O(\log n)$ time. We compute m regular k-gons and thus $mk = n$ vertices. Therefore we need $O(n \log n)$ time in order to compute the partition of P and the rotational orders on the partition sets. The rotational order on each subset is given by the well-defined rotational order of the corresponding regular k-gon. Thus we can apply Algorithm 3.4.2 which runs in $O(n)$ time. The total running time is $O(n \log n)$. □

Algorithm 3.5.1 Polynomial time algorithm for the ε-SD decision problem for symmetry group C_k and 4ε-disjoint point set P with given rotation center c.

SDC$_\mathbf{k}$DisjointRotationCenter(P, c, ε)

```
1  answer = YES;
2  i = 1;
3  while P ≠ ∅ do
4     Choose an arbitrary point p_(i,1) ∈ P;
5     Compute the regular k-gon Q_i defined by p_(i,1) and the rotation center c;
6     Compute the subset P_i ⊆ P with rotational order (p_(i,1), ..., p_(i,k)), so that
      d(p_(i,j), q_(i,j)) ≤ 2ε, for all 1 ≤ j ≤ k;
7     if for at least one vertex of Q_i no such point in P exists then
8        answer = NO;
9     end if
10    Remove all points in P_i from P;
11    i = i + 1;
12 end while
13 return answer ∧
       SDC_kPartitionOrder( {(p_(1,1), ..., p_(1,k)), ..., (p_(m,1), ..., p_(m,k))}, c)≤ ε;
```

The procedure **SDC$_\mathbf{k}$PartitionOrder**$\{(p_{(1,1)}, \ldots, p_{(1,k)}), \ldots, (p_{(m,1)}, \ldots, p_{(m,k)})\}, c)$ is stated in Algorithm 3.4.2.

3.5.2 ε-SD Problem for γ-Disjoint Point Sets Without Given Rotation Center

We will now consider the case where the rotation center of the C_k-symmetric point set Q is not known. We will present polynomial time algorithms, but we need the point set P to be $4t\varepsilon$-disjoint, where $t > 1$ will be determined later.

Lemma 3.5.4. *Let a point set $P \subset \mathbb{R}^2$, $|P| = n = mk$, a symmetry group C_k and a value $\varepsilon > 0$ be given. Suppose there is a C_k-symmetric point set Q which ε-approximates P and let c_{sol} be the rotation center of Q. Let $c \in \mathbb{R}^2$ be a point in the δ-neighborhood of c_{sol}. Then there exists a C_k-symmetric point set $\tilde{Q}$ with rotation center c which $(\delta + \varepsilon)$-approximates P.*

Proof. Since the C_k-symmetric point set Q ε-approximates P, there is a bijection $f : P \to Q$, so that $d(p - f(p)) \leq \varepsilon$, for all $p \in P$. Consider the point set $\tilde{Q} = \{\tilde{f}(p) = f(p) + \overline{c_{sol}c} \,|p \in P\}$. The point set $\tilde{Q}$ is C_k-symmetric since Q is C_k-symmetric and $\tilde{Q}$ is the translation of Q by the vector $\overline{c_{sol}c}$. The rotation center of $\tilde{Q}$ is the point c by construction. The distance $d(p, \tilde{f}(p))$ for each $p \in P$ can be computed as follows:

$$d(p, \tilde{f}(p)) = d(p, (f(p) + \overline{c_{sol}c})) \leq \underbrace{d(p, f(p))}_{\leq \varepsilon} + \underbrace{d(c, c_{sol})}_{\leq \delta} \leq \varepsilon + \delta$$

Thus Q $(\delta + \varepsilon)$-approximates P. □

We will now present a generic algorithm which decides the ε-SD decision problem for symmetry group C_k in polynomial time if the point set P is $4t\varepsilon$-disjoint. We will also specify the value t. The algorithm we will present is based on the same idea as Algorithm 3.5.1 which decides the ε-SD decision problem with given rotation center in polynomial time for 4ε-disjoint point sets. Instead of the given rotation center used in Algorithm 3.5.1 we use a point c so that there exists a point set $\tilde{Q}$ having rotation center c which ε_c-approximates P and $\varepsilon_c \leq t\varepsilon$ if it is possible at all to ε-approximate the point set P.

Lemma 3.5.5. *Let a point set $P \subset \mathbb{R}^2$, $|P| = n = mk$, a symmetry group C_k and a value $\varepsilon > 0$ be given. Suppose there is a point $c \in \mathbb{R}^2$ which can be computed in time $ARC(n)$ and has the following property:*
If there exists a C_k-symmetric point set Q which ε-approximates P, then there exists a C_k-symmetric point set $\tilde{Q}$ with rotation center c which ε_c-approximates P and $\varepsilon_c \leq t\varepsilon$.
Then Algorithm 3.5.2 decides the ε-SD decision problem for point set P if P is $4t\varepsilon$-disjoint.
The running time is $O\left(ARC(n) + \left(n^4k^6\right)\right)$, where $ARC(n)$ is the time needed to compute c.

Proof. Consider the regular k-gon Q_i defined by c and a point $p_{(i,1)} \in P$. If for one vertex $q_{(i,j)}$ of Q_i there is no point of P in the $2t\varepsilon$-neighborhood of $q_{(i,j)}$ then $\varepsilon_c > t\varepsilon$. This is due to the fact that the optimal value for a given partition is the radius of the smallest enclosing disk of the rotated points and thus is larger than half the distance between each pair of rotated points (see Section 3.3). Thus there exists no point set Q which ε-approximates P by the property of P. This implies that the answer NO is correct in line 8.
On the other hand, assume a partition of P into m subsets of size k is computed by the algorithm. Then Algorithm 3.4.3 which is used in line 13 gives the correct answer.
It is not possible that there are two points of P in the $2t\varepsilon$-neighborhood of the vertex of a regular k-gon, due to the fact that P is $4t\varepsilon$-disjoint by assumption. Therefore Algorithm 3.5.2 is correct. What remains to be done is to analyze the running time. Computing a regular k-gon from c and a point $p_{i_1} \in P$ takes $O(k)$ time. There is at most one point of P in the $2t\varepsilon$-neighborhood of each vertex of the regular k-gon. Thus we can build the Voronoi diagram of the point set P in $O(n \log n)$ preprocessing time and determine in time $O(\log n)$ for each vertex of the regular k-gon the Voronoi cell in which it is contained and thus the point of P closest to that vertex. Overall it takes $O(k \log n)$ time to perform the operations in the while-loop. In each step we delete k points from P and thus the while-loop is processed at most $m = \frac{n}{k}$ times. Therefore processing the while-loop in total takes $O(n \log n)$ time. The running time

of the procedure **SDC$_k$PartitionOrder** is $O\left(n^4k^6\right)$. Thus the overall running time is $O\left(ARC(n) + \left(n^4k^6\right)\right)$. □

Algorithm 3.5.2 Polynomial time algorithm for the ε-SD decision problem for symmetry group C_k and a $4t\varepsilon$-disjoint point set.

SDC$_k$TEpsDisjoint(P, ε)
1 $c =$ **RotationCenter**(P);
2 **return** **SDC$_k$TEpsDisjoint**(P, c, ε);

Algorithm 3.5.3 Polynomial time algorithm for the ε-SD decision problem for symmetry group C_k and a $4t\varepsilon$-disjoint point set and given rotation center c.

SDC$_k$TEpsDisjoint(P, c, ε)
1 answer = YES;
2 $i = 1$;
3 **while** $P \neq \emptyset$ **do**
4 Choose an arbitrary point $p_{(i,1)} \in P$;
5 Compute the regular k-gon Q_i defined by $p_{(i,1)}$ and c;
6 Compute the subset $P_i \subseteq P$, so that each point in P_i has distance smaller or equal $2t\varepsilon$ to a vertex of Q_i;
7 **if** for at least one vertex of Q_i no such point in P exists **then**
8 answer = NO;
9 **end if**
10 Remove all points in P_i from P;
11 $i = i + 1$;
12 **end while**
13 partition = **SDC$_k$PartitionOrder**$(\{\{p_{(1,1)}, \ldots, p_{(1,k)}\}, \ldots, \{p_{(m,1)}, \ldots, p_{(m,k)}\}, \varepsilon\})$
14 answer = answer $\wedge$ partition;
15 **return** answer;

We will now explain how to compute points from P which fulfill the properties needed for Lemma 3.5.5.
In order to do so, we need to consider the corresponding computational problem:

Problem 3.5.6 (The ε-SD computational problem for symmetry group C_k.)

Given: A set $P \subset \mathbb{R}^2$ of $n = mk$ points, and a symmetry group C_k.

Task: Compute the smallest $\varepsilon \geq 0$, so that there is a C_k-symmetric point set Q which ε- approximates P.

The ε-SD computational problem is an optimization problem. Therefore we can describe a solution of the ε-SD computational problem as a function s which maps a point set P of size $n = mk$ to the real numbers. The optimal solution for the ε-SD computational problem is denoted by s_{opt}, and $s_{opt}(P) = \varepsilon_{opt}$ is the optimal value. We call the corresponding C_k-symmetric point set Q_{opt}. If we could compute the rotation center c_{opt} of Q_{opt} in polynomial time, we would get a polynomial time algorithm for the ε-SD problem for 4ε-disjoint point sets by applying Algorithm 3.5.1. Unfortunately, it is not clear how to compute c_{opt} in polynomial time.

Definition 3.5.7. Let B be an optimization problem with input b and let s be a solution for B. A solution s_{opt} is called optimal iff $s_{opt}(b) \leq s(b)$ for all solutions s if B is a minimization problem or iff $s_{opt}(b) \geq s(b)$ for all solutions s if B is a maximization problem, respectively.

This definition leads to Lemma 3.5.8 which is similar to Lemma 3.5.5. Lemma 3.5.8 relates the value of ε_c to ε_{opt} and not to ε as Lemma 3.5.5 does. The advantage is that ε_{opt} is related to the input set P whereas ε is given independently of P.

Lemma 3.5.8. *Let a point set $P \subset \mathbb{R}^2, |P| = n$, a symmetry group C_k and a value $\varepsilon > 0$ be given. Let $c \in \mathbb{R}^2$ be a point which can be computed from P in time $ARC(n)$. Let furthermore ε_c be the solution of the ε-SD computational problem for a given point c and input set P and let ε_{opt} be the solution of the ε-SD computational problem for input set P and let $\varepsilon_c \leq t\varepsilon_{opt}$. Then the ε-SD decision problem can be decided in time $O\left(ARC(n) + \left(n^3k + n^2k^3\right)\right)$ if P is $4t\varepsilon$-disjoint.*

Proof. If there exists a C_k-symmetric point set Q which ε-approximates P, then $\varepsilon_{\text{opt}} \leq \varepsilon$ and therefore $\varepsilon_c \leq t\varepsilon_{opt} \leq t\varepsilon$. Thus we can apply Algorithm 3.5.2. □

Algorithm 3.5.2 gives a framework for polynomial time algorithms solving the ε-SD decision problem for symmetry group C_k and $4t\varepsilon$-disjoint point sets. One only needs to plug in a polynomial time algorithm which computes a point c, so that the optimal solution of the ε-SD decision problem for symmetry group C_k and rotation center c is a t-approximation of the optimal solution of the ε-SD decision problem for symmetry group C_k.

The goal is now to find a rotation center c which can be computed in polynomial time and leads to a small value t. Similar to Lemma 3.5.4, one attempt is to find a point that lies in the δ neighborhood of the optimal rotation center c_{opt} for a small value δ:

Lemma 3.5.9. *Let a point set P in the plane and a symmetry group C_k be given. Assume a point c to be in the δ-neighborhood of the optimal rotation center c_{opt}. Then there is a C_k-symmetric point set Q with rotation center c that $(\delta+\varepsilon_{opt})$-approximates P, where ε_{opt} is the optimal solution of the ε-SD computation problem for P.*

Proof. By setting $\varepsilon = \varepsilon_{opt}$ we can apply Lemma 3.5.4. □

A point that is easy to compute from the point set P is the center of mass of P. Since the center of mass c is given by

$$c = \frac{1}{|P|}\sum_{p\in P}(p)$$

it can be computed in linear time. The center of mass lies in the ε_{opt}-neighborhood of c_{opt} as the following lemma shows:

Lemma 3.5.10. *Let a point set $P \subset \mathbb{R}^2$ and a symmetry group C_k be given. Let ε_{opt} be the solution of the ε-SD computational problem for input set P, let Q_{opt} be the corresponding C_k-symmetric point set and let c_{opt} be its rotation center. Furthermore, let $s = \frac{1}{|P|}\sum_{p\in P} p$ be the center of mass of P. Then $d(s, c_{opt}) \leq \varepsilon_{opt}$.*

Proof.

$$d(s, c_{\text{opt}}) = d\left(\frac{1}{|P|}\sum_{p\in P} p, \frac{1}{|P|}\sum_{f(p)\in Q_{\text{opt}}} f(p)\right) \leq \frac{1}{|P|}\sum_{p\in P} d(p, f(p)) \leq \frac{1}{|P|}|P|\varepsilon_{opt} = \varepsilon_{opt}$$

□

Using the center of mass as rotation center, we can easily prove a result already shown by Iwanowski [18]:

Theorem 3.5.11. *Let a point set P in the plane, a symmetry group C_k and a value $\varepsilon > 0$ be given. The ε-SD decision problem can be decided in $O\left(n^3k + n^2k^3\right)$, if P is 8ε-disjoint.*

Proof. Let c be the center of mass of the point set P, so $c = \frac{1}{|P|}\sum_{p\in P} p$. By Lemma 3.5.10 and Lemma 3.5.9, $\varepsilon_c \leq 2\varepsilon_{\text{opt}}$ and thus by Lemma 3.5.8 the ε-SD decision problem can be decided in time

$$O\left(n + \left(n^3k + n^2k^3\right)\right) = O\left(n^3k + n^2k^3\right),$$

if P is 8ε-disjoint. □

As seen above, the center of mass of the point set P leads to a 2-approximation for the ε-SD computation problem. In the following we will introduce a point that gives a $\frac{2}{\sqrt{3}}$-approximation. We do not know if this point can be computed in polynomial time for arbitrary points sets P. But we can give a polynomial time algorithm computing this point if the point set P is $4\frac{2}{\sqrt{3}}\varepsilon$-disjoint.

3.5.3 The ε-SD Problem for $\frac{8\varepsilon}{\sqrt{3}}$-Disjoint Point Sets

In this section, we will explain how to compute a rotation center $c \in \mathbb{R}^2$ so that the optimal solution ε_c of the ε-SD computational problem with rotation center c is a $\frac{2}{\sqrt{3}}$-approximation of the optimal solution of the ε-SD computational problem. Using Lemma 3.5.8, we can give a polynomial time algorithm for the ε-SD decision problem if the input set is $\frac{8\varepsilon}{\sqrt{3}}$-disjoint, where $\frac{8}{\sqrt{3}} \approx 4.6188$. This is an improvement of the result presented in Theorem 3.5.11.

As in Section 3.3 we will use the apex point $a_{i,j}^{\alpha}$ defined by two points p_i, p_j and an angle α. In Section 3.3 we considered the ε-SD decision problem for symmetry group $D_{|P|}$. The problem we need to consider now is more complicated. On the one hand, we consider the symmetry group C_k where k does not depend on the size of the input set. On the other hand, we need to consider the ε-SD computational problem since we want to show that the rotation center we compute leads to a $\frac{2}{\sqrt{3}}$-approximation.

The algorithm we will state in order to compute the rotation center c is closely related to the algorithm given in Section 3.2. There we also needed to compute a rotation center, namely the rotation center which gives the optimal solution for the ε-SD computational problem for symmetry group C_2. For a given partition into subsets of size 2 we proved that the optimal rotation center is the center of the smallest enclosing disk of the midpoints defined by the two points in each subsets. We used a matching algorithm on general graphs in order to find the partition which results in the optimal rotation center.

The basic observation we used in order to prove the correctness of the algorithm is that the distance between a point p_i and the rotational image $\rho_c^{\pi}(p_j)$ of p_j with respect to the rotation around c by an angle π is twice the distance between c and the midpoint of p_i and p_j (see Lemma 3.2.4).

When considering the symmetry group C_k the midpoint of two points is generalized to the apex point as defined in Definition 3.3.12. By Lemma 3.3.14 we can relate the distance between a rotation center c and an apex point $a_{i,j}^{\alpha}$ to the distance between p_i and $\rho_c^{\alpha}(p_j)$ and vice versa. Note that $d(p_i, \rho_c^{\alpha}(p_j)) = 2\sin\left(\frac{\alpha}{2}\right) d(c, a_{i,j}^{\alpha})$ by Lemma 3.3.14. Thus the distance between a point p_i and the rotational image of p_j with respect to a rotation center c and an angle α is not only dependent on the distance between the apex point defined by p_i, p_j and α and c, but also on the angle α.

In Section 3.3 we explained how to solve the ε-SD computational problem for symmetry group $D_{|P|}$ in the case where the rotation center c and the rotational order on the input set P are given. We will now relate the optimal solution ε_{opt} to

the distance between apex points and rotation center. First we state a general lemma which relates the radius of a smallest enclosing disk of a point set to the maximum distance between two points in that set:

Lemma 3.5.12. *Let a point set $P = \{p_1, \ldots, p_n\}$ be given.*
Let $d_{max} = \max\{d(p_i, p_j) | 1 \leq i < j \leq n\}$ be the maximum distance between pairs of points in P. Furthermore, let D denote the smallest enclosing disk of P and let r denote its radius. Then r is bounded by $\frac{1}{2}d_{max} \leq r \leq \frac{1}{\sqrt{3}}d_{max}$.

Proof. The smallest enclosing disk D in P is defined by two or three points in P. First, suppose the smallest enclosing disk is defined by two points and w.l.o.g. let these two points be p_1 and p_2. Then $r = \frac{1}{2}d(p_1, p_2)$. Furthermore, the line segment $\overline{p_i p_j}$, for all $p_i, p_j \in P$ must be contained in D thus $d(p_j, p_i) \leq 2r = d(p_1, p_2)$. Therefore, $d(p_1, p_2) = d_{\max}$ and thus $r = \frac{1}{2}d_{\max}$ which proves the lemma for this case.
Now suppose the smallest enclosing disk is defined by three points p_1, p_2, p_3. These three points define a triangle. Let us assume w.l.o.g. that $\overline{p_1 p_2}$ is the side of the triangle with maximum length and let γ be the angle opposite to this side.
For the radius r of the disk defined by p_1, p_2, p_3 $r = \frac{|\overline{p_1,p_2}|}{2\sin(\gamma)}$ holds. Thus $\frac{|\overline{p_1,p_2}|}{2} \leq r \leq \frac{|\overline{p_1,p_2}|}{2\sqrt{3}}$ holds for $\frac{\pi}{3} \leq \gamma \leq \frac{\pi}{2}$. This is the only range possible for γ since we assume $\overline{p_1, p_2}$ to be the side with maximum length and all three points to be on the boundary of the disk. The line segment with length $d_{\max}$ between a pair of points is contained in the smallest enclosing disk. Therefore $\frac{1}{2}d_{\max} \leq r$ and $r \leq \frac{1}{\sqrt{3}}|\overline{p_1, p_2}| \leq \frac{1}{\sqrt{3}}d_{\max}$. This proves that the radius of the smallest enclosing disk of P is bounded by $\frac{1}{2}d_{\max} \leq r \leq \frac{1}{\sqrt{3}}d_{\max}$. □

Notation 3.5.13. *Let a point set $P, |P| = mk = n$ in the plane and a partition of P into subsets $P_1, \ldots, P_m$ all of size k be given. For two points $p_{(i,j)}, p_{(i,l)} \in P_i$ and an angle α the apex point defined by $p_{(i,j)}, p_{(i,l)}$ and α is denoted by $a^{\alpha}_{(i,j,l)}$.*

We now use Lemma 3.5.12 in order to relate the value ε_{opt} to the maximum distance between an apex point and the rotation center.

Lemma 3.5.14. *Let a point set P of size $|P| = mk$, a partition of P into subsets $P_1, \ldots, P_m$ of size $|P_i| = k$ and a rotation center c be given. Furthermore, let a rotational order $(p_{(i,1)}, \ldots, p_{(i,k)})$ on each partition set P_i, $1 \leq i \leq m$ be given. Let $\{a^{(l-j)\alpha}_{(i,j,l)} | 1 \leq i \leq m, 1 \leq j < l \leq k\}$ be the set of apex points where $\alpha = \frac{2\pi}{k}$. Let ε_{opt} be the solution of the ε-SD computational problem for symmetry group C_k with given rotation center, partition of the input set and rotational order on the partition sets. Then $\frac{d_{max}}{2} \leq \varepsilon_{opt} \leq \frac{d_{max}}{\sqrt{3}}$, where $d_{max} = \max\{2\sin\left(\frac{(l-j)\alpha}{2}\right) d\left(a^{(l-j)\alpha}_{(i,j,l)}, c\right) | 1 \leq i \leq m, 1 \leq j < l \leq k\}$ is the maximum weighted distance between the rotation center c and an apex point.*

Proof. Let $p_{(i,j)}$, $p_{(i,l)} \in P$ be the pair of points, where $d_{max} = 2\sin\left(\frac{(l-j)\alpha}{2}\right) d\left(a_{(i,j,l)}^{(l-j)\alpha}, c\right)$. Thus the distance between $p_{(i,j)}$ and $\rho_c^{(l-j)\alpha}\left(p_{(i,l)}\right)$ is $d\left(p_{(i,j)}, \rho_c^{(l-j)\alpha}\left(p_{(i,l)}\right)\right) = d_{max}$ by Lemma 3.3.14. Considering the set of rotated points $\{\rho_c^{(j-1)\alpha}\left(p_{(i,j)}\right) | 1 \leq j \leq k\}$ as computed in Algorithm 3.3.1. Then the distance

$$d\left(\rho_c^{(j-1)\alpha}\left(p_{(i,j)}\right), \rho_c^{(l-1)\alpha}\left(p_{(i,l)}\right)\right) = d\left(p_{(i,j)}, \rho_c^{(l-j)\alpha}\left(p_{(i,l)}\right)\right) = d_{max}.$$

Since the value of $\varepsilon_{i_{opt}}$ is given by the radius of the smallest enclosing disk of the rotated points $\{\rho_c^{(j-1)\alpha}\left(p_{(i,j)}\right) | 1 \leq j \leq k\}$, the line segment $\overline{\rho_c^{(j-1)\alpha}\left(p_{(i,j)}\right)\rho_c^{(l-1)\alpha}\left(p_{(i,l)}\right)}$ of length d_{max} must be contained in this smallest enclosing disk. Thus by Lemma 3.5.12, the value of $\varepsilon_{i_{opt}}$ for the partition set P_i is bounded by $\frac{d_{max}}{2} \leq \varepsilon_{i_{opt}} \leq \frac{d_{max}}{\sqrt{3}}$. Since $\varepsilon_{opt} = \max\{\varepsilon_{i_{opt}} | 1 \leq i \leq m\}$, as seen in Lemma 3.4.2, ε_{opt} is also bounded by $\frac{d_{max}}{2} \leq \varepsilon_{opt} \leq \frac{d_{max}}{\sqrt{3}}$ □

We will now state Algorithm 3.5.4, which computes a rotation center c in polynomial time, so that the solution of the ε-SD computational problem for symmetry group C_k with rotation center c is a $\frac{2}{\sqrt{3}}$-approximation of the ε-SD computational problem. We assume for a moment that the partition of P into subsets of size k each and a rotational order on the partition sets are given. We will explain later on how to find a partition and rotational order in polynomial time in the case where the input set P is $\frac{8\varepsilon}{\sqrt{3}}$-disjoint.

Algorithm 3.5.4 Algorithm deciding the ε-SD decision problem for $\frac{8}{\sqrt{3}}$-disjoint point sets (outline).

RotationCenter$(\{(p_{(1,1)}, \ldots, p_{(1,k)}), \ldots, (p_{(m,1)}, \ldots, p_{(m,k)})\})$

$\alpha = \frac{2\pi}{k}$

Compute the set of apex points $\{a_{(i,j,l)}^{(l-j)\alpha} | 1 \leq i \leq m, 1 \leq j < l \leq k\}$

Compute point c, so that

$d_{max} = \max\{2\sin\left(\frac{(l-j)\alpha}{2}\right) d\left(a_{(i,j,l)}^{(l-j)\alpha}, c\right) | 1 \leq i \leq m, 1 \leq j < l \leq k\}$ is minimized.

return c.

We will first prove the solution of the ε-SD computational problem with rotation center c to be a $\frac{2}{\sqrt{3}}$-approximation of the ε-SD computational problem. Afterwards, we will give more details on the steps in Algorithm 3.5.4, especially on how to find the point c which minimizes the maximum distance to the apex points.

Lemma 3.5.15. *Let a point set P, a partition of P into sets $P_1, \ldots, P_m$ of size k each and a rotational order on each partition set be given. Let c be a point which minimizes $d = \max\{2\sin\left(\frac{(l-j)\alpha}{2}\right) d\left(a_{(i,j,l)}^{(l-j)\alpha}, c\right) | 1 \leq i \leq m, 1 \leq j < l \leq k\}$. The solution of the ε-SD computational problem with rotation center c is then a $\frac{2}{\sqrt{3}}$-approximation of the solution of the ε-SD computational problem.*

Proof. Let ε_{opt} be the solution of the ε-SD computational problem for input set P, let Q_{opt} be the C_k-symmetric point set which ε_{opt}-approximates P and let c_{opt} denote its rotation center. Let $d_{opt} = \max\{2\sin\left(\frac{(l-j)\alpha}{2}\right) d\left(a_{(i,j,l)}^{(l-j)\alpha}, c_{opt}\right) | 1 \leq i \leq m, 1 \leq j < l \leq k\}$ be the maximum distance between c_{opt} and the set of apex points. Since d has the property that it minimizes the maximum distance to the set of apex points, $d \leq d_{opt}$. Thus by Lemma 3.5.14 $\frac{d_{opt}}{2} \leq \varepsilon_{opt} \leq \frac{d_{opt}}{\sqrt{3}}$ and $\frac{d}{2} \leq \varepsilon \leq \frac{d}{\sqrt{3}}$, where ε is the solution of the ε-SD computational problem for input set P and rotation center c. Therefore,

$$\varepsilon \leq \frac{d}{\sqrt{3}} \leq \frac{d_{opt}}{\sqrt{3}} \leq \frac{2\varepsilon_{opt}}{\sqrt{3}}$$

Thus the solution of the ε-SD computational problem for input set P and rotation center c is a $\frac{2}{\sqrt{3}}$-approximation of the solution of the ε-SD computational problem for input set P. □

We will now explain how to find for a given set $A \subset \mathbb{R}^2$ of apex points the point c which minimizes $\max\{2\sin\left(\frac{\alpha}{2}\right) d(\alpha(a), c) | a \in A\}$ where $\alpha(a)$ is the defining angle of the apex point a. Since the distances are dependent on the angle which defines the apex points, we can view the apex points as weighted points.

Finding the point which minimizes the maximum weighted distance to n points with weights can be done in $O(n(\log n)^3(\log\log(n))^2)$ by an algorithm of Megiddo [31].

Definition 3.5.16. We call the point minimizing the maximum weighted distance to n weighted points *smallest weighted center* of these n weighted points. The minimized distance is called the *smallest weighted radius* of the point set.

Lemma 3.5.17. *Let a point set P of n weighted points be given. The smallest weighted center and radius of P is defined by two or three points of P.*

Proof. Let c be the smallest weighted center and let r be the smallest weighted radius of P. Consider the point set $P' = \{c + w(p)\overline{cp} | p \in P\}$, where $w(p)$ denotes the weight of $p \in P$. Then c is the center of the smallest enclosing disk of P' and r is the radius of that disk. Thus c is defined by two or three points in P' and therefore also by two or three points of P. □

In Algorithm 3.5.4 we assume a partition of P into subsets of size k and a rotational order on each subset to be given. We will now explain how to partition P and how to find the rotational order on each subset under the assumption that P is $\frac{8}{\sqrt{3}}$-disjoint. We will use a similar approach to the one used in Algorithm 3.2.2. There we computed all midpoints defined by two points of the input set P. from this set of $O(n^2)$ midpoints we needed to find $\frac{n}{2}$ midpoints, so that each point of the input set is represented by exactly one midpoint. We computed all possible disks defined by two or three points and used a matching algorithm in order to verify that each point is represented by exactly one midpoint.

We can extend this approach to the weighted apex points.

For a given partition and rotational orders on the partition sets we compute the smallest weighted center of the subset of apex points given by the partition and rotational orders.

In the case where the partition and rotational orders are not known, we can compute the set A of all apex points defined by two points of the input set P and an angle $\alpha \in \Lambda = \{\frac{i2\pi}{k} | 1 \leq i \leq k-1\}$. A smallest weighted center is defined by two or three points of A. We therefore compute all smallest weighted centers defined by two or three apex points and examine the set of apex points with weighted distance smaller or equal to the smallest weighted radius. For a smallest weighted center c and radius r let this set be denoted by $A(c,r) = \{a \in A | 2\sin\left(\frac{\alpha(a)}{2}\right) d(a,c) \leq r\}$. It remains to check if $A(c,r)$ contains a subset of apex points which are the apex points given by a partition of P into m subsets of size k each and a rotational order on each subset.

In the case of symmetry group C_2 we needed to find a partition into subsets of size 2 each. We represented the partition sets by the midpoints defined by pairs of points of the input set.

When considering the symmetry group C_k, the situation is slightly different. An apex point defined by two points and a rotation angle α does not represent a subset of size k on its own. We need to consider a set of apex points in order to get a partition set.

Suppose c is a smallest weighted center and r the corresponding smallest weighted radius of a set of apex points given by a partition of P into set of size k and rotational order on each partition set. Then for each partition set P_i with rotational order $(p_{(i,1)}, \ldots, p_{(i,n)})$ the apex points $a_{(i,j,l)}^{(l-j)\alpha}, 1 \leq j < l \leq k$ define this partition and this order and therefore need to be in the set $A(c,r)$.

We can again use the approach of constructing a graph defined by the apex points and use a graph algorithm in order to solve the assignment problem. We construct a graph $G = (V, E)$ where the vertex set V is the point set P.

We assume that to each apex point a label representing the two defining points and the defining angle is assigned. For each apex point $a_{i,j}^{l\alpha} \in A(c,r)$, where $\alpha = \frac{2\pi}{k}$, we add the edge $e_{i,j}^{l}$ with label l connecting p_i and p_j to the edge set E. A subset $E' \subseteq E$ of the edges corresponds to a partition set P_i with rotational order if for each point of P_i there are $k-1$ distinct edges incident only to points in P_i and the set of labels of these edges is exactly Λ. Additionally, the local rotational orders need to be globally the same. So if there is an edge $e_{i,j}^{l_1}$ and an edge $e_{j,t}^{l_2}$ then there needs to be the edge $e_{i,t}^{l_1+l_2}$ We will prove in the following two lemmata that we only need to consider a smallest weighted center c defined by two or three apex points if the corresponding radius r is smaller or equal to $\frac{4\varepsilon}{\sqrt{3}}$ and that for such a smallest weighted center the graph defined by apex point in $A(c,r)$ can be build and the above explained property can be checked in polynomial time.

Lemma 3.5.18. *Let a point set $P, |P| = mk = n$, a symmetry group C_k and a value $\varepsilon \geq 0$ be given. Furthermore, let P be $\frac{8\varepsilon}{\sqrt{3}}$-disjoint. Let $A = \{a_{i,j}^{l\alpha} | 1 \leq i, j \leq n, 1 \leq$*

$l \leq k-1\}$ be the set of labeled apex points defined by P, where $\alpha = \frac{2\pi}{k}$. Let c be a smallest weighted center of a subset $A' \subseteq A$ of apex points. Suppose A' contains all apex points defined by a partition of P and rotational orders on each partition set. If there is a C_k-symmetric point set Q which ε-approximates P with respect to this partition and rotational order, then the weighted radius of c is smaller than $\frac{4\varepsilon}{\sqrt{3}}$.

Proof. Let c be the smallest weighted center and let r be the corresponding smallest weighted radius. Assume $r > \frac{4\varepsilon}{\sqrt{3}}$:
Then $\varepsilon_c \leq \frac{2}{\sqrt{3}}\varepsilon_{\text{opt}} \leq \frac{2}{\sqrt{3}}\varepsilon$, where the first inequality holds by Lemma 3.5.15 and the second inequality holds since we assume that there exists a C_k-symmetric point set Q which ε-approximates P.

Let $a_{i,j}^{l\alpha}$ be an apex point with weighted distance r to c. Then $d(a_{i,j}^{l\alpha}, c) > \frac{4\varepsilon}{\sqrt{3}} \frac{1}{2\sin(\frac{l\alpha}{2})}$ and thus by Lemma 3.5.14 $d(p_i, \rho_c^{l\alpha}(p_j)) > \frac{4\varepsilon}{\sqrt{3}}$. Thus the smallest enclosing disk of the rotated points must contain a line segment of length larger than $\frac{4\varepsilon}{\sqrt{3}}$. Its radius has therefore to be greater than $\frac{2\varepsilon}{\sqrt{3}}$ and thus $\varepsilon_c > \frac{2\varepsilon}{\sqrt{3}}$. This is a contradiction. Thus the lemma holds. □

The above Lemma 3.5.18 allows us to neglect all weighted smallest enclosing disks defined by two or three apex points whose radius is larger than $\frac{4\varepsilon}{\sqrt{3}}$. In the following lemma we will explain how to check for a subset of apex points if it defines a partition of P into subsets of size k each and a rotational order on each partition set in polynomial time.

Lemma 3.5.19. *Let a point set $P, |P| = mk = n$, a symmetry group C_k and a value $\varepsilon \geq 0$ be given. Let P be $\frac{8\varepsilon}{\sqrt{3}}$ disjoint. Let $A = \{a_{i,j}^{l\alpha} | 1 \leq i, j \leq n, 0 \leq l \leq k-1\}$, where $\alpha = \frac{2\pi}{k}$ be the set of labeled apex points defined by P. Let c be a weighted center with weighted radius $r \leq \frac{4\varepsilon}{\sqrt{3}}$. Then for each angle $l\alpha$, $0 \leq l \leq k-1$ and each point $p_i \in P$ there is at most one apex point $a_{i,j}^{l\alpha}$ with weighted distance smaller or equal to r to c.*

Proof. Suppose there are two apex points $a_{i,j_1}^{l\alpha}$ and $a_{i,j_2}^{l\alpha}$ with weighted distance smaller or equal to r to c. Then

$$d(a_{i,j_1}^{l\alpha}, c) \leq \frac{4\varepsilon}{\sqrt{3}} \frac{1}{2\sin\left(\frac{l\alpha}{2}\right)} \text{ and } d(a_{i,j_2}^{l\alpha}, c) \leq \frac{4\varepsilon}{\sqrt{3}} \frac{1}{2\sin\left(\frac{l\alpha}{2}\right)}.$$

Thus

$$d(a_{i,j_1}^{l\alpha}, a_{i,j_2}^{l\alpha}) \leq \frac{8\varepsilon}{\sqrt{3}} \frac{1}{2\sin\left(\frac{l\alpha}{2}\right)}.$$

By Lemma 3.3.14 it follows that $d(p_{j_1}, p_{j_2}) \leq \frac{8\varepsilon}{\sqrt{3}}$. This is a contradiction to the assumption that P is $\frac{8\varepsilon}{\sqrt{3}}$-disjoint. □

Lemma 3.5.19 assures that for each point p_i and each angle $l\alpha$ there is at most one point in P which can be the partner of p_i with respect to $l\alpha$ in the rotational order. Using this result we can check if $A(c, r)$ contain all the apex points defined

by a partition of P into subsets of size k each and a rotational order on each subset as follows:

Lemma 3.5.20. *Let a point set $P, |P| = mk$ and a value $\varepsilon \geq 0$ be given.*
Let $A = \{a_{i,j}^{l\alpha} | 1 \leq i, j \leq n, 0 \leq l \leq k-1\}$ be the set of apex points. $2\sin(\frac{l\alpha}{2})$ is the weight of apex point $a_{i,j}^{l\alpha}$.
Let c be a weighted center with weighted radius $r \leq \frac{4\varepsilon}{\sqrt{3}}$.
Let $A(c,r) \subset A$ be the set of apex points defined by c and r.
Assume that for all $j_1 \neq j_2$ and $a_{i,j_1}^{l_1\alpha}, a_{i,j_2}^{l_2\alpha} \in A(c,r)$ $l_1 \neq l_2$ holds.
Then we can decide in time $O(mk^2)$ if all apex points defined by a partition of P into m subsets of size k each are contained in $A(c,r)$.

Proof. For given weighted center c and weighted radius r and a set $A(c,r) \subseteq A$ of apex points we build a directed graph $G = (V, E)$. The vertex set V is the set P. The directed edge (i,j) is in E, iff $a_{i,j}^{\alpha} \in A(c,r)$, where $\alpha = \frac{2\pi}{k}$. By the assumption of this lemma, there is at most one edge incident to each vertex. Thus we can decide if there are m directed circles each of size k in G in time $O(n)$ since for each vertex we need to check if there is an edge incident to that vertex. If that is the case, we have found a partition of P into m subsets of size k each and a rotational order on each subsets. Let P_i be one partition set and let $(p_{(i,1)}, \ldots, p_{(i,k)})$ be its rotational order with respect to angle α. It remains to check that for each pair $(p_{(i,j)}, p_{(i,l)})$ the apex point $a_{(i,j,l)}^{(l-j)\alpha}$ is in $A(c,r)$. This can be done in time $O(mk^2)$ since we need to check at most one apex point for each pair in a partition set and have m partition sets. □

Theorem 3.5.21. *Algorithm 3.5.5 solves the ε-SD decision problem for symmetry group C_k and $\frac{8\varepsilon}{\sqrt{3}}$-disjoint point sets in time $O\left(n^6k^3\left(n^4k^6\right)\right)$.*

Proof. The correctness follows from Lemma 3.5.18, Lemma 3.5.19 and Lemma 3.5.20. What remains to be done is to analyze the running time.
Computing all apex points takes $O(n^2k)$ time. Computing a weighted center and weighted radius defined by two or three points can be done in constant time. We need to compute $O(n^6k^3)$ many of those centers, thus the overall running time for computing the weighted centers is $O(n^6k^3)$. For each center and radius computing the set of apex points $A(c,r)$ and building the graph takes time $O(n)$ since there are $O(n)$ edges contained in each graph. Checking if the disk defines a partition with rotational order on the partition sets takes time $O(mk^2) = O(nk)$ by Lemma 3.5.20. Each of the $O(n^6k^3)$ graphs gives at most one partition with rotational order which needs to be checked. We need to apply the procedure **SDC$_k$PartitionOrder** which has running time $O\left(n^4k^6\right)$ to each of the computed partitions. Thus the overall running time is $O\left(n^6k^3\left(n^4k^6\right)\right)$. □

Algorithm 3.5.5 Polynomial time algorithm solving the ε-SD decision problem for $\frac{8\varepsilon}{\sqrt{3}}$-disjoint point sets.

$\frac{\mathbf{8\varepsilon}}{\sqrt{\mathbf{3}}}$**Disjoint**$(P = \{p_1, \ldots, p_n\}, \varepsilon)$
  $\alpha = \frac{2\pi}{k}$;
  //Compute the set of apex points
  $A = \{a_{i,j}^{l\alpha} | 1 \leq i < j \leq n, 1 \leq l \leq k-1\}$;
  $C = \{(c,r) | (c,r)$ is weighted center/radius defined by two or three points in$P\}$
  for all $(c,r) \in C, r \leq \frac{4\varepsilon}{\sqrt{3}}$ **do**
    Compute the set of apex points $A(c,r) \subseteq A$;
    Check if all apex points defined by a partition of P into subsets of size k and a rotational order on each partition set are contained in $A(c,r)$ by applying Lemma 3.5.20;
    if SDC$_\mathbf{k}$PartitionOrder$(\{(p_{(1,1)}, \ldots, p_{(1,k)}), \ldots, (p_{(m,1)}, \ldots, p_{(m,k)})\}, \varepsilon)$ **then**
      return YES;
    end if
  end for
  return NO;

3.5.4 The ε-SD Problem for $4(1+\delta)\varepsilon$-Disjoint Point Sets

In this section we will show how to solve the ε-SD decision problem for a point set $P \subset \mathbb{R}^2$, $|P| = n$ and each value $\delta > 0$ in time $O\left(\left(\frac{1}{\delta^2}\right) n^2\right)$ if P is $4(1+\delta)\varepsilon$-disjoint. By Lemma 3.5.5 we can state a polynomial time algorithm for the ε-SD decision problem for a $4(1+\delta)$-disjoint point set if we can compute a point $c \in \mathbb{R}^2$ in polynomial time, so that $\varepsilon_c \leq (1+\delta)\varepsilon$. By Lemma 3.5.9 this is the case if c lies in the $\delta\varepsilon$-neighborhood of the rotation center of a C_k-symmetric point set Q which ε-approximates P.
In order to find such a point, we will construct a grid in such a way that at least one of the grid points has a distance smaller or equal to $\delta\varepsilon$ to the rotation center of a C_k-symmetric point set Q which ε-approximates P, if such a point set exists. For this point we can then apply Algorithm 3.5.3.

Algorithm 3.5.6 Algorithm deciding the ε-SD decision problem for symmetry group C_k and $4(1+\delta)\varepsilon$-disjoint point sets in polynomial time.

SDC$_\mathbf{k}$4(1 + δ)εDisjoint(P, ε, δ)
1 *// The point s denotes the center of mass of the points in P.*
2 $s = \frac{1}{|P|}\sum_{p\in P}(p)$;
3 Compute grid with grid side length $\sqrt{2}\delta\varepsilon$ centered at s on a square of side length $(2\varepsilon \times 2\varepsilon)$;
4 answer = NO;
5 **for** p_g is grid point **do**
6 answer = answer ∨ **SDC$_\mathbf{k}$TEpsDisjoint**(P, p_c, ε);
7 **end for**
8 **return** answer;

Theorem 3.5.22. *Algorithm 3.5.6 solves the ε-SD decision problem for $4(1+\delta)\varepsilon$-disjoint point sets in $O\left(\left(\frac{1}{\delta^2}\right)\left(n^4k^6\right)\right)$ time.*

Proof. Suppose there is a C_k-symmetric point set which ε-approximates P. Let ε_{opt} be the solution of the ε-SD computational problem for input set P and let Q_{opt} be the C_k-symmetric point set which ε-approximates P. Let c_{opt} be the rotation center of Q_{opt}. Then $\varepsilon_{\text{opt}} \leq \varepsilon$. The center of mass s of P lies in the ε_{opt}-neighborhood of c_{opt} by Lemma 3.5.10 and thus in the ε-neighborhood of c_{opt}. Since the grid has size $(2\varepsilon \times 2\varepsilon)$ and side length $\sqrt{2}\delta\varepsilon$, there is a grid point p_g, so that

$$d(p_g, c_{\text{opt}}) \leq \frac{1}{2}\sqrt{\left(\left(\sqrt{2}\delta\varepsilon\right)^2 + \left(\sqrt{2}\delta\varepsilon\right)^2\right)} = \delta\varepsilon$$

Therefore $\varepsilon_{p_g} \leq (1+\delta)\varepsilon$ and thus applying Algorithm 3.5.3 to p_g gives the correct answer by Lemma 3.5.5.
On the other hand, assume there is no C_k-symmetric point set which ε-approximates P. Then in line 13 of Algorithm 3.5.3, even if the algorithm managed to compute a partition with rotational order, the answer of Algorithm 3.4.3 would be NO since there exists no C_k-symmetric point set which ε-approximates P and thus no such point set with this particular partition and rotational order. The call of the procedure **SDC$_\mathbf{k}$TEpsDisjoint** in line 6 of Algorithm 3.5.6 is the only line where the answer YES could possibly be produced.

This shows the correctness of Algorithm 3.5.6.

What remains to be done is to analyze the running time. A $(2\varepsilon \times 2\varepsilon)$-grid with side length $\sqrt{2}\delta\varepsilon$ has $\left(\frac{2\varepsilon}{\sqrt{2}\delta\varepsilon}\right)^2 = \left(\frac{2}{\delta^2}\right)$ grid points. Thus Algorithm 3.5.3 is applied $\left(\frac{2}{\delta^2}\right)$ times. Its running time is $O\left(n^4k^6\right)$, thus the overall running time is $O\left(\left(\frac{1}{\delta^2}\right)\left(n^4k^6\right)\right)$. □

Remark 3.5.23. By Theorem 3.5.22 the ε-SD decision problem can be decided in time $O(n^4k^6)$ for $\frac{8\varepsilon}{\sqrt{3}}$-disjoint point sets. This is a much better result regarding the running time than the one given by Algorithm 3.5.5. This is due to the fact, that the computation of the rotation center used in Algorithm 3.5.5 is complicated and expensive. We nevertheless state the result of Algorithm 3.5.5 in this thesis since we believe the computed rotation center to be interesting while trying to understand the ε-SD problem in detail. We believed this point to be the rotation center of the solution of the ε-SD computational problem for symmetry group C_k with given partition but could not prove it. By now we believe that this is not true but think that the rotation center of the solution of the ε-SD computational problem for given partition can be computed by using the apex points in a similar way as the computation of the rotation center computed in Algorithm 3.5.5.

3.6 Summary

The following two tables provide an overview of the considered variations of the ε-SD problems. For each variation, the running time and reference to the algorithm is presented.
In Table 3.1 the results for the ε-SD decision problem for disjoint point sets is presented. Note that the results for 4ε-disjoint point sets hold for a given rotation center. For $4t\varepsilon$-disjoint point sets, the function $\mathrm{ARC}(n)$ describes the time needed to compute a point c out of the input point set P, s.t. the solution of the ε-SD computational problem for P w.r.t. rotation center c is a t-approximation of the solution of the ε-SD computational problem for P.

disjointness factor	running time	reference
4ε	$O(n \log n)$	Alg. 3.5.1
$4t\varepsilon$	$O\left(\mathrm{ARC}(n) + \left(n^4k^6\right)\right)$	Alg. 3.5.3
8ε	$O\left(n^4k^6\right)$	Thm. 3.5.11
$\frac{8}{\sqrt{3}}\varepsilon$	$O\left(n^6k^3\left(n^4k^6\right)\right)$	Alg. 3.5.5
$4(1+\delta)\varepsilon$	$O\left(\left(\frac{1}{\delta^2}\right)\left(n^4k^6\right)\right)$	Alg.3.5.6

Table 3.1: Results for the ε-SD decision problem for disjoint point sets.

In Table 3.2 we present the results for the variations of the ε-SD problem we considered in this thesis.

			decision problem	computational problem
C_2	partition		$O(n)$ Alg. 3.2.1	$O(n)$ Alg. 3.2.1
	no partition		$O\left(n^6\right)$ Iwanowski [18]	$O\left(n^8\sqrt{n}\right)/O(n^8)$ Alg. 3.2.2/Thm. 3.2.15
$D_{\lvert P\rvert}$	center	order	$O(n)$ Alg. 3.3.1	$O(n)$ Alg. 3.3.1
		no order	$O\left(n^6M(n)\right)/O\left(n^7\sqrt{n}\log n\right)$ Alg. 3.3.2/3.3.3	$O\left(n^6M(n)\right)/O\left(n^7\sqrt{n}\log n\right)$ Alg. 3.3.2/3.3.3
	vertex	order	$O(n)$ Alg. 3.3.4	
		no order	$O\left(n^6\right)/O\left(n^5\sqrt{n}\log n\right)$ Thm. 3.3.22	
	original	order	$O(n^6)$ Alg. 3.3.6	
		no order	$O\left(n^9M(n)\right)/O\left(n^{10}\sqrt{n}\log n\right)$ Thm. 3.3.29	
C_k	center	order	$O(n)$ Alg. 3.4.2	$O(n)$ Alg. 3.4.2
		no order	$O\left(nk^5M(k)\right)$ Alg. 3.4.1	$O\left(nk^5M(k)\right)$ Alg. 3.4.1
	original	order	$O\left(n^4k^6\right)$ Alg. 3.4.3	
		no order	$O(n^4k^{20}M(k))$ Alg. 3.4.4	

Table 3.2: Results for the variants of the ε-SD problem considered in this thesis. For symmetry group C_k we assume the partition of P into subsets of size k to be given.

3.7 The ε-SD Problem and Hypergraph Matching

In this section we will consider the relation between the ε-SD problem and the problem of finding a maximum matching in hypergraphs.
We start by stating the definition of hypergraphs and matchings in hypergraphs:

Definition 3.7.1. A hypergraph $H = (V, E)$ consists of a set of vertices V and a set of edges E, where each edge $e \in E$ is a subset of the vertex set, so $e \subseteq V$.
We call a hypergraph *k-uniform* iff $|e| = k$ for all $e \in E$.
A b-matching of a hypergraph H is a subset $M \subseteq E$ of the edges set where no vertex is contained in more than b edges.
A b-matching $M_{\max}$ of a hypergraph H is called *maximum b-matching* iff $|M_{\max}| \geq |M|$ for all b-matchings M of H.

In this section we will consider k-uniform hypergraphs and ask for a maximum 1-matching.

In Section 3.4 we solved the the ε-SD decision problem for symmetry group C_k in polynomial time in the case where a partition of the input set P was given.
One possibility to use this algorithm in order to solve the ε-SD decision problem for symmetry group C_k without given partition of P is to test all partitions of P into subsets of size k.
We formulate this approach by using the notion of hypergraphs and matchings in hypergraphs. The ε-SD decision problem for symmetry group C_k can be reduced to the problem of finding maximum 1-matchings in a k-uniform hypergraph as follows:

Lemma 3.7.2. *Let a point set $P \subset \mathbb{R}^2, |P| = mk$, a symmetry group C_k and a value $\varepsilon \geq 0$ be given.*
Construct a hypergraph $H = (V, E)$ as follows:
$V = P$ and an edge $e \in \binom{P}{k}$ is in E iff the answer to the ε-SD decision problem for symmetry group $C_{|e|}$ and input set e is YES.
Then the answer to the ε-SD decision problem for symmetry group C_k, input point set P and value ε is YES iff there is a maximum 1-matching $M_{max}, |M_{max}| = m$ so that the answer of the ε-SD decision problem for symmetry group C_k, the partition given by M_{max} and value ε is YES.

Proof. $\Rightarrow$ Suppose there is a C_k-symmetric point set Q which ε-approximates P. Then there is a partition of P into subsets $P_1, \ldots, P_m$ all of size k and a partition of Q into regular k-gons $Q_1, \ldots, Q_m$, all having the same rotation center c and Q_i ε-approximates P_i, for all $1 \leq i \leq m$. Thus there is an edge $e_i \in E$ defined by P_i contained in the hypergraph H, for all $1 \leq i \leq m$ and the set $M_{\max} = \{e_1, \ldots, e_m\}$ is a 1-matching of size m and thus a maximum 1-matching in H.
$\Leftarrow$ Suppose there is a maximum 1-matching $M_{\max}$ of size m in H and the answer to the ε-SD decision problem for symmetry group C_k, the partition of P defined by $M_{\max}$ and value ε is YES. Thus there is a C_k-symmetric point set Q which ε-approximates P and therefore the answer to the ε-SD decision problem is YES. $\square$

Remark 3.7.3. The problem of finding a maximum matching in a hypergraph for $k \geq 3$ is known to be NP-complete. This was already shown by Karp [22] in 1972.

Using this reduction we can state the following algorithm which solves the ε-SD decision problem for symmetry group C_k:

Algorithm 3.7.1 Algorithm solving the ε-SD decision problem for symmetry group C_k.

SDC$_\mathbf{k}$Hypergraph$(P = \{p_1, \ldots, p_n\}, \varepsilon)$
  $V = P$;
  //For each subset $e \subset P$ of size k...
  for $e \in \binom{P}{k}$ **do**
    //... test if a regular k-gon exists which ε-approximates e.
    if SDC$_{|\mathbf{e}|}$ (e, ε) **then**
      //If that is the case, add the subset e to the set of edges.
      $E = E \cup \{e\}$;
    end if
  end for
  //Test for each matching M if the partition of P given by M leads to a positive answer to the ε-SD problem.
  $\mathcal{M} =$**FindMaximumMatchings**$(H = (V, E))$;
  for $M \in \mathcal{M}$ **do**
    if SDC$_\mathbf{k}$Partition(M, ε) **then**
      return TRUE;
    end if
  end for
  return FALSE;

Theorem 3.7.4. *Let a point set $P \subset \mathbb{R}^2$, a symmetry group C_k and a value $\varepsilon \geq 0$ be given. Then Algorithm 3.7.1 solves the ε-SD decision problem in time $O\left(n^{(mk)} n^4 k^{20} M(k)\right)$, where $M(k)$ is the time needed to compute a maximum matching in bipartite graphs with k vertices.*

Proof. The correctness follows from the correctness of Algorithm 3.3.7 and Lemma 3.7.2.

The k-uniform graph constructed in the algorithm has $O(n^k)$ edges. Thus there are $O\left(\left(n^{mk}\right)\right)$ many matchings to be tested. Therefore the overall running time of this approach is $O\left(n^{mk} n^4 k^{20} M(k)\right)$. □

Using this algorithm is not practical for real world input point sets due to the running time.

Unfortunately, even computing a constant factor approximation of the maximum 1-matching in an k-uniform hypergraph is not possible in polynomial time unless P=NP, which was proven by Hazan et al. [14].

The high running time resolves not only from the problem of finding maximum matchings in a hypergraph but also from the high running time of Algorithm 3.4.4 which decides the ε-SD decision problem for symmetry group C_k with given partition of P. This is due to the fact that we do not know the rotation center of the C_k-symmetric point set Q.

In the remaining part of this section we will develop an algorithm which computes an acceptable solution for the ε-SD computation problem for symmetry group C_k and a realistic input set P. We assume the point set P to behave well w.r.t the considered symmetry group, which means that we assume the optimal value ε_{opt} to be small.

We can then achieve good result for the ε-SD computational problem for symmetry group C_k by using Algorithm 3.7.2.

The procedure **MinimumWeightMaximumMatching**$(H = (V, E))$ computes an approximation of a maximum cardinality matching for the hypergraph H and is given in Algorithm 3.7.3.

Theorem 3.7.5. *For a given weighted k-uniform hypergraph $H = (V, E)$ Algorithm 3.7.3 computes a 1-matching in time $O\left(|E|\left(\left(\frac{|V|}{k}\right) + \log|V|\right)\right)$.*

Proof. The computed set M is a 1-matching of the hypergraph H since each vertex is contained in at most one edge in M by construction. After adding an edge to the set of matching edges, all edges incident to any vertex in e are deleted from E.

The while loop is iterated $O\left(\frac{|V|}{k}\right)$ times since in each iteration k vertices are covered by an edge and only edges not incident to any covered vertex are left in E. Finding the vertex with minimum degree can be done in $O(1)$ time by using a heap. Decreasing the degree of a vertex when deleting edges form the edge set takes $O(\log|V|)$ time. Choosing the edge with smallest weight of a vertex v can be done in time $O(|E|)$ by using a doubly linked list for the edges. Deleting an edge from that list takes time $O(1)$. Thus the overall running time is $O\left(\left(\frac{|V|}{k}\right)(1 + |E|) + |E|\log|V| + |E|\right) = O\left(|E|\left(\left(\frac{|V|}{k}\right) + \log|V|\right)\right)$. □

Remark 3.7.6. For realistic inputs we do not assume any more that the number of elements in the point set is a multiple of the symmetry group number k. Since the matching algorithm stops when all edges are deleted from the set E, the supernumerous points are ignored. In Figure 3.7.1 the input set in (a) consists of 43 points. Since we search for a C_8-symmetric point set, 3 points are left over and are not mapped to any vertex of the C_8-symmetric point set Q depicted in (b). In (c) the green points are those points from the original point set which are mapped to a point of Q. The radii of the circles indicate the weight of the corresponding matching edges.

Algorithm 3.7.2 Algorithm computing an approximation of the solution of the ε-SD computational problem for symmetry group C_k.

SDApproximation$(P = \{p_1, \ldots, p_n\}, k)$
  //Use the center of mass as rotation center.
  $s = \frac{1}{n}\sum_{i=1}^{n} p_i$;
  $\alpha = \frac{2\pi}{k}$
  $V = P$;
  $E = \emptyset$;
  for $i = 1$ to n **do**
    //Compute regular k-gon defined by p_i and s.
    $Q_i = \{q_{(i,j)} = \rho_s^{(j-1)\alpha}(p_i) | 1 \leq j \leq k\}$;
    //ε_i is the minimum value s.t. a point of P is contained in each disk of radius ε_i centered at the vertices of Q_i.
    $\varepsilon_i = \max_{q \in Q_i} \min\{d(q,p) | p \in P\}$;
    for $j = 1$ to k **do**
      //The set $P_{(i,j)}$ contains all points of P with distance smaller or equal to ε_i to the j^{th} vertex of Q_i.
      $P_{(i,j)} = \{p \in P | d(q_{(i,j)}, p) \leq \varepsilon_i\}$;
    end for
    //Add all possible combinations of points in the sets $P_{(i,1)}, \ldots, P_{(i,k)}$ to the set of edges.
    $E = E \cup \{\{p_{(i,1)}, \ldots, p_{(i,k)}\} | p_{(i,j)} \in P_{(i,j)}, 1 \leq j \leq k\}$;
    //The weight of the edge is the solution of the ε-SD computational problem for symmetry group D_k and given rotation center s applied to the points in the edge.
    $w(\{p_{(i,1)}, \ldots, p_{(i,k)}\})$ =**SDD$_\mathbf{k}$Center**$(\{p_{(i,1)}, \ldots, p_{(i,k)}\}, s)$;
  end for
  M =**MinimumWeightMaximumMatching**$(H = (V, E))$;
  return $\max\{w(e) | e \in M\}$;

The running time of Algorithm 3.7.2 depends on the behavior of the input set P. The more points of P are contained in the ε-neighborhoods of $k-1$ vertices of the regular k-gon defined by s and a point of P, the more edges are in the hypergraph. Therefore the construction of the hypergraph and especially the algorithm finding a 1-matching take more time than in the case where only few points of P lie in the ε-neighborhoods of the vertices of the regular k-gons.

The aim of Algorithm 3.7.2 is to construct a C_k-symmetric point set which is a good approximation of the input set. As an application of this algorithm think of a picture of a figure with symmetry group C_k which contains noise due to certain preprocessing steps. For examples consider a pipeline which converts a picture of an object to a .jpg image and then to a polygonal curve. The result of Algorithm 3.7.2 applied to the vertices of the polygonal curve is then a C_k-symmetric polygonal

Algorithm 3.7.3 Algorithm for approximating the maximum cardinality matching with minimum weight for hypergraph H.

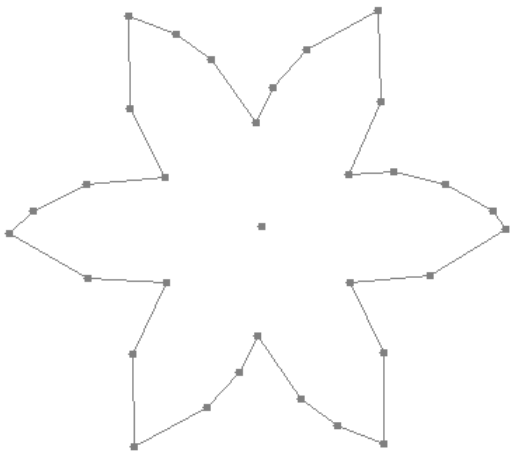
MinimumWeightMaximumMatching$(H = (V, E))$
  $M = \emptyset$;
  while $E \neq \emptyset$ **do**
    Choose vertex v with smallest degree;
    Choose edge e incident to v with minimum weight;
    $M = M \cup \{e\}$;
    Remove e and all other edges incident to any vertex in e from E;
  end while
  return M;

representation of the original symmetric figure. In order to get good results, we assume that the symmetry group considered in Algorithm 3.7.2 is the one of the original figure.

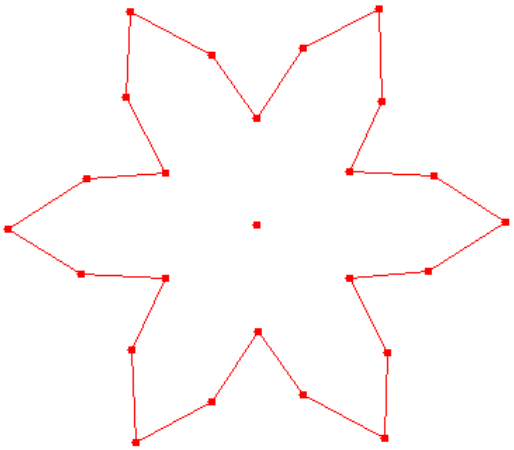
For a given point set P which results from an originally symmetric figure but is slightly distorted due to some processing steps, we can construct a symmetric point set which is a close approximation of P by combining the algorithms presented in Chapter 2 with the one presented in this section. The algorithms presented in Chapter 2 detect the symmetry group of a point set or an image even if it contains noise. Thus we can apply these algorithms in order to detect the symmetry group which we then use in Algorithm 3.7.2 in order to construct the symmetric point set Q.

A program using this procedure in order to detect a symmetry group of an object represented by a set of points or straight line segments and approximate it by an object having the detected symmetry group was implemented by using the language JAVA. The program is attached to this thesis.

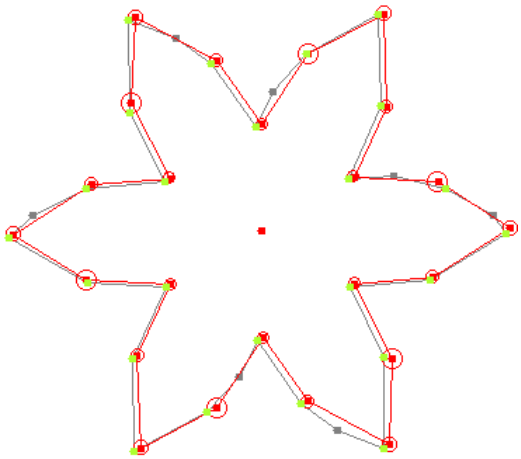
The point sets in Figure 3.7.2 are constructed using the JAVA-program.



(a) Input point set P.

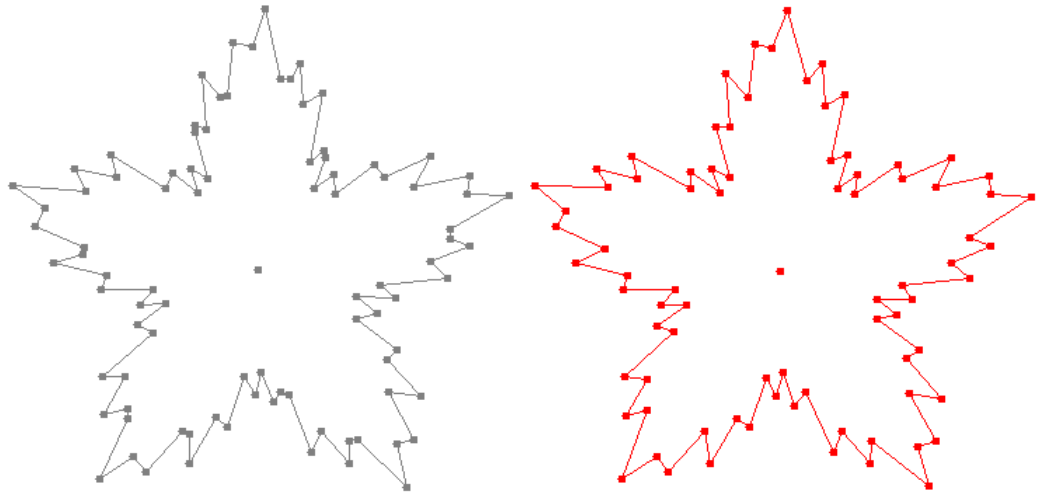


(b) C_6-symmetric point set Q.

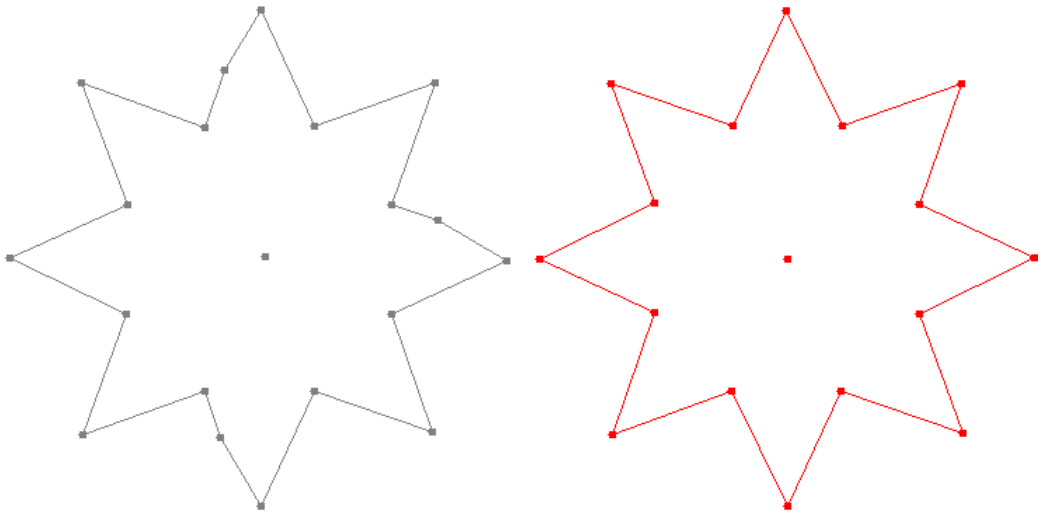


(c) The green points are those point of the input point set P which are mapped to a point of the point set Q.

Figure 3.7.1: Example of symmetric approximation.



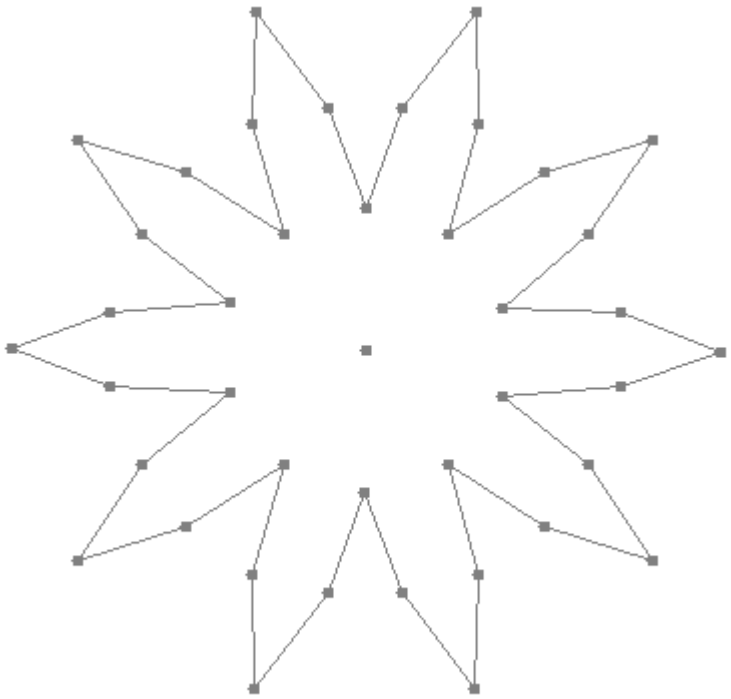
(a) Original point set (b) C_5-symmetric point set



(c) Original point set (d) C_8-symmetric point set

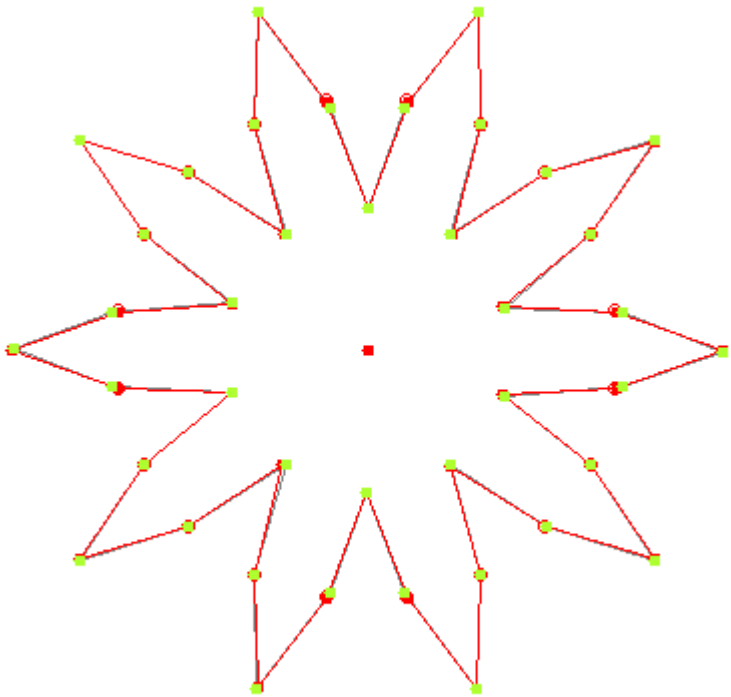
Figure 3.7.2: Examples of point sets and their symmetric reconstruction.

The last example is a very interesting one. It shows how misleading the human perception is. Let us consider the following point set:



(a) Original point set

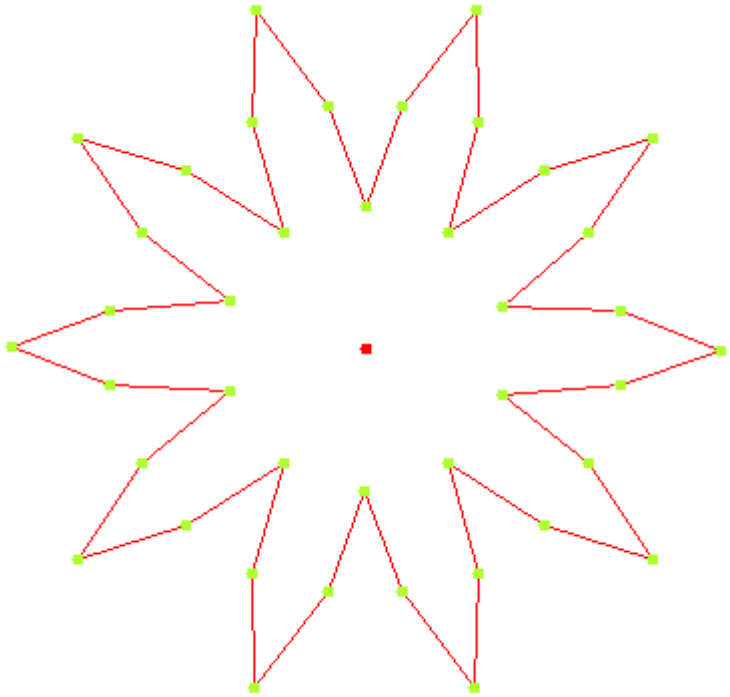
By looking at the point set one would consider it to be D_{10}-symmetric. And indeed this is one of the answers with a large number of votes in the probabilistic algorithm. The approximation is quite close as can be seen in the following figure:



(b) Approximating point set with small value of ε, which is indicated by the disks.

Figure 3.7.3: Example where the human perception is misleading.

The symmetry group with the largest number of votes, however, is C_2. And indeed this is the correct answer since the input set itself is C_2-symmetric as the approximation shows:



(c) The input point set itself is C_2-symmetric since the value of ε of the approximating set is zero.

Bibliography

[1] D. Alberts and M. R. Henzinger. Average case analysis of dynamic graph algorithms. In *Proceedings of the 6th Symposium on Discrete Algorithms*, pages 312–321, 1995.

[2] H. Alt and L. Scharf. Shape matching by random sampling. In S. Das and R. Uehara, editors, *3rd Annual Workshop on Algorithms and Computation (WALCOM 2009)*, volume 5431 of *Lecture Note in Computer Science*, pages 381–393. Springer-Verlag Berlin Heidelberg, 2009.

[3] H. Alt, N. Blum, K. Mehlhorn, and M. Paul. Computing a maximum cardinality matching in a bipartite graph in time $O(n^{1.5}\sqrt{m/\log n})$. *Information Processing Letters*, 37(4):237–240, 1991. ISSN 0020-0190. doi: http://dx.doi.org/10.1016/0020-0190(91)90195-N.

[4] N. M. Amato, M. T. Goodrich, and E. A. Ramos. Computing the arrangement of curve segments: Divide-and-conquer algorithms via sampling. In *In Proc. 11th ACM-SIAM Sympos. Discrete Algorithms*, pages 705–706, 1999.

[5] E. M. Arkin, C. Dieckmann, C. Knauer, J. S. B. Mitchell, V. Polishchuk, L. Schlipf, and S. Yang. Convex transversals. In *Proceedings of the 12th international conference on Algorithms and data structures*, WADS'11, pages 49–60, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-22299-3. URL `http://dl.acm.org/citation.cfm?id=2033190.2033195`.

[6] M. Atallah. On symmetry detection. *IEEE Transactions on Computers*, 34(7): 663–666, July 1985.

[7] S. Basu, R. Pollack, and M.-F. Roy. On computing a set of points meeting every cell defined by a family of polynomials on a variety. *Journal of Complexity*, 13 (1):28–37, Mar. 1997. ISSN 0885-064X. doi: 10.1006/jcom.1997.0434. URL `http://dx.doi.org/10.1006/jcom.1997.0434`.

[8] L. G. Brown. A survey of image registration techniques. *ACM Computing Surveys*, 24(4), 1992.

[9] W. M. Brown, M. E. Price, J. Kang, N. Pound, Y. Zhao, and H. Yu. Fluctuating asymmetry and preferences for sex-typical bodily characteristics. *Proceedings of*

the National Academy of Sciences of the United States of America, 105(35):pp. 12938–12943, 2008. ISSN 00278424.

[10] J. W. Cooley and J. W. Tukey. An algorithm of the machine calculation of complex Fourier series. *Mathematics of computation*, 19:297–301, 1965.

[11] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. The MIT Press, 1994.

[12] S. Derrode and F. Ghorbel. Shape analysis and symmetry detection in gray-level objects using the analytical Fourier-Mellin representation. *Signal Processing*, 84: 25–39, January 2004. ISSN 0165-1684.

[13] A. Efrat, A. Itai, and M. J. Katz. Geometry helps in bottleneck matching and related problems. *Algorithmica*, 31(1):1–28, 2001.

[14] E. Hazan, S. Sara, and O. Schwartz. On the complexity of approximating k-set packing. *Computational Complexita*, 15(1):20–39, 2006.

[15] J. Hopcroft and R. M. Karp. An $n^{(5/2)}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal Computing*, 2(4):225–231, 1973.

[16] A. Imiya and I. Fermin. Voting method for planarity and motion detection. *Image and Vision Computing*, pages 867–879, 1999.

[17] A. Imiya, T. Ueno, and I. Fermin. Planar symmetry detection by random sampling and voting process. In *Proceedings of the Joint IAPR International Workshops on Advances in Pattern Recognition*, pages 347–356, London, UK, 2000. Springer-Verlag. ISBN 3-540-67946-4.

[18] S. Iwanowski. *Testing approximate symmetry in the plane is NP-hard*. PhD thesis, 1991.

[19] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. 1999.

[20] B. Johansson, H. Knutsson, and G. Granlund. Detecting rotational symmetries using normalized convolution. In *Proceedings of the 15th international conference on pattern recognition*, pages 500–504, 2000.

[21] S. C. Johnson. Hierarchical clustering algorithm. *Psychometrika*, 1967.

[22] R. M. Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations*, 1972.

[23] Y. Keller and Y. Shkolnisky. A signal processing approach to symmetry detection. *IEEE Transactions on Image Processing*, 15:2198 – 2207, 2006.

[24] D. E. Knuth, J. H. Morris, and V. R. Pratt. Fast pattern matching in strings. *SIAM Journal of Computing*, 6,2:323–350, 1977.

[25] Y. Liu and R. T. Collins. Frieze and wallpaper symmetry groups classification under affine and perspective distortion. Technical report, 1998.

[26] Y. Liu and R. T. Collins. A computational model for repeated pattern perception using frieze and wallpaper groups. In *Computer Vision and Pattern Recognition Conference*, pages 537–544. IEEE Computer Society Press, 2000.

[27] G. E. Martin. *Transformation Geometry.* Springer, 1982.

[28] T. Mateer. *Fast Fourier Transform Algorithms with Applications.* PhD thesis, Clemson University, 2008.

[29] T. Matsuyama, S.-I. Miura, and M. Nagao. Structural analysis of natural textures by Fourier transform. *Computer Vision, Graphics and Image Processing*, 24:347–362, 1983.

[30] N. Megiddo. Linear-time algorithm for linear programming in $\mathbb{R}^3$ and related problems. *SIAM Journal of Computing*, 1983.

[31] N. Megiddo. The weighted Euclidean 1-center problem. *Mathematics of Operations Research*, 8(4), 1983.

[32] S. Micali and V. Vazirani. An $O(\sqrt{|V|}|E|)$ algorithm for finding maximum matching in general graphs. In *Proceedings of the 21st IEEE Symposium on Foundations of Computer Science.*

[33] N. J. Mitra, L. J. Guibas, and M. Pauly. Partial and approximate symmetry detection for 3d geometry. *ACM Transactions on Graphics*, 25:560–568, July 2006. ISSN 0730-0301.

[34] N. J. Mitra, L. J. Guibas, and M. Pauly. Symmetrization. *ACM Transactions on Graphics*, 26, July 2007. ISSN 0730-0301.

[35] M. Reichling. On the detection of a common intersection of k convex objects in the plane. *Information Processing Letters*, 29:25–29, 1988.

[36] A. Rosenfeld and A. C. Kak. *Digital Picture Processing.* Academic Press, Inc., Orlando, FL, USA, 2nd edition, 1982. ISBN 0125973020.

[37] J.-R. Sack and J. Urrutia. *Handbook of computational geometry.* North-Holland Publishing Co., Amsterdam, The Netherlands, The Netherlands, 2000. ISBN 0-44482-537-1.

[38] L. Scharf. *Probabilistic Matching of Planar Shapes.* PhD thesis, Freie Universität Berlin, Institut für Informatik, June 2009.

[39] D. Shen, H. H. S. Ip, K. K. T. Cheung, and E. K. Teoh. Symmetry detection by generalized complex (gc) moments: A close-form solution. *IEEE Transactions*

on Pattern Analysis and Machine Intelligence, 21:466–476, May 1999. ISSN 0162-8828.

[40] H. Weyl. *Symmetry*. Princeton University Pr., 1980.

[41] J. D. Wolter, T. C. Woo, and R. A. Volz. Optimal algorithms for symmetry detection in two and three dimensions. *The Visual Computer*, 1985.

[42] H. Zabrodsky, S. Peleg, and D. Avnir. Continuous symmetry for shapes. In *2nd International Workshop on Visual Form*, pages 594–613, 1994.

Erklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Berlin, den 4. Februar 2013 Claudia Dieckmann

Curriculum Vitae

For reasons of data protection, the curriculum vitae is not included in the online version.