On the Computational Complexity of Projective Image Matching

Christian Hundt

Dissertation

Universität zu Lübeck
Institut für Theoretische Informatik

Aus dem Institut für Theoretische Informatik
der Universität zu Lübeck
Direktor: Prof. Dr. Rüdiger Reischuk

On the Computational Complexity of Projective Image Matching

Inauguraldissertation

zur
Erlangung der Doktorwürde
der Universität zu Lübeck

Aus der Sektion Informatik / Technik

vorgelegt von

Christian Hundt
aus Rostock

Lübeck, März 2011

1. Berichterstatter:	PD Dr. Maciej Liśkiewicz
2. Berichterstatter:	Prof. Dr. Marek Karpinski
3. Berichterstatter:	Prof. Dr. Wojciech Rytter

Tag der mündlichen Prüfung: 08. Juli 2011

Zum Druck genehmigt. Lübeck, den 12. Januar 2012

Acknowledgments

I would like to thank my *Doktorvater* Maciej Liśkiewicz who introduced me to the vast field of image matching. For six years he gave me continuous support and valuable advice, which made this thesis possible in the first place.

I am also much obliged to Rüdiger Reischuk, Till Tantau, and Andreas Brandstädt for providing the opportunity to work at the excellent computer science departments of the universities of Lübeck and Rostock. My current and former colleagues of both departments have my deep gratitude for many valuable discussions. Special thanks go to Ragnar Nevries, Stefan Goldmann, and Ulf Adams for proofreading.

Finally, I am deeply indebted to my parents Norbert and Petra Hundt and my fiancée Eva Rosenke for their indispensable encouragement, for helping me outside university life, and for their understanding.

Abstract

Image matching is an important problem in image processing and arises in such diverse fields as video compression, optical character recognition, medical imaging, watermarking and others. Given two digital images A and B, image matching determines a transformation f for A such that it most closely resembles B. Conventional approaches require either exponential time, or find only an approximate solution, even when f has to consist of only rotations and scalings.

This thesis introduces the first known discretization technique for the class $\mathcal{F}_{\mathtt{p}}$ of projective transformations. Based on this, it shows a polynomial bound on the cardinality of $\mathcal{D}(A)$, the set containing all different transformations of A with $f \in \mathcal{F}_{\mathtt{p}}$. Accordingly, structural properties of $\mathcal{D}(A)$ lead to a polynomial time algorithm that finds the exact solution to the projective image matching problem by searching through the complete set $\mathcal{D}(A)$. The algorithm is applicable for various natural subclasses of projective transformations, such as e. g. affine transformations.

From a theoretical point of view providing a polynomial time algorithm for projective image matching is not enough to completely characterize the complexity of the problem. Accordingly, TC_O, a class of problems deep in the hierarchy within *PTIME*, is shown to exactly describe the complexity of image matching with projective transformations. This relates the problem to a number of most basic challenges in computer science, like integer multiplication, division and sorting. Beyond this, the containment in TC_O implies extremely efficient parallel algorithms for image matching.

Contents

Chapter 1

Introduction

Image matching is an important problem in image processing because it arises in various practical applications. Given two digital images A and B, image matching determines a transformation f for A such that it most closely resembles B. In most settings f is required to be an invertible function $\mathbb{R}^2 \to \mathbb{R}^2$ contained in a given class $\mathcal{F}$.

The research in image matching is motivated by a computational challenge that occurs in many practical applications. In fact, finding the similarity between objects embedded in digital images is a complex task which is often approached by image matching methods. The reason for this is that objects in digital images are often exposed to geometric distortions occurring when the image is created or due to some subsequent image processing. Image matching can be used to compensate such distortions.

MPEG video compression [42] provides a typical scenario of this kind. Videos consist of many images representing projections of the same real world objects. The challenge is to reduce the redundant representation by finding the similarity between successive images A and B in the video stream. For this purpose it is necessary to compensate distortions introduced due to object and camera movements. See e. g. [42] and the references therein.

In computer vision, the problem is to detect the representations of a certain object, like a Latin letter [29] or a human silhouette [38], in a given image B. To realize this, it is possible to measure the similarity of B relative to reference representations of these objects in another image A. Again, it is usually the case that the object representations in B undergo geometric distortions that have to be compensated.

One challenge in medical imaging is to correlate two images A and B of the same medical object taken in different times or using different medical image devices. The difficulty of realizing identical perspectives in both recordings makes it necessary to compensate the accordingly introduced distortions. See e. g. [34].

Finally, image matching can be applied for robust digital watermarking [14, 43]. The goal of watermarking is to embed an imperceptible pattern A into an image B to resolve certain legal issues. If B is exposed to geometric distortions, e. g., by a removal attack or the print-and-scan-process, it still has to be possible to prove that it contains the pattern A. This purpose requires distortion compensation before cross correlating B with a reference of A.

In such applications the class $\mathcal{F}$ models the specific set of possible distortions. Often, it is a subclass of projective transformations $\mathcal{F}_{\mathsf{p}}$ because they describe natural geometric basic operations like scaling, rotating, translating and shearing. This motivates research in image matching under projective transformations.

Despite the various fields of application the analysis of image matching under projective transformations is virtually uncharted territory. Applying standard image matching approaches for this class requires either exponential time or finds only an approximate solution. Research in combinatorial pattern matching has succeeded in solving the image matching problem in polynomial time only for very small subsets of projective transformations like rotation or scaling [4, 5]. But there has been very little progress in understanding the discrete nature of projective image transformations which ulteriorly form the foundation of the subcase's tractability.

This thesis pinpoints the computational complexity of finding the exact solution of image matching under projective transformations. Particularly, the problem belongs to TC_O, a class that exactly captures the complexity of a variety of basic operations in computation, such as integer addition, multiplication and sorting.

The complexity result builds on a new structural characterization of projective image transformations. In fact, the class $\mathcal{F}_{\mathsf{p}}$ of all projective transformations is uncountable. On the other hand, all digital images with a designated size $m \times m$ form a finite set. But, in image matching it is sufficient to consider only those transformations $f(A)$ of the given image A which have the same size $m \times m$ as the other given image B. Then, transforming the image A by all possible $f \in \mathcal{F}_{\mathsf{p}}$ results in a subset of images with size $m \times m$. Consequently, A and m define a partition of $\mathcal{F}_{\mathsf{p}}$ into a finite amount of equivalence classes where every class consists of transformations f with the same result image $f(A)$.

The central achievement of this thesis is a discretization technique to identify equivalence class representatives $\{f_1, f_2, \dots, f_r\}$, which forms a finite subset of $\mathcal{F}_{\mathsf{p}}$ sufficient to generate all possible outcomes of transforming A into an image of size $m \times m$. This implies a solution to image matching instances that consist of A and any given image B of size $m \times m$, because the transformation of A which is closest to B can be found in $f_1(A)$ to $f_r(A)$.

The value of r depends on m and the size of A. Based on the discretization, combinatorial estimation shows that r grows polynomially in its arguments. Subsequently, the newly discovered structural properties also lead to the first polynomial time algorithm for the aforementioned class of projective transformations. Beside this class, the algorithm works for image matching under many subclasses in a generic fashion, respectively in less time.

The presentation in this thesis is structured as follows: The rest of this chapter gives a short overview on previous work and an informal, yet more detailed, discussion of the author's contribution. Then, the next chapter provides the preliminary notation needed to understand the used mathematical concepts and to follow the ideas.

Chapter 3 presents structural properties of projective image transformations and describes the equivalence class partition of $\mathcal{F}_{\text{p}}$. Based on the new characterization of projective image transformations, Chapter 4 introduces a new generic polynomial time image matching algorithm. This algorithm is not only applied for projective image matching, but also for various subclasses which all introduce their own technical difficulties.

Chapter 5 regards the complexity class of projective image matching. This means that instead of concentrating only on time consumption, the chapter determines the lowest class that contains the image matching problem. In fact, this works in two steps. The first one shows that the problem is contained in TC_O and the second provides the completeness in this class, which means that membership in a lower complexity class $\mathcal{K}$ implies $\mathcal{K} = TC_O$.

Finally, Chapter 6 extends the results to the closely related two-dimensional pattern matching. Moreover, the chapter highlights future directions of work.

In all chapters technical proofs are moved to a separate section at the chapter's end to improve readability and understanding.

1.1 Previous Work

In many practical applications, especially in medical imaging [36], image matching is solved by looking at the images A and B in an analogue way and interpreting them as continuous mappings $a : \mathbb{R}^2 \to \mathbb{R}$ and $b : \mathbb{R}^2 \to \mathbb{R}$ over the real plane. Typically a and b are derived from A and B by some spline interpolation. Subsequently, the function $\Delta(f, A, B) = \int_{(x,y)\in\mathbb{R}^2} \|a(f(x, y)) - b(x, y)\| \, dx \, dy$ describes the difference between the transformation of A and B for a given transformation $f \in \mathcal{F}$. The image match is found by $\arg\min_{f\in\mathcal{F}}\{\Delta(f, A, B)\}$, i. e., by minimizing Δ over the set of transformations. In this technique, called *image registration* [36], the problem is reformulated to fit the continuous nature of $\mathcal{F}$ and to be solved by techniques based on continuous analysis.

Such methods have proven successful in applications. However, they guarantee achieving only local optima. The disadvantage of this approach is the hardness of finding the global optimum due to the continuous nature of the considered objects. This means also that analyzing this setting may not easily uncover the discrete structural properties of projective transformations on digital images.

Another approach bases on feature matching techniques [7, 32]. Such methods are used in computer vision and work in two steps. The first one extracts salient features, geometrical objects like points, lines, regions etc., from the images A

and B. The second step tries to find $f \in \mathcal{F}$ that transforms the features extracted from A closest to those of B. Although being successfully applied in practical settings, feature extraction complicates the analysis of the discrete nature of image transformations like the previous approach.

Moreover, beside the technical difficulties introduced by feature extraction, feature matching is a highly non-trivial task that remains difficult even for points. A survey on this well studied problem, also called geometric point set matching, is given by Indyk [27]. Kenyon et al. [30] and Papadimitriou et al. [41] present some related problems. In fact, known algorithms for point set matching give only approximate solutions and finding the global optimum is an involved task even for simple classes of transformations as compositions of rotation and translation [28]. Interestingly, Indyk et al. [28] give a discretization technique to reduce the geometric point set matching problem to an image matching like problem.

There are various other continuous approaches applied in the image processing community. An overview and discussions can be found in the books by Modersitzki [36, 37] and in the survey by Brown [9] as well as in the references therein.

Motivated by the issues caused by modeling image matching with continuous methods concentrating on discrete aspects of image transformations leads to new insights into the combinatorial properties of the problem. In fact, research in combinatorial pattern matching recently directs attention to the analysis of discrete structures for the transformation sets $\mathcal{F}$. The pattern matching problem in focus is closely related to image matching although it has a different origin [33]. Instead of searching for a best match between A and B, pattern matching looks for a transformations $f \in \mathcal{F}$ such that $f(A)$ occurs as an exact copy in B. Despite some slight differences in problem formalization many results for pattern matching can be easily transferred to the image matching problem and vice versa. Accordingly, first polynomial time algorithms finding exact globally optimal solutions to the image matching problem for some very small subclasses of projective transformations are derived from results in combinatorial pattern matching [3, 18]. Apart from these algorithmic achievements, improved techniques for the analysis of image matching complexity have been developed.

In particular, the observation that there are transformations in $\mathcal{F}$ that, although mathematically different, lead to the same transformed version of a given image A, inspires the idea of computing the dictionary $\mathcal{D}[\mathcal{F}](A)$ containing all images $f(A)$ which result from applying all transformations $f \in \mathcal{F}$ to the image A. Essentially, all algorithms computing an exact or globally best match $f(A)$ with B share the same basic idea, to perform exhaustive search of the entire dictionary $\mathcal{D}[\mathcal{F}](A)$. The main challenge is to find a suitable discretization for the set $\mathcal{F}$.

Amir et al. [3] discovered the first discretization of the continuous space of scalings $\mathcal{F}_{\mathsf{s}}$. Using purely combinatorial techniques, they show for all images A with size $n \times n$ that the dictionary $\mathcal{D}[\mathcal{F}_{\mathsf{s}}](A)$ contains $O(n^2)$ transformed images of the same size $n \times n$. Based on their new discretization of $\mathcal{F}_{\mathsf{s}}$ a simple $O(n^4)$

time algorithm solving image matching under scaling is easily found. First, this algorithm computes all $O(n^2)$ scaled images of the dictionary and then it searches for the best match with B, simply by evaluating the distortion between every image $A' \in \mathcal{D}[\mathcal{F}_{\mathtt{s}}](A)$ against B, which takes $O(n^2)$ time each. A fast algorithm for image matching with scaling can immediately be derived from a result of Amir and Chencinski [4] running in time $O(n^3)$.

Fredriksson and Ukkonen [18] are the first who find a discretization for rotations $\mathcal{F}_{\mathtt{r}}$. For images A with size $n \times n$ they show that the dictionary $\mathcal{D}[\mathcal{F}_{\mathtt{r}}](A)$ consists of $O(n^3)$ rotated images. This provides a straightforward $O(n^5)$ time algorithm for image matching under rotations, simply by computing the whole dictionary and comparing all $O(n^3)$ elements with B in $O(n^2)$ time each.

After a series of algorithmic improvements [5, 17] that base on the discretization technique of Fredriksson and Ukkonen [18], Nouvel and Rémila [40] provide a technique to incrementally compute all images in $\mathcal{D}[\mathcal{F}_{\mathtt{r}}](A)$. Particularly, $\mathcal{D}[\mathcal{F}_{\mathtt{r}}](A)$ can be ordered linearly such that computing an image from the preceding one takes only very few updates. This means that, instead of first computing the complete dictionary, they benefit from the similarity between succeeding images and iterate through $\mathcal{D}[\mathcal{F}_{\mathtt{r}}](A)$ comparing one image after the other against B. Applying this incremental strategy image matching allowing rotations can be done in $O(n^3 \log n)$ time.

On the other hand, image matching was shown to be NP-complete for elastic transformations by Keysers and Unger [31]. But, compared to scalings or rotations, elastic transformation represent a considerably more complex class. It remains a great open problem to describe the complexity of image matching for transformations between these two cases.

Actually, it was even open how to generalize the discretizations for $\mathcal{F}_{\mathtt{s}}$ and $\mathcal{F}_{\mathtt{r}}$ into a characterization for the set $\mathcal{F}_{\mathtt{sr}}$ of all combinations of scaling and rotation. Whereas the commutative concatenation of $\mathcal{F}_{\mathtt{s}}$ and $\mathcal{F}_{\mathtt{r}}$ leads to $\mathcal{F}_{\mathtt{sr}}$ it is the case that the sets $\bigcup_{A' \in \mathcal{D}[\mathcal{F}_{\mathtt{s}}](A)} \mathcal{D}[\mathcal{F}_{\mathtt{r}}](A')$ and $\bigcup_{A' \in \mathcal{D}[\mathcal{F}_{\mathtt{r}}](A)} \mathcal{D}[\mathcal{F}_{\mathtt{s}}](A')$ differ essentially from $\mathcal{D}[\mathcal{F}_{\mathtt{sr}}](A)$. The problem is that the common foundation of the discretizations for $\mathcal{F}_{\mathtt{s}}$ and $\mathcal{F}_{\mathtt{r}}$ have not been understood well enough.

Recently, the author and Maciej Liśkiewicz [23, 24, 25] solve this open problem by giving a combinatorial geometric discretization for image transformations combining scaling and rotation. Noticeably, combinatorial geometry as a tool for problems like image matching was first used by by Amir et al. [2]. The new characterization provides the fastest known algorithms for image matching with $\mathcal{F}_{\mathtt{sr}}$ and $\mathcal{F}_{\mathtt{r}}$, i. e., running in $O(n^3)$ time for $\mathcal{F}_{\mathtt{r}}$ [25] and in $O(n^6)$ time for $\mathcal{F}_{\mathtt{sr}}$ [26].

Moreover, the author and Maciej Liśkiewicz show, for the first time, lower worst case bounds for the cardinalities of the dictionaries $\mathcal{D}[\mathcal{F}_{\mathtt{r}}](A)$ and $\mathcal{D}[\mathcal{F}_{\mathtt{sr}}](A)$. In fact, they describe families A_n of images for which $|\mathcal{D}[\mathcal{F}_{\mathtt{r}}](A_n)| \in \Omega(n^3)$ and $|\mathcal{D}[\mathcal{F}_{\mathtt{sr}}](A_n)| \in \Omega(n^6/\log n)$ [25]. This means that image matching by exhaustive search on dictionaries cannot be solved essentially faster as performed by the algorithms introduced in [23, 24, 25, 26].

Then, the author and Maciej Liśkiewicz [22] give polynomial time image matching algorithms for the currently largest transformation classes, i. e., for projective, affine and linear transformations. Moreover, beside the plain algorithmic advances, the author [21] applies their characterization of $\mathcal{D}[\mathcal{F}_\mathsf{a}](A)$ to describe the exact complexity of image matching under affine transformations. He shows that affine image matching is TC_O-complete. Hence, the complexity of image matching can be compared to various most basic problems in computation such as integer arithmetic.

1.2 Thesis Contribution

The main objective of this thesis is to introduce a general polynomial time framework to image matching for projective transformations and its subclasses. In contrast to the unique methods presented in the individual papers [22, 23, 24, 25, 26] the generic algorithm developed in Chapter 4 works out of the box for numerous classes, like affine and linear transformations, translations, scalings, rotations, their combinations and many others. This section gives a short outline of the established methods and results.

The generic approach works on basis of a characterization of the parameter space of projective transformations. Particularly, every transformation $f \in \mathcal{F}_\mathsf{p}$ is defined by

$$f(x, y) = \left(\frac{xp_1 + yp_2 + p_3}{xp_7 + yp_8 + 1}, \frac{xp_4 + yp_5 + p_6}{xp_7 + yp_8 + 1} \right)$$

where $(p_1, \dots, p_8)^T \in \mathbb{R}^8$. Vice versa, every point $P = (p_1, \dots, p_8)^T$, that renders the denominator in the equation non-zero, defines a projective transformation and consequently, $\mathbb{R}^8$ can be seen as a parameter space for projective transformations. Thus, considering an image A there is a natural correspondence between the point P and the transformed image $f(A)$ given by the transformation f defined by P. According to the observation that the number of transformations, and thus, points, is much bigger than the number of images, there have to be many points P corresponding to the same transformation of A. The discretization describes a partition of the parameter space $\mathbb{R}^8$ into a polynomial amount of subspaces $\varphi_1, \dots, \varphi_r$ such that any pair of points P and P' of the same subspace φ are related to the same transformed version of A.

Hence, selecting representative points $P_1, \dots, P_r$ from all r subspaces is equivalent to finding a polynomial size subset $\{f_1, \dots, f_r\}$ of $\mathcal{F}_\mathsf{p}$ that is sufficient to render the complete dictionary $\mathcal{D}[\mathcal{F}_\mathsf{p}](A)$ of all images gained by transforming A with projective transformations. Using $\mathcal{D}[\mathcal{F}_\mathsf{p}](A)$ it is possible to answer image matching queries involving A and another image B simply by comparing all images of the dictionary with B.

To be a feasible method for image matching, it is fundamental that the partition $\{\varphi_1, \ldots, \varphi_r\}$ of the parameter space $\mathbb{R}^8$ can be computed efficiently. Chapter 3 shows that the partition of the parameter space is determined by a set $\{h_1, \ldots, h_t\}$ of hyperplanes. Because every hyperplane cuts $\mathbb{R}^8$ into two subspaces the partition $\varphi_1, \ldots, \varphi_r$ is formed by the intersections of the cuttings defined by all hyperplanes $h_1, \ldots, h_t$. Such a partition is called hyperplane arrangement in combinatorial geometry and according to the practice in this field of research the subspaces $\varphi_1, \ldots, \varphi_r$ are called faces.

The proposed algorithm takes advantage of the geometric parameter space characterization. In fact, every subset $\mathcal{F}$ of projective transformations forms a point subspace χ in $\mathbb{R}^8$. Then, to solve image matching for $\mathcal{F}$, it is possible to enumerate the faces $\varphi_1, \ldots, \varphi_r$. For every face φ_i that intersects χ the algorithm computes a representative point P_i to finally obtain all image $f_i(A)$ obtained by transforming a given image A with transformations in $\mathcal{F}$.

The polynomial running time of the proposed matching method is argued by the help of geometrical properties of hyperplane arrangements. Precisely, it is shown that the number r of faces $\{\varphi_1, \ldots, \varphi_r\}$ in the hyperplane arrangement is polynomial. Consequently, the number of enumerated transformed images is polynomial for every subset $\mathcal{F}$ of projective transformations. Moreover, research in combinatorial geometry provides efficient methods and data structures to compute and represent all faces defined by the given hyperplanes.

Applying the straightforward image matching approach of preprocessing the dictionary $\mathcal{D}[\mathcal{F}](A)$ for the subclass $\mathcal{F}$ and then comparing every transformed version A' in $\mathcal{D}[\mathcal{F}](A)$ against an image B of size m involves an $O(m^2)$ overhead for the comparison operation. Similarly to Nouvel and Rémila's characterization [40], the analysis of projective transformations and their geometric properties reveals a linear ordering of $\mathcal{D}[\mathcal{F}](A)$ that allows an incremental way of computing a transformed image from the preceding one by only very few updates. In fact, traversing the faces in an order implied by their geometrical incidence results in minimal updates between successively iterated images.

The presented algorithm uses only integer arithmetic which means that no numerical difficulties occur due to the use of floating point arithmetic.

The existence of a polynomial time algorithm for projective image matching alone cannot answer the open question whether projective image matching belongs to a complexity class that is structurally "easier" than the class *PTIME* of problems decidable in polynomial time. Particularly, the question is whether it corresponds to a complexity class in the hierarchy inside *PTIME*:

$$AC_O \subset TC_O \subseteq NC_1 \subseteq LOGSPACE \subseteq PTIME.$$

Every class in the hierarchy implies a structural computational advantage against the hardness in *PTIME*. Hence, the second objective of this thesis generalizes the author's results in [21] and applies the new discrete parameter space characterization to exactly describe the complexity for image matching under projective transformations. The problem is complete in TC_O, which is a very natural complexity class because it exactly expresses the complexity of a variety of basic problems in computation, such as integer addition, multiplication, division and sorting.

Basically, the challenge in establishing the problem's membership in TC_O evolves from the open question of how to compute the data structure for dictionary $\mathcal{D}[\mathcal{F}_\mathrm{p}](A)$ by the power of TC_O. Chapter 3 shows that it is possible to select a lattice of a polynomial amount of points in the parameter space $\mathbb{R}^8$ to find at least one representative from every face in $\{\varphi_1, \dots, \varphi_r\}$. This enables another convenient way of obtaining a polynomial subset of projective transformations eligible to compute the complete dictionary $\mathcal{D}[\mathcal{F}_\mathrm{p}](A)$ even without realizing the complex data structure needed to represent $\{\varphi_1, \dots, \varphi_r\}$. In this fashion the computational complexity of image matching is decreased to fit the restrictions of TC_O.

From an algorithmic point of view, the containment of the projective image matching in $TC_O \subseteq NC_1$ means that the problem can be solved efficiently in a parallel fashion, i. e., in logarithmic time by circuits with bounded fan-in. In fact, if the fan-in is unbounded, it even works in constant time. Moreover, on single processor machines the containment in $TC_O \subseteq LOGSPACE$ causes very low memory consumption.

Completeness in TC_O states that image matching for projective transformations belongs to the most complex problems in that class. It is shown by reducing the TC_O-complete majority problem to image matching. More precisely, for every binary string s there are images A_s and B_s such that s contains a majority of positive bits if and only if there is a projective transformation f such that $f(A_s)$ resembles B_s with an exception of at most $0.5|s|$ pixels. Because image matching is complete in TC_O, computation formalisms of less expressive power are not able to model the problems. In fact, there is no polynomially sized, uniformly shaped family of Boolean formulas expressing projective image matching, since this captures exactly $AC_O \neq TC_O$.

Chapter 2

Preliminaries

2.1 Mathematical Basic Notations

The aim of this section is to fix notations of some fundamental mathematical concepts used in the thesis. Many details are omitted, but adequate references are given to the reader not familiar with the field.

Algebraic Concepts

The text books of Courant et al. [11, 12, 13] provides a substantial and detailed fund of knowledge about notions in this field. Subsequently, the important tools are briefly specified to clarify the applied notation.

$\mathbb{N}$, $\mathbb{Z}$ and $\mathbb{R}$ denote the common sets of *natural*, *integer* and *real numbers* with the standard operations $+$, $-$, $\cdot$, $|\ |$, max, min, $\sqrt{\ }$, log and so forth. For real numbers p let, as usual, $\lfloor p \rfloor$ be the largest integer less or equal to p, $[p] = \lfloor p+0.5 \rfloor$ and $\lceil p \rceil$ be the smallest integer greater or equal to p.

Any k-dimensional *vector spaces* over the set of real numbers is denoted by $\mathbb{R}^k$, for $k > 0$. For every *vector* $P = (p_1, \ldots, p_k)^T$ define the *length* $\|P\| = \sqrt{p_1^2 + \ldots + p_k^2}$. Following the concept of $[p]$ for real numbers let $[P] = ([p_1], \ldots, [p_k])^T$. If $p_1 = \ldots = p_k = 0$ then P is the *zero vector.*

A $(k \times k)$-*matrix* M is an array of k^2 real numbers. If

$$M = \begin{pmatrix} x_{1,1} & \cdots & x_{1,k} \\ \vdots & \ddots & \vdots \\ x_{k,1} & \cdots & x_{k,k} \end{pmatrix}$$

then the real number $\det(M) = \sum_{\phi \in \Phi(k)} \left(sgn(\phi) \prod_{i=1}^{k} x_{i,\phi(i)} \right)$ is the *determinant* of M where $\Phi(k)$ contains all permutations ϕ of the numbers $\{1, \ldots, k\}$ and $sgn(\phi)$ gives 1 if the number of mutual transpositions in ϕ is even and (-1) otherwise. The *inverse* M^{-1} of M is a $(k \times k)$-matrix with entries $\tilde{m}_{i,j}$ such that the value $\sum_{n=1}^{k} m_{i,n} \cdot \tilde{m}_{n,j}$ is one if $i = j$ and zero otherwise, for all $i, j \in \{1, \ldots, k\}$. Clearly, the inverse has the property $M^{-1} \cdot M \cdot P = P$ for all vectors $P \in \mathbb{R}^k$. The inverse M^{-1} can be found by the *adjunct* A, a $(k \times k)$-matrix where each entry $a_{i,j} = (-1)^{i+j} \cdot \det(M_{i,j})$, $i, j \in \{1, \ldots, k\}$ is determined by $M_{i,j}$, the *submatrix* of M obtained by deleting the ith row and the jth column. Every *regular matrix* M, i. e., when $\det(M) \neq 0$, has an inverse $M^{-1} = \det(M)^{-1} \cdot A^T$.

Polynomials $\ell : \mathbb{R}^k \to \mathbb{R}$ are exactly the functions built by a finite number of $+$ and $\cdot$ operations. A special case are *linear polynomials* ℓ. Every linear polynomial is given by a nonzero vector $(n_1, \ldots, n_k)^T \in \mathbb{R}^k$ and an additional number $n_{k+1} \in \mathbb{R}$ such that for all $P = (p_1, \ldots, p_k)^T \in \mathbb{R}^k$ it is true

$$\ell(p_1, \ldots, p_k) = n_1 p_1 + \ldots + n_k p_k + n_{k+1}.$$

Such a linear polynomial can be multiplied by a real number r and then $r\ell$ is the linear polynomial with vector $(rn_1, \ldots, rn_k)^T$ and number rn_{k+1}.

Computational Complexity Concepts

An introduction to computational complexity goes beyond the scope of this section, even for very basic notions. For a convenient overview, see the text books of Hopcroft et al. [20], Aho et al. [1] and Vollmer [45].

Binary strings $s \in \{0,1\}^*$, i. e., sequences over $\{0,1\}$ are used to encode numbers, graphs, images and so forth. Let $|s|$ denote the number of bits in s and if s is not the empty string, let $s(i)$ be the bit at ith position for all $i \in \{0, \ldots, |s| - 1\}$. Moreover, let 0^k and 1^k be the strings of k sequent zero bits, respectively positive bits, and if s and s' are strings then let $s|s'$ be their concatenation. If it is clear from the context that two strings are concatenated, the short form ss' is used.

Complexity measures depend on the used *computation model.* For sequential models *random access machines* are applied because they describe real computers most closely. Such a machine $\mathcal{M}$ encodes strings s as numbers stored in *registers.* $\mathcal{M}$ processes a sequence of *instructions* to initiate, copy and modify its registers and finally to represent the *output number.* The *time complexity* of $\mathcal{M}$ can be measured by the relation between the length of input/output string and the number of instruction performed by the machine. A machine runs in $O(|s|^k)$ time if the calculation takes at most $O(|s|^k)$ instructions, where k is a fixed natural number. If $k = 1$ then the machine runs in *linear time.*

The *space complexity* of $\mathcal{M}$ can be measured in terms of the number of bits of memory used during calculation. An interesting class of functions are those computable within $O(\log |s|)$ space because they form an important subset of the functions calculated in polynomial time.

For many functions, calculation requires the execution of basic arithmetic integer operations. Usually, the length of representations of numbers depend on

$|s|$, but often only to some very small extend, like e. g. $O(\log |s|)$. Then it is common practice to assume that the length of number representations has no impact on integer operations and that they can be done in $O(1)$ time. Merge sort, e. g., is known to sort a list of n elements in $O(n \log n)$ time, but this is true only if comparing integer values works in constant time. Complexity statements about sequential algorithms in Chapters 4 apply this kind of measure which is called *arithmetic complexity.* This means all basic integer operations are counted as one step and consequently, the time complexity of algorithms refers to the number of operations.

Another computation model used in the thesis are *circuits* which work especially as a measure for *parallel computation complexity.* Circuit complexity is explained in detail in Section 5.1.

Decision problems $\Pi \subseteq \{0,1\}^*$ are used to describe string properties, i. e., s has the property if and only if $s \in \Pi$. A lot of complexity theory builds on decision problems and hence, to describe the complexity of image matching, Chapter 5 formulates the problem as a decision problem.

Graph Concepts

In Chapter 4 some basic *graph* concepts are used. The text books of Brandstädt et al. [8] and Golumbic [19] present a considerable overview on graph theoretic and graph algorithmic notions. The new image matching algorithm operates just on *simple directed graphs* $G = (V, E)$ without *loops.* Hence, *nodes* V and *edges* E are finite sets and E consists of ordered pairs uv with $u, v \in V$ and $u \neq v$.

A graph $G = (V, E)$ is *connected* if for every pair u, v of nodes there is a (not necessarily directed) *path* $u = w_1, \ldots, w_p = v$ such that for all $i \in \{1, \ldots, p-1\}$ at least one edge $w_i w_{i+1}$ or $w_{i+1} w_i$ is in E . The graph G is called *acyclic* if there is no node set $w_1, \ldots, w_p$ with $w_p w_1 \in E$ and $w_i w_{i+1} \in E$ for all $i \in \{1, \ldots, p-1\}$.

Depth first search is a technique for random access machines to traverse a given graph in $O(|V|+|E|)$ time visiting every node and edge. The search travels along edges from one unvisited node to the next as long as finding unvisited neighbors. Then it applies *backtracking* to return from dead ends to nodes with left unvisited neighbors. On connected graphs the process stops after visiting all nodes and backtracking to the start node.

2.2 Concepts for Digital Images

A *digital image* (just image, for short) is a finite two-dimensional array of color values:

Definition 2.1. *The set of the first C natural numbers*

$$\mathcal{C} = \{0, 1, 2, 3, \ldots, C-1\}$$

are called ***colors****. A* ***digital image*** *A of size n is a mapping $A : \mathbb{Z}^2 \to \mathcal{C}$ with* ***support***

$$z(n) = \Big\{(i,j) \;\Big|\; -n \le i \le n, -n \le j \le n\Big\}$$

such that $A(i,j) = 0$ for all $(i,j) \notin z(n)$.

Nearest Neighbor Interpolation

Considering only the definition in terms of arrays does not reflect the purpose of digital images. In many real scenarios digital images A are used to represent two-dimensional real valued functions $\mathbb{R}^2 \to \mathbb{R}$ in a discrete fashion. But to interpret A as a continuous function requires the definition of color values in between indices $(i,j) \in \mathbb{Z}^2$, a mechanism called *interpolation*.

Definition 2.2. *If A is an image of size n then let $A_I : \mathbb{R}^2 \to \mathcal{C}$ be the* ***nearest neighbor interpolation*** *of A, i. e., for all $(x,y) \in \mathbb{R}^2$*

$$A_I(x,y) = A([x],[y]).$$

Nearest neighbor interpolation is commonly used in image processing. Although nowadays higher-order interpolation methods are available, like *linear* or *cubic interpolation*, they can be approximated in a nearest neighbor interpolation scheme working on a larger image. Consequently, it is reasonable to concentrate on nearest neighbor interpolation. The major advantage in a practical setting is the high efficiency of nearest neighbor interpolation.

The mapping $A_I : \mathbb{R}^2 \to \mathcal{C}$ has the continuous domain of $\mathbb{R}^2$ but it maps to the discrete color space $\mathcal{C}$. Consequently, there are coherent regions separated by discontinuous jumps where the color value changes abruptly. Let A be an image of size n. Then A_I defines color values for all points in the rectangular region

$$Z = \Big\{(x,y) \;\Big|\; -n - 0.5 \le x, y < n + 0.5\Big\} \subset \mathbb{R}^2$$

and for all $(x,y) \notin Z$ it is true $A_I(x,y) = 0$. Define a partition $Pix[n] = \{pix(i,j) \mid (i,j) \in z(n)\}$ of Z into squares

$$pix(i,j) = \Big\{(x,y) \;\Big|\; i - 0.5 \le x < i + 0.5 \text{ and } j - 0.5 \le y < j + 0.5\Big\},$$

such that for all $(i,j) \in z(n)$ and for all $(x,y) \in pix(i,j)$ it is true that $A_I(x,y) = A(i,j)$. The regions $pix(i,j)$ are called *pixels* (short for picture elements) of A.

Image Distortion

After filling digital images with color information at all points in $\mathbb{R}^2$ the next question is how to compare two images A and B. The simplest way would be to ignore the continuous interpretation of digital images:

Definition 2.3. *Let A be an image of size n and B be an images of size m. For a* ***color distortion measure*** $\omega : \mathcal{C} \times \mathcal{C} \to \mathbb{N}$ *let*

$$\delta(A, B) = \sum_{(i,j) \in z(\max\{n,m\})} \omega(A(i, j), B(i, j))$$

be a ***distortion*** *between A and B.*

Such a notion fulfills the conditions

1. non-negativity ($\delta(A, B) \geq 0$),
2. definiteness ($\delta(A, B) = 0$ if and only if $A = B$),
3. symmetry ($\delta(A, B) = \delta(B, A)$) and
4. the triangle inequality ($\delta(A, B) \leq \delta(A, X) + \delta(X, B)$)

of a *metric* if the measure $\omega : \mathcal{C} \times \mathcal{C} \to \mathbb{N}$ has these properties, too.

There are other reasonable ways to define the difference between images. For example, sticking to the continuous interpretation of A and B leads to the idea to measure the difference between two functions $\mathbb{R}^2 \to \mathcal{C}$. A common approach would be to consider the integral

$$\delta(A, B) = \iint_{R^2} \omega(A_I(x, y), B_I(x, y))\ dx\ dy$$

for some color distortion measure $\omega : \mathcal{C} \times \mathcal{C} \to \mathbb{N}$. Interestingly, for nearest neighbor interpolation this leads to the same notion:

Lemma 2.4 (Proof in Section 2.6)**.**

$$\iint_{R^2} \omega(A_I(x, y), B_I(x, y))\ dx\ dy = \sum_{(i,j) \in z(\max\{n,m\})} \omega(A(i, j), B(i, j))$$

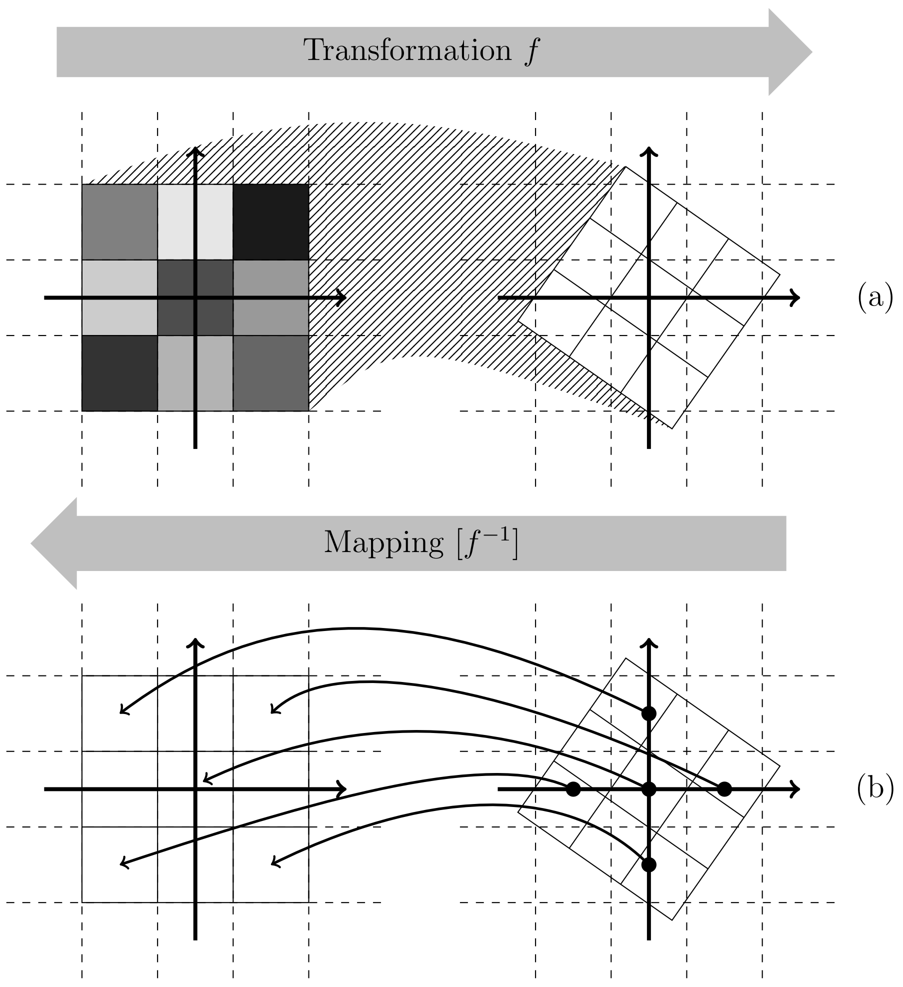
Image Transformations

Another concept for digital images introduced here are *transformations* $\mathbb{R}^2 \to \mathbb{R}^2$. According to definition transformations are continuous and cannot be straightforward applied for digital images. Consequently, *image transformations* apply the notion of image interpolation:

Definition 2.5. *A* ***transformation*** *is an invertible function $f : \mathbb{R}^2 \to \mathbb{R}^2$. For a digital image A of size n and an integer $m \in \mathbb{N}$ the* ***image transformation*** *$f(A)[m]$ of size m is a digital image such that*

$$f(A)[m](i, j) = A_I(f^{-1}(i, j))$$

for all $(i, j) \in z(m)$ and $f(A)[m](i, j) = 0$ if $(i, j) \notin z(m)$.



a) f defines points $f(x, y)$ for all $(x, y) \in \mathbb{R}^2$.

b) $f(A)[2]$ needs $[f^{-1}(i, j)]$ for all $(i, j) \in z(2)$. For visual convenience only five examples for $[f^{-1}(i, j)]$ are shown.

c) Finally, every color $f(A)(i, j)$ is obtained by $A([f^{-1}(i, j)])$.

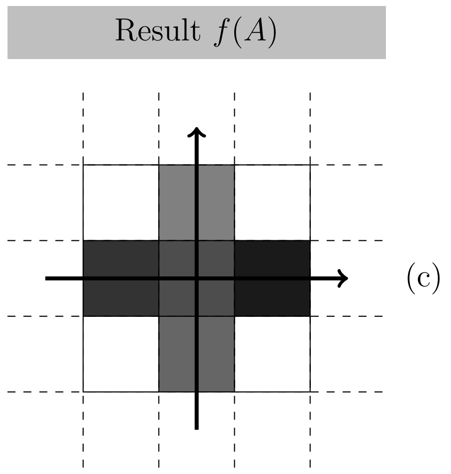


Figure 2.1: Digital image A scaled by 0.83 and rotated by 35°.

The idea for image $f(A)[m]$ is to approximate the continuous function $f(A)$. Notice that the existence of the *inverse* f^{-1} of a transformation f is crucial to compute the image transformations $f(A)[m]$. Figure 2.1 demonstrates the principles of image transformations. Particularly, it shows an image A that is transformed with scaling by 0.83 and rotating by 35 degrees.

Due to constraints given by the application or the computational complexity image matching is usually performed for a certain infinite set $\mathcal{F}$ of transformations which forms some specific class, given by a unique representation or a shared property.

Definition 2.6. *A set $\mathcal{F}$ of transformations is called a* ***transformation class*** *if for all $f \in \mathcal{F}$ it is true that $f^{-1} \in \mathcal{F}$. A transformation class $\mathcal{F}$ is called* ***parameterized transformation class*** *if there is a surjective (not necessarily total) mapping $\mathcal{P}[\mathcal{F}] : \mathbb{R}^k \to \mathcal{F}$ for some $k \in \mathbb{N}$.*

The advantage of parameterized transformation classes is that every element can be represented by a parameter vector. The next section presents a list of well known parameterized transformation classes.

Since transformations are continuous objects but images are discrete it follows that not every pair f_1 and f_2 of distinct transformations can form different image transformations $f_1(A)[m]$ and $f_2(A)[m]$. Relative to a transformation class $\mathcal{F}$ and an integer m there is always a finite set $\mathcal{D}[\mathcal{F}, m](A)$ of images of size m resulting from transforming A according to transformations in $\mathcal{F}$:

Definition 2.7. *Let $\mathcal{F}$ be a set of transformations and $m \in \mathbb{N}$ be an integer. For given image A, the set*

$$\mathcal{D}[\mathcal{F}, m](A) = \Big\{ f(A)[m] \;\Big|\; f \in \mathcal{F} \Big\}$$

is called the ***dictionary*** *of A.*

The dictionary $\mathcal{D}[\mathcal{F}, m](A)$ represents the *search space* of image matching algorithms. In fact, a lot of analysis in the following chapters is focused on the dictionary structure and the question of how to compute it.

An example of a dictionary can be found in Figure 2.2. It shows $\mathcal{D}[\mathcal{F}_{\mathtt{s}}, 2](A)$, $\mathcal{D}[\mathcal{F}_{\mathtt{r}}, 2](A)$ and $\mathcal{D}[\mathcal{F}_{\mathtt{sr}}, 2](A)$, dictionaries of images with size two, given by transforming the image A of Figure 2.1 with scalings $\mathcal{F}_{\mathtt{s}}$, rotations $\mathcal{F}_{\mathtt{r}}$ and combinations of scaling and rotation $\mathcal{F}_{\mathtt{sr}}$, transformation classes introduced in the next section.

2.3 Projective Transformation Classes

This section presents the covered transformation classes. Often, *projective transformations* represent a superset of the supported transformation class in many

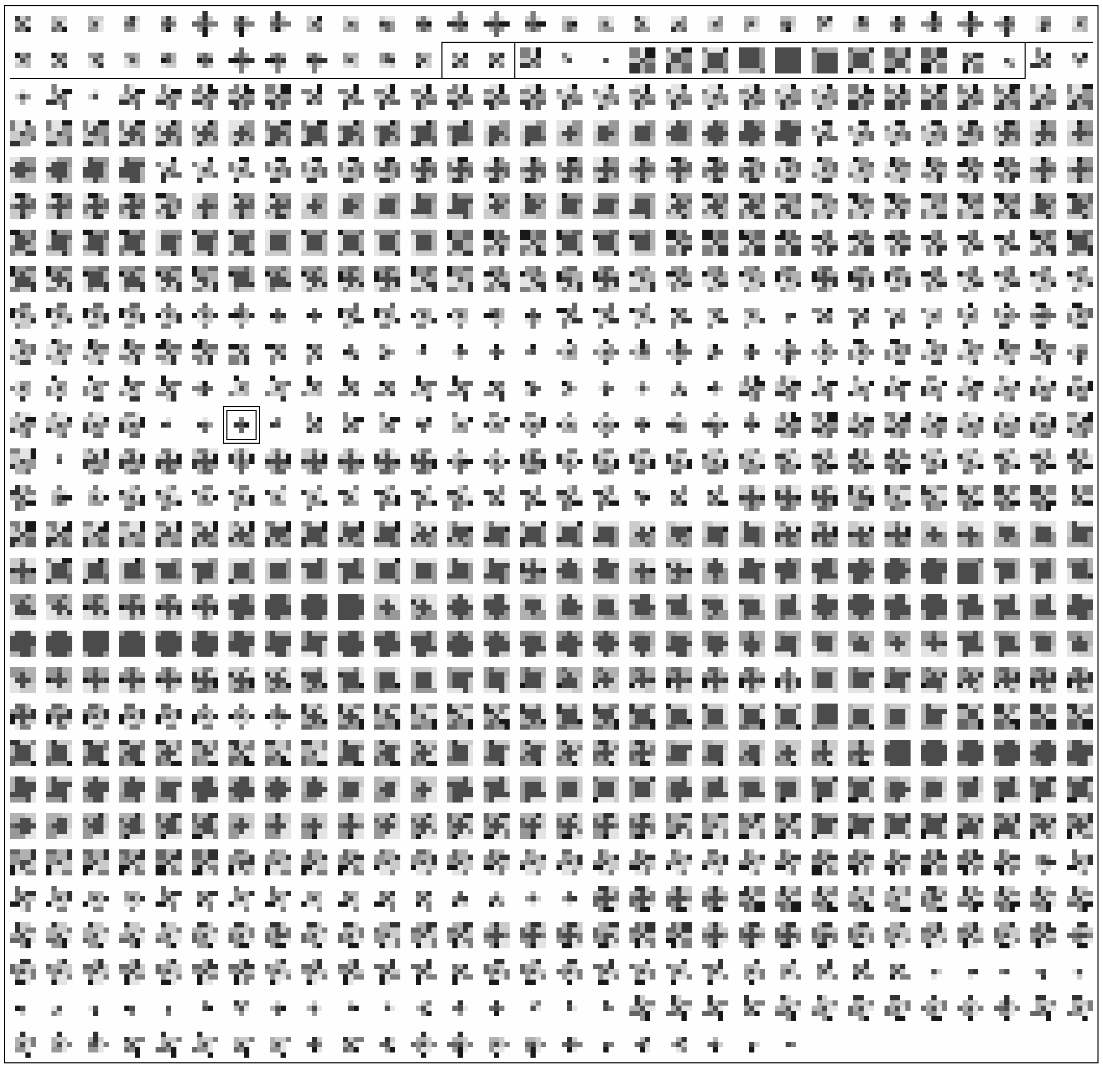


Figure 2.2: Dictionary $\mathcal{D}[\mathcal{F}_{\mathtt{sr}}, 2](A)$ of all 862 transformations of the image A given in Figure 2.1 by combinations of scaling and rotation. Using rotation only results in the first 44 images which form $\mathcal{D}[\mathcal{F}_{\mathtt{r}}, 2](A)$. The last two are found by scaling, too, and together with the next 14 images, they give $\mathcal{D}[\mathcal{F}_{\mathtt{s}}, 2](A)$. To get the remaining 804 requires true combinations of scaling and rotation and the double framed one is given by 0.83 scaling and 35° rotation like in Figure 2.1.

image processing applications. Consequently, it is reasonable to focus on image matching with projective transformations and its natural subclasses.

Definition 2.8. ***Projective transformations*** $\mathcal{F}_{\mathtt{p}}$ *are a parameterized set of functions* $f : \mathbb{R}^2 \to \mathbb{R}^2$ *obtained by a surjective mapping* $\mathcal{P}[\mathcal{F}_{\mathtt{p}}] : \mathbb{R}^8 \to \mathcal{F}_{\mathtt{p}}$*. For* *all* $P = (p_1, \dots, p_8)^T \in \mathbb{R}^8$ *with*

$$p_1p_5 - p_2p_4 \neq 0$$

$$\textit{and}$$

$$p_1p_5 - p_1p_6p_8 - p_2p_4 + p_2p_6p_7 + p_3p_4p_8 - p_3p_5p_7 \neq 0$$

let $f = \mathcal{P}[\mathcal{F}_\mathrm{p}]\,(P)$ *be the projective function defined as*

$$f(x,y) = \begin{cases} \textit{undefined} & : \textit{ if } xp_7 + yp_8 + 1 = 0 \\ \left(\frac{x'}{z'}, \frac{y'}{z'}\right) & : \textit{ otherwise, where } \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} p_1 & p_2 & p_3 \\ p_4 & p_5 & p_6 \\ p_7 & p_8 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}. \end{cases}$$

Notice that points P where $\mathcal{P}[\mathcal{F}_\mathrm{p}]$ is undefined would lead to functions f which are not invertible. In fact, the two inequalities guarantee that the (3×3)-matrix defining the transformation is invertible and that its inverse gives a proper projective transformation. However, the definition of projective transformations alone cannot state that $\mathcal{F}_\mathrm{p}$ is a transformation class at all. First it has to be shown that all elements of $\mathcal{F}_\mathrm{p}$ are invertible and that these inverse transformations are in $\mathcal{F}_\mathrm{p}$.

Lemma 2.9 (Proof in Section 2.6)**.** *The set* $\mathcal{F}_\mathrm{p}$ *is a parameterized transformation class.*

Projective transformations have the following capabilities: For eight given points X_1 to X_4 and Y_1 to Y_4 in $\mathbb{R}^2$ there is one projective transformation f with $f(X_i) = Y_i$ for $i \in \{1,2,3,4\}$. On the other hand, for ten points X_1 to X_5 and Y_1 to Y_5 a projective transformation f with $f(X_i) = Y_i$ for $i \in \{1,2,3,4,5\}$ does not necessarily exist. Moreover, projective transformations are *line preserving*:

Lemma 2.10 (Proof in Section 2.6)**.** *For any* $s_x, s_y, v_x, v_y \in \mathbb{R}$ *let*

$$L = \left\{ (x,y) \;\middle|\; \exists t \in \mathbb{R} \;\; \left(\begin{smallmatrix} x \\ y \end{smallmatrix}\right) = \left(\begin{smallmatrix} s_x \\ s_y \end{smallmatrix}\right) + t \cdot \left(\begin{smallmatrix} v_x \\ v_y \end{smallmatrix}\right) \right\}$$

be the straight line in $\mathbb{R}^2$ *through* $(s_x, s_y)^T$ *going along* $(v_x, v_y)^T$*. For every projective transformation* $f \in \mathcal{F}_\mathrm{p}$ *it is true that*

$$L' = \left\{ f(x,y) \;\middle|\; (x,y) \in L \right\}$$

is a straight line, too.

Consequently, one can think of a projective transformation f as the alignment of a quadrangle given by four points X_1 to X_4 to another quadrangle given by points Y_1 to Y_4. The rest of the points in $\mathbb{R}^2$ adapt to this alignment. Figure 2.3 demonstrates this.

The problem with the projective transformations is that, in general, they are not total functions. According to Definition 2.8 a transformation $f = \mathcal{P}[\mathcal{F}_\mathtt{p}]\,(p_1, \ldots, p_8)$ is undefined for points (x, y) with $p_7x + p_8y + 1 = 0$ because then the result vector would go to infinity. This behavior is unpleasant for the analysis of projective transformations and the following chapters invest some effort to handle it.

Because projective transformations are the most powerful transformation class considered, it is always assumed that notions are defined with respect to them, unless marked differently. Consequently, the explicit reference to projective transformations is mostly omitted. For example, the abbreviation $\mathcal{D}$ and $\mathcal{P}$ are applied instead of $\mathcal{D}[\mathcal{F}_\mathtt{p}]$ and $\mathcal{P}[\mathcal{F}_\mathtt{p}]$.

Subclasses of Projective Transformations

Beside projective transformations there are a number of natural subclasses supported by applications:

Definition 2.11. *The following list defines subsets of $\mathcal{F}_\mathtt{p}$, sets of functions $f : \mathbb{R}^2 \to \mathbb{R}^2$, by name, parameter and representation:*

Name	Parameter	Representation
Scaling $\mathcal{F}_\mathtt{s}$	$\mathcal{P}[\mathcal{F}_\mathtt{s}] : \mathbb{R} \to \mathcal{F}_\mathtt{s}$ $\forall s \in \mathbb{R} :\ \mathcal{P}[\mathcal{F}_\mathtt{s}]\,(s) =$ $\mathcal{P}[\mathcal{F}_\mathtt{p}]\,(s, 0, 0, 0, s, 0, 0, 0)$	$f(x,y) = \begin{pmatrix} s & 0 \\ 0 & s \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}$
Rotation $\mathcal{F}_\mathtt{r}$	$\mathcal{P}[\mathcal{F}_\mathtt{r}] : \mathbb{R} \to \mathcal{F}_\mathtt{r}$ $\forall \alpha \in \mathbb{R} :\ \mathcal{P}[\mathcal{F}_\mathtt{r}]\,(\alpha)$ $= \mathcal{P}[\mathcal{F}_\mathtt{p}]\,(c\alpha, s\alpha, 0, -s\alpha, c\alpha, 0, 0, 0)$ *with* $c\alpha = \cos\alpha$ *and* $s\alpha = \sin\alpha$	$f(x,y) =$ $\begin{pmatrix} c\alpha & s\alpha \\ -s\alpha & c\alpha \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}$
Scaling and Rotation $\mathcal{F}_\mathtt{sr}$	$\mathcal{P}[\mathcal{F}_\mathtt{sr}] : \mathbb{R}^2 \to \mathcal{F}_\mathtt{sr}$ $\forall p, q \in \mathbb{R} :\ \mathcal{P}[\mathcal{F}_\mathtt{sr}]\,(p, q) =$ $\mathcal{P}[\mathcal{F}_\mathtt{p}]\,(p, q, 0, -q, p, 0, 0, 0)$	$f(x,y) = \begin{pmatrix} p & q \\ -q & p \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}$
Translation $\mathcal{F}_\mathtt{t}$	$\mathcal{P}[\mathcal{F}_\mathtt{t}] : \mathbb{R}^2 \to \mathcal{F}_\mathtt{t}$ $\forall t_1, t_2 \in \mathbb{R} :\ \mathcal{P}[\mathcal{F}_\mathtt{t}]\,(t_1, t_2) =$ $\mathcal{P}[\mathcal{F}_\mathtt{p}]\,(1, 0, t_1, 0, 1, t_2, 0, 0)$	$f(x,y) =$ $\begin{pmatrix} 1 & 0 & t_1 \\ 0 & 1 & t_2 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$
Linear Transformation $\mathcal{F}_\ell$	$\mathcal{P}[\mathcal{F}_\ell] : \mathbb{R}^4 \to \mathcal{F}_\ell$ $\forall l_1, l_2, l_3, l_4 \in \mathbb{R} :$ $\mathcal{P}[\mathcal{F}_\ell]\,(l_1, l_2, l_3, l_4) =$ $\mathcal{P}[\mathcal{F}_\mathtt{p}]\,(l_1, l_2, 0, l_3, l_4, 0, 0, 0)$	$f(x,y) = \begin{pmatrix} l_1 & l_2 \\ l_3 & l_4 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}$

Affine Transformation $\mathcal{F}_{\mathtt{a}}$	$\mathcal{P}[\mathcal{F}_{\mathtt{a}}] : \mathbb{R}^6 \to \mathcal{F}_{\mathtt{a}}$ $\forall a_1, a_2, a_3, a_4, a_5, a_6 \in \mathbb{R}$: $\mathcal{P}[\mathcal{F}_{\mathtt{a}}]\,(a_1, a_2, a_3, a_4, a_5, a_6) =$ $\mathcal{P}[\mathcal{F}_{\mathtt{p}}]\,(a_1, a_2, a_3, a_4, a_5, a_6, 0, 0)$	$f(x,y) =$ $\begin{pmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$

According to definition, all sets $\mathcal{F}_{\mathtt{s}}$, $\mathcal{F}_{\mathtt{r}}$, $\mathcal{F}_{\mathtt{sr}}$, $\mathcal{F}_{\mathtt{t}}$, $\mathcal{F}_{\ell}$, and $\mathcal{F}_{\mathtt{a}}$, contain projective transformations. Notice that the parameters of scalings and rotations are $p, q \in \mathbb{R}$. A more natural pair of parameters would be given by a scaling factor $s \in \mathbb{R}$ and an angle $\alpha \in \mathbb{R}$. It is easy to see that these parameters are encoded in $p = s\cos\alpha$ and $q = s\sin\alpha$. This way establishes a better relationship between the parameters of $\mathcal{F}_{\mathtt{sr}}$ and $\mathcal{F}_{\mathtt{p}}$.

Notice that not all combinations of $l_1, l_2, l_3, l_4 \in \mathbb{R}$ give a linear transformation with matrix $\left(\begin{smallmatrix} l_1 & l_2 \\ l_3 & l_4 \end{smallmatrix}\right)$. Particularly, if $l_1 l_4 = l_2 l_3$ then the matrix is not regular and $\mathcal{P}[\mathcal{F}_{\mathtt{p}}]\,(l_1, l_2, 0, l_3, l_4, 0, 0, 0)$ is undefined. The same holds for affine transformations, scalings and rotations as well as solely scalings where the parameters fulfill $a_1 a_5 = a_2 a_4$ or $p = q = 0$ or respectively $s = 0$.

After defining these sets of functions it remains to answer whether all of them are classes of transformations:

Lemma 2.12 (Proof in Section 2.6). *The sets $\mathcal{F}_{\mathtt{s}}$, $\mathcal{F}_{\mathtt{r}}$, $\mathcal{F}_{\mathtt{sr}}$, $\mathcal{F}_{\mathtt{t}}$, $\mathcal{F}_{\ell}$, and $\mathcal{F}_{\mathtt{a}}$ are transformation classes.*

Because the classes $\mathcal{F}_{\mathtt{s}}$, $\mathcal{F}_{\mathtt{r}}$, $\mathcal{F}_{\mathtt{sr}}$, $\mathcal{F}_{\mathtt{t}}$, $\mathcal{F}_{\ell}$, and $\mathcal{F}_{\mathtt{a}}$ are subsets of $\mathcal{F}_{\mathtt{p}}$ they can be understood as restrictions to the transformation capabilities of $\mathcal{F}_{\mathtt{p}}$. For example, translation are the transformations which can align only one point P to another point Q. Figure 2.3 demonstrates the capabilities of each class.

2.4 Formalizing the Image Matching Problem

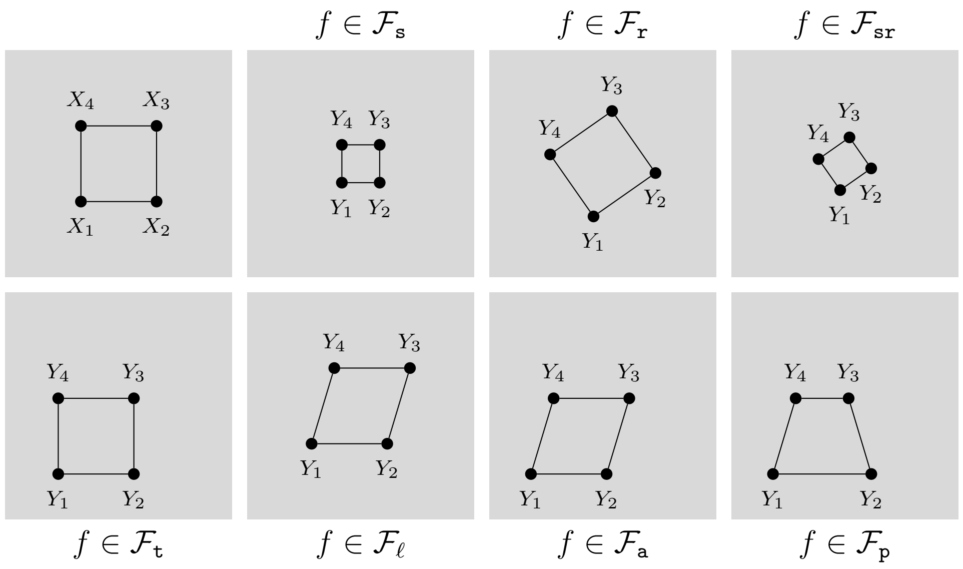
Although it has already been abstractly motivated in Chapter 1 the following formalizes the *image matching problem.*

Problem 2.13. Image Matching

Input: *Digital image A of size n and image B of size m.*

Constraints:
Image distortion measure $\omega : \mathcal{C} \times \mathcal{C} \to \mathbb{N}$.
Subclass $\mathcal{F}$ of projective transformations with $\mathcal{P}[\mathcal{F}] : \mathbb{R}^d \to \mathcal{F}$ and $d \leq 8$.

Output: *A Parameter $P \in \mathbb{R}^d$ such that the transformation $f^{-1} = \mathcal{P}[\mathcal{F}]\,(P)$ minimizes $\delta(f([m]A), B)$.*



$f \in \mathcal{F}_{\mathtt{s}}$
$f \in \mathcal{F}_{\mathtt{r}}$
$f \in \mathcal{F}_{\mathtt{sr}}$
X_4 X_3 X_1 X_2
Y_4 Y_3 Y_1 Y_2
$f \in \mathcal{F}_{\mathtt{t}}$
$f \in \mathcal{F}_{\ell}$
$f \in \mathcal{F}_{\mathtt{a}}$
$f \in \mathcal{F}_{\mathtt{p}}$

Figure 2.3: Four points $X_1, X_2, X_3, X_4 \in \mathbb{R}^2$ and the ways how a transformation $f \in \mathcal{F}_{\mathtt{s}}, \mathcal{F}_{\mathtt{r}}, \mathcal{F}_{\mathtt{sr}}, \mathcal{F}_{\mathtt{t}}, \mathcal{F}_{\ell}, \mathcal{F}_{\mathtt{a}}$, and $\mathcal{F}_{\mathtt{p}}$ maps to points $Y_i = f(X_i), 1 \leq i \leq 4$.

Roughly speaking, image matching is a optimization problem to find a transformation in $\mathcal{F}$ that modifies A such that it looks as similar to B as possible. For complexity analysis, however, it is sometimes favorable to analyze decision problems:

Problem 2.14. Image Matching (Decision Variant)

Input: *Digital image A of size n and image B of size m and a threshold $\mathcal{T} \in \mathbb{N}$.*

Constraints:
Image distortion measure $\omega : \mathcal{C} \times \mathcal{C} \to \mathbb{N}$.
Subclass $\mathcal{F}$ of projective transformations with $\mathcal{P}[\mathcal{F}] : \mathbb{R}^d \to \mathcal{F}$ and $d \leq 8$.

Question: *Is there a transformation $f \in \mathcal{F}$ such that $\delta(f(A)[m], B) \leq \mathcal{T}$.*

Chapter 5 describes the complexity of the above decision problem by pinpointing TC_O as the corresponding complexity class. Clearly, solving Problem 2.13 is at least as complex as deciding Problem 2.14. But actually it is the case that the computational costs needed to solve both problems are nearly the same. In particular, the main result states that, to find the best transformation f, it is sufficient to consider a polynomial amount of projective transformations $f_1, \ldots, f_r$. Moreover, all these transformations have a very efficient representation.

2.5 Algebraic and Computational Geometry

This section describes an important tool used in the thesis, namely a partition of the parameter space $\mathbb{R}^8$ of projective transformations into appropriate *subspaces*. The notion subspace does not refer to classical spaces but denotes connected subsets of $\mathbb{R}^8$, in most cases affine spaces. The complete definition of the partition is divided into parts, including the definition of simple subspaces in Definitions 2.15, *slices* in Definition 2.16, *faces* in Definition 2.18, and finally *arrangements* in Definition 2.20. To represent the space partition, *incidence graphs* are used – a well known data structure that allows efficient manipulation of arrangements. A formal description of incidence graphs is given in Definition 2.23.

To describe the partitions of $\mathbb{R}^8$ consider as a first step the following definition:

Definition 2.15. *For any polynomial $\ell : \mathbb{R}^8 \to \mathbb{R}$ the following subsets of $\mathbb{R}^8$ are called **simple subspaces**:*

$$\begin{aligned} h^0(\ell) &= \{(p_1, \dots, p_8)^T \in \mathbb{R}^8 \mid \ell(p_1, \dots, p_8) = 0\}, \\ h^+(\ell) &= \{(p_1, \dots, p_8)^T \in \mathbb{R}^8 \mid \ell(p_1, \dots, p_8) > 0\} \text{ and} \\ h^-(\ell) &= \{(p_1, \dots, p_8)^T \in \mathbb{R}^8 \mid \ell(p_1, \dots, p_8) < 0\}. \end{aligned}$$

*The subspace $h^0(\ell)$ is called **hypersurface** and the subspaces $h^+(\ell)$ and $h^-(\ell)$ are called **positive**, respectively **negative half space**. If ℓ is linear, then call $h^0(\ell)$ a **hyperplane**. Moreover, the sets*

$$\begin{aligned} H^+(\ell) &= h^0(\ell) \cup h^+(\ell) \text{ and} \\ H^-(\ell) &= h^-(\ell) \end{aligned}$$

are also simple subspaces and they are called half spaces.

An example of a half space is the 8-*ball* with center $(0, \dots, 0)^T$ and radius r. It is defined by the polynomial $\ell(p_1, \dots, p_8) = p_1^2 + \dots + p_8^2 - r^2$. The 8-ball $h^-(\ell)$ consists of all points with an Euclidean distance of less then r to its center point. The boundary of the 8-ball is given by the hypersurface $h^0(\ell)$, also called hypersphere, that describes all points with exact distance r to the ball's center point.

Subsequently, it is mostly the case that the polynomials ℓ, used to define subspaces $h^0(\ell)$, $h^+(\ell)$, $h^-(\ell)$ and so on, are linear. For such hyperplanes $h^0(\ell)$ it is very simple to get the distance to a given point. In fact, if the linear polynomial ℓ is determined by a nonzero vector $N \in \mathbb{R}^8$ and a number in $\mathbb{R}$, then the distance between $h^0(\ell)$ and a point $P \in \mathbb{R}^8$ is given by $\frac{|\ell(P)|}{\|N\|}$.

The following describes subspaces by the help of linear polynomials:

Definition 2.16. *Let $\mathcal{E} = \{\ell_1, \dots, \ell_w\}$ be a (possibly empty) set of linear polynomials $\mathbb{R}^8 \to \mathbb{R}$. The nonempty intersection of hyperplanes*

$$\chi = \bigcap_{\ell \in \mathcal{E}} h^0(\ell) \neq \emptyset$$

is called ***slice****. If $\mathcal{E} = \emptyset$ then it is assumed that $\bigcap_{\ell\in\mathcal{E}} h^0(\ell) = \mathbb{R}^8$. A slice χ is k-dimensional for some $k \in \{0, \dots, 8\}$ if the minimum number of linear polynomials sufficient to define χ is $8 - k$. In that case, χ is called k-slice for short.*

Slices are used to define linear subspaces isomorphic to $\mathbb{R}^k \subseteq \mathbb{R}^8$ with $k \in \{0, \dots 8\}$, i. e., affine spaces. For example the polynomials $\ell_1(p_1, \dots, p_8) = p_8 - 5$ and $\ell_1(p_1, \dots, p_8) = p_6 - p_7$ define the slice $\chi = h^0(\ell_1) \cap h^0(\ell_2)$ which spans an $\mathbb{R}^6$ within $\mathbb{R}^8$ where only the parameters p_1 to p_6 can be chosen arbitrarily and where $p_7 = p_6$ and $p_8 = 5$. All subsequent geometric structures are introduced relative to slices. In this way it is possible to generalize their definition to subspaces of $\mathbb{R}^8$. Trivially, $\mathbb{R}^8$ is a slice defined by the empty set of polynomials. For every statement on geometric structures that does not explicitly define the relative slice χ, it is assumed that $\chi = \mathbb{R}^8$.

Based on slices it is possible to introduce a common measure for the complexity of subspaces. Particularly, a subspace is *k-dimensional*, if it is contained in a k-slice but not in any $(k - 1)$-slice. The measure of dimension is also connected to linear (in)dependence. Particularly, a set $P_1, \dots, P_w$ of points is said to be *linearly dependent* if there exists a k-slice χ such that (1) $P_1, \dots, P_w \in \chi$ and (2) $1 \le k < w - 1$. Otherwise, $P_1, \dots, P_w$ are called *linearly independent.*

Linear independence is also easily generalized to sets of linear polynomials and hyperplanes:

Definition 2.17. *A set $\mathcal{E} = \{\ell_1, \dots, \ell_w\}$ of linear polynomials is called* ***linearly independent*** *relative to a k-slice χ with $k \in \{0, \dots, 8\}$, if the slice*

$$\chi' = \chi \cap \left(\bigcap_{\ell\in\mathcal{E}} h^0(\ell) \right) \neq \emptyset$$

is $(k - w)$-dimensional. Then, the set $\{h^0(\ell_1), \dots, h^0(\ell_w)\}$ of hyperplanes is also called linearly independent in χ.

Linearly independent polynomials do not describe parallel or collinear hyperplanes. That linear independence of polynomials and hyperplanes depends on the considered slice χ can be observed in Figure 2.4.

The basic building blocks for the partitions of $\mathbb{R}^8$ and moreover for partitions of slices in $\mathbb{R}^8$ are higher order subspaces given by the intersection of hyperplanes and half spaces:

Definition 2.18. *Let χ be a slice and $\mathcal{E} = \{\ell_1, \dots, \ell_w\}$ be a set of linear polynomials. For signs $s_1, \dots, s_w \in \{+, 0, -\}$ the (non-empty)* ***convex subspace***

$$\chi \cap \left(\bigcap_{i=1}^{w} h^{s_i}(\ell_i) \right) \neq \emptyset$$

is called an ***open face*** *in χ. Similarly, for signs $s'_1, \dots, s'_w \in \{+, -\}$ the (non-empty) intersection*

$$\chi \cap \left(\bigcap_{i=1}^{w} H^{s'_i}(\ell_i) \right) \neq \emptyset$$

is called an ***closed face*** *in χ. For short, a (open/closed) face φ is called* ***k-face*** *for some $k \in \{0, \dots, 8\}$, if it is k-dimensional.*

The notions open and closed are known in topology. The whole space of a k-slice χ is the most basic example of an open k-face because it is defined by the empty set of polynomials. Hyperplanes $h^0(\ell)$ are also open faces and they are $(k-1)$-dimensional and defined by a single polynomial ℓ. The half spaces $H^+(\ell)$ and $H^-(\ell)$ are two examples for closed k-faces given by a single polynomial ℓ. Another remarkable kind of faces are 0-faces, the ones that that consist of only one point. Figure 2.4 shows a number of example faces contained in a two-dimensional slice.

A more complex open face, that is important for the thesis, is the *hypercube* $\mathcal{Q}_r$ of radius r, i. e., a subspace that consists of all points $(p_1, \dots, p_8)^T$ with $|p_i| < r$ for all $i \in \{1, \dots, 8\}$. Since the absolute values of the coordinates have to be less than r, the hypercube is described by polynomials

$$\ell_i^+(p_1, \dots, p_8) = p_i + r \qquad \text{and} \qquad \ell_i^-(p_1, \dots, p_8) = p_i - r$$

for all $i \in \{1, \dots, 8\}$. That means,

$$\mathcal{Q}_r = \left(\bigcap_{i=1}^{8} h^+(\ell_i^+) \cap h^-(\ell_i^-) \right).$$

Open faces φ represent a generalization of open intervals on the real line. This means for all $\ell_i \in \mathcal{E}$ with $s_i \neq 0$ that $\varphi \subseteq h^{s_i}(\ell_i)$ but all points P at the infinitely thin border $h^0(\ell_i)$ do not belong to face. Closed faces, on the other hand, are not necessarily closed on all sides. Depending on the specific polynomials in $\mathcal{E}$, individual borders may or may not belong to the face. The following definition of the boundary simplifies statements about bordering points of faces:

Definition 2.19. *Let χ be a slice and $\mathcal{E} = \{\ell_1, \dots, \ell_w\}$ be a set of linear polynomials. If φ is a (open/closed) face given by $\mathcal{E}$ and signs $s_1, \dots, s_w$, then the* ***interior*** *of φ, denoted as $\downarrow\!\varphi$, is the open face*

$$\downarrow\!\varphi = \chi \cap \left(\bigcap_{i=1}^{w} h^{s_i}(\ell_i) \right)$$

and the ***boundary*** *of φ, denoted as $\hat{\varphi}$, is defined by*

$$\hat{\varphi} = \chi \cap \left(\bigcap_{i=1}^{w} H_i \right) \setminus \downarrow\!\varphi$$

where for all $i \in \{1, \ldots, w\}$

$$H_i = \begin{cases} h^0(\ell_i) & : \text{ if } s_i = 0 \\ H^+(\ell_i) & : \text{ if } s_i = + \\ H^+(-\ell_i) & : \text{ if } s_i = -. \end{cases}$$

The interior of a given face φ is the largest open face contained in φ. Clearly, if φ is already an open face, then $\downarrow\varphi = \varphi$. On the other hand, the boundary $\hat{\varphi}$ of a face φ consists of the infinitely thin film of points around φ. For open faces and many closed faces it not true that $\hat{\varphi} \subseteq \varphi$. The announced partitions of a slice χ, particularly of the slice $\mathbb{R}^8$, that are used in the new algebraic characterization of image transformations, are the following sets $\mathcal{H}$ of closed faces which are called *closed arrangements* in the thesis. However, to use this structure efficiently in algorithms the following defines also a finer partition of χ called *open arrangement.*

Definition 2.20. *Let χ be a slice and $\mathcal{E} = \{\ell_1, \ldots, \ell_w\}$ be a (possibly empty) set of linear polynomials. Then, the **closed arrangement** in χ defined by $\mathcal{E}$ is the set*

$$\mathcal{H}_\chi[\mathcal{E}] = \left\{ \varphi = \chi \cap \left(\bigcap_{k=1}^{w} H^{s_k}(\ell_k) \right) \;\middle|\; \exists s_1, \ldots, \exists s_w \in \{+, -\}, \varphi \neq \emptyset \right\}$$

*and the **open arrangement** in χ is the set*

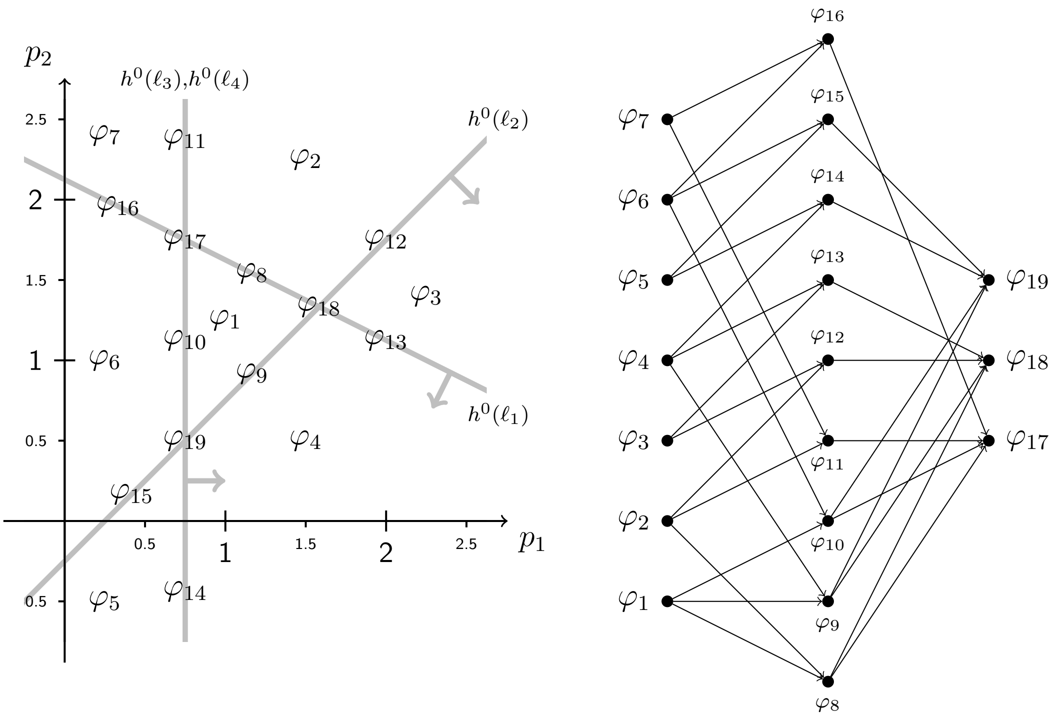
$$\mathcal{A}_\chi[\mathcal{E}] = \left\{ \varphi = \chi \cap \left(\bigcap_{k=1}^{w} h^{s_k}(\ell_k) \right) \;\middle|\; \exists s_1, \ldots, \exists s_w \in \{+, 0, -\}, \varphi \neq \emptyset \right\}$$

For short, let $\mathcal{H}[\mathcal{E}] = \mathcal{H}_{\mathbb{R}^8}[\mathcal{E}]$ and $\mathcal{A}[\mathcal{E}] = \mathcal{A}_{\mathbb{R}^8}[\mathcal{E}]$.

The arrangements $\mathcal{H}_\chi[\mathcal{E}]$ and $\mathcal{A}_\chi[\mathcal{E}]$ consist of all closed, respectively open, faces that can be defined by the polynomials in $\mathcal{E}$. Clearly, if χ is a k-slice then $\mathcal{A}_\chi[\mathcal{E}]$ contains k'-faces for all k' in $\{0, \ldots, k\}$. For the dimensions $k' = 0$, $k' = 1$ or $k' = 2$ it is usual that k'-faces have special names that describe there appearance. Particularly, 0-faces are called *vertices* or *points*, 1-faces are called *(straight) line segments* and 2-faces are *plane segments.* Figure 2.4 illustrates the two arrangement within a two-dimensional slice.

It is evident that both sets define partitions of the slice χ:

Lemma 2.21 (Proof in Section 2.6)**.** *For every slice χ and every set $\mathcal{E}$ of linear polynomials it is true that $\mathcal{A}_\chi[\mathcal{E}]$ and $\mathcal{H}_\chi[\mathcal{E}]$ are partitions of χ.*


p_2
$h^0(\ell_3), h^0(\ell_4)$
$h^0(\ell_2)$
$h^0(\ell_1)$
p_1
φ_1 φ_2 φ_3 φ_4 φ_5 φ_6 φ_7 φ_8 φ_9 φ_{10} φ_{11} φ_{12} φ_{13} φ_{14} φ_{15} φ_{16} φ_{17} φ_{18} φ_{19}

(a) (b)

Figure 2.4: (a) A two-dimensional slice χ with $p_3 = \ldots = p_8 = 0$ containing an open arrangement $\mathcal{A}_\chi[\mathcal{E}]$ defined by a set of linear polynomials $\mathcal{E} = \{\ell_1, \ell_2, \ell_3, \ell_4\}$ and (b) the incidence graph $\mathcal{G}_\chi[\mathcal{E}]$ of the arrangement. The polynomials

$$\begin{aligned}
\ell_1(p_1, \ldots, p_8) &= -4p_1 - 8p_2 + 17,\\
\ell_2(p_1, \ldots, p_8) &= 4p_1 - 4p_2 - 1,\\
\ell_3(p_1, \ldots, p_8) &= 4p_1 + p_3 - 3 \text{ and}\\
\ell_4(p_1, \ldots, p_8) &= 8p_1 - p_3 - 6
\end{aligned}$$

define only three hyperplanes in χ because, although linearly independent in $\mathbb{R}^8$, the polynomials ℓ_3 and ℓ_4 are linearly dependent in χ. The gray arrows indicate the positive half spaces with respect to all hyperplanes. The incidence graph $\mathcal{G}_\chi[\mathcal{E}]$ is three-partite, with one partition for plane segments $\varphi_1, \ldots, \varphi_7$, one for line segments $\varphi_8, \ldots \varphi_{16}$, and one for vertices $\varphi_{17}, \varphi_{18}, \varphi_{19}$. The closed arrangement $\mathcal{H}_\chi[\mathcal{E}] = \{\varphi_1', \varphi_2', \varphi_3', \varphi_4', \varphi_5', \varphi_6', \varphi_7'\}$ defined by $\mathcal{E}$ consists of the seven closed faces

$$\begin{aligned}
\varphi_1' &= \varphi_1 \cup \varphi_8 \cup \varphi_{10} \cup \varphi_{17}, & \varphi_2' &= \varphi_2 \cup \varphi_{11}, & \varphi_3' &= \varphi_3 \cup \varphi_{12},\\
\varphi_4' &= \varphi_4 \cup \varphi_9 \cup \varphi_{13} \cup \varphi_{14} \cup \varphi_{18} \cup \varphi_{19}, & & & \varphi_5' &= \varphi_5 \cup \varphi_{15},\\
\varphi_6' &= \varphi_6 \cup \varphi_{16}, & \varphi_7' &= \varphi_7.
\end{aligned}$$

Incidence Graphs

An important property of arrangements is that they can be nicely represented in data structures. Moreover, such representation can be computed very efficiently. Regard the following property of open arrangements:

Definition 2.22. *For a slice χ and a set $\mathcal{E} = \{\ell_1, \dots, \ell_w\}$ of linear polynomials let*

$$\varphi = \chi \cap \left(\bigcap_{i=1}^{w} h^{s_i}(\ell_i) \right) \qquad \text{and} \qquad \varphi' = \chi \cap \left(\bigcap_{i=1}^{w} h^{s'_i}(\ell_i) \right)$$

be two faces in $\mathcal{A}_\chi[\mathcal{E}]$ given by signs $s_1, \dots, s_w, s'_1, \dots, s'_w \in \{+, 0, -\}$. Then φ and φ' are called ***incident*** *if there is an $i \in \{1, \dots, w\}$ with $s_i \in \{+, -\}$ and $s'_i = 0$ and for all $k \in \{1, \dots, w\}$ with $k \neq i$ the signs s_k and s'_k fulfill*

1. *$s_k = s'_k$ if ℓ_i and ℓ_k are linearly independent relative to χ and otherwise*
2. *$\chi \cap h^{s_k}(\ell_k) = \chi \cap h^{s_i}(\ell_i)$ and $s'_k = 0$.*

Moreover, φ is a ***superface*** *of φ' and φ' is a* ***subface*** *of φ.*

Incidence describes the property of φ' being a $(k-1)$-face in the boundary of the k-face φ. Moreover, since every boundary separates two faces of equal dimension, the incidence is also a way of describing geometrical adjacency. For example, in Figure 2.4 the faces

$$\varphi_1 = \chi \cap \left(\bigcap_{i=1}^{4} h^{s_i}(\ell_i) \right) \quad \text{and} \quad \varphi_{10} = \chi \cap \left(\bigcap_{i=1}^{4} h^{s'_i}(\ell_i) \right)$$

given by the signs $s_1 = +$, $s_2 = -$, $s_3 = s_4 = +$ and $s'_1 = +$, $s'_2 = -$, $s_3 = s_4 = 0$ are incident because $s_3 = +$ and $s'_3 = 0$ and (1) for the polynomials ℓ_1 and ℓ_2, which, relative to χ, are linearly independent of ℓ_3, it is true $s_1 = s'_1$ and $s_2 = s'_2$ and (2) for the polynomial ℓ_4, which, relative to χ, is linearly dependent of ℓ_3, it is true that $\chi \cap h^{s_4}(\ell_4) = \chi \cap h^{s_3}(\ell_3)$ and $s'_4 = 0$.

The incidence of faces can be encoded into a so-called incidence graph:

Definition 2.23. *For a k-slice χ with $k \in \{0, \dots, 8\}$ and a set $\mathcal{E}$ of linear polynomials, a directed graph $\mathcal{G}_\chi[\mathcal{E}] = (V, E)$ to encode the incidence between faces of the open arrangement $\mathcal{A}_\chi[\mathcal{E}]$ is called* ***incidence graph*** *if*

1. *there is a bijection $\varphi : V \to \mathcal{A}_\chi[\mathcal{E}]$ between the nodes V and the faces in $\mathcal{A}_\chi[\mathcal{E}]$,*
2. *for all $u, v \in V$ there is a directed edge uv if and only if $\varphi(u)$ is a superface of $\varphi(v)$,*

3. *for all $v \in V$ the neighborhood is partitioned into superface neighbors $E_{in}(v) = \{u \mid uv \in E\}$ and subface neighbors $E_{out}(v) = \{u \mid vu \in E\}$,*

4. *every node $v \in V$ is labeled by $P(v)$, a point contained in $\varphi(v)$ and*

5. *every node $v \in V$ is labeled by $\ell(v)$, the set of polynomials $\ell \in \mathcal{E}$ describing hyperplanes containing $\varphi(v)$, i. e., with $\varphi(v) \subseteq \chi \cap h^0(\ell)$.*

Incidence graphs have a number of useful properties. For example, they are connected. If χ is a k-slice then $\mathcal{G}_\chi[\mathcal{E}] = (V, E)$ is a $(k+1)$-partite graph. This means that V is partitioned into the sets V_0, containing all nodes of 0-faces, V_1, the nodes of 1-faces and so forth to V_k, the set of k-face nodes, such that if $u, v \in V_i, i \in \{0, \ldots, k\}$ then $uv \notin E$. Additionally, every edge uv fulfills that $u \in V_{i-1}$ and $v \in V_i$ for some $i \in \{1, \ldots, k\}$. An example incidence graph can be seen in Figure 2.4.

Edelsbrunner et al. [15, 16] shows that the computation of $\mathcal{G}_\chi[\mathcal{E}]$ on basis of $\mathcal{E}$ takes $O(|\mathcal{E}|^k)$ time, if χ is a k-slice with $k > 1$. Actually, they show that the incidence graph $\mathcal{G}[\mathcal{E}]$ of an arrangement $\mathcal{A}[\mathcal{E}]$ in $\mathbb{R}^k$ defined by n linear polynomials of the form $\mathbb{R}^k \to \mathbb{R}$ can be computed in $O(n^k)$ time. According to definition it is true that the linear polynomials in $\mathcal{E}$ are of the form $\mathbb{R}^8 \to \mathbb{R}$ independent of the dimension k of subspace χ in which they define the arrangement. But it is easy to project the polynomials to the space χ and obtain in $O(|\mathcal{E}|)$ time linear polynomials of the form $\mathbb{R}^k \to \mathbb{R}$. For example, if χ is the 2-slice with $p_7 = p_6$ and $p_8 = 5$ then any polynomial

$$\ell(p_1, \ldots, p_8) = c_1 p_1 + \ldots + c_8 p_8 + c_9$$

can be projected to

$$\ell'(p_1, \ldots, p_6) = c_1 p_1 + \ldots + (c_6 + c_7) p_6 + (5c_8 + c_9).$$

In this way $\mathcal{E}$ can be efficiently transformed into a suitable input of Edelsbrunner's algorithm and the incidence graph of the arrangement $\mathcal{A}_\chi[\mathcal{E}]$ can be computed just in the announced time bound.

Moreover, the original algorithm of Edelsbrunner et al. [15, 16] computes $\ell(v)$ only for a subset of nodes $v \in V$. In fact, if χ is a k-slice then $\ell(v)$ is available only for nodes v representing $(k-1)$-faces. Clearly, for k-face nodes v it is true $\ell(v) = \emptyset$. The following lemma states that it is possible to label all remaining nodes in the same fashion with asymptotically no additional effort.

Lemma 2.24 (Proof in Section 2.6)**.** *Let χ be a k-slice in $\mathbb{R}^8$ with $1 < k \leq 8$ and $\mathcal{E}$ be a set of polynomials $\mathbb{R}^8 \to \mathbb{R}$. It is possible to compute $\ell(v)$ for all nodes $v \in V$ of the incidence graph $\mathcal{G}_\chi[\mathcal{E}] = (V, E)$ in $O(|\mathcal{E}|^k)$ time.*

Accordingly, it is convenient to assume that $\ell(v)$ is computed for all nodes $v \in V$ during the construction of the incidence graph $\mathcal{G}_{\mathcal{X}}[\mathcal{E}] = (V, E)$ by the algorithm of Edelsbrunner et al. [15, 16].

Once established, the incidence graph can be used to solve a number of computational problems in arrangements. In the following image matching algorithms incidence graphs are traversed by depth first search. Since incidence graphs are directed it is sometimes necessary to distinguish between *directed depth first search* and *undirected depth first search*. In the first case it is only possible to traverse along the direction of an edge and in the second case the directions are simply ignored which makes bidirectional traversing possible.

2.6 Technical Proofs

The Proof of Lemma 2.4

Proof.

$$\iint\limits_{\mathbb{R}^2} \omega(A_I(x,y), B_I(x,y))\ dx\ dy = \sum_{(i,j)\in\mathbb{Z}^2} \int\limits_{j-0.5}^{j+0.5} \int\limits_{i-0.5}^{i+0.5} \omega(A([x],[y]), B([x],[y]))\ dx\ dy$$

$$= \sum_{(i,j)\in\mathbb{Z}^2} \omega(A(i,j), B(i,j)) \cdot \int\limits_{j-0.5}^{j+0.5} \int\limits_{i-0.5}^{i+0.5} 1\ dx\ dy = \sum_{(i,j)\in z(\max\{n,m\})} \delta(A(i,j), B(i,j)).$$

□

The Proof of Lemma 2.9

Proof. Every $f \in \mathcal{F}_{\mathrm{p}}$ is defined by a (3×3)-matrix $M = \left(\begin{smallmatrix} p_1 & p_2 & p_3 \\ p_4 & p_5 & p_6 \\ p_7 & p_8 & 1 \end{smallmatrix}\right)$ given by some parameter point $P = (p_1, \ldots, p_8)^T \in \mathbb{R}^8$. By definition it is true that $p_1p_5 + p_2p_6p_7 + p_3p_4p_8 - p_1p_6p_8 - p_2p_4 - p_3p_5p_7 \neq 0$ and thus, the inverse

$$M^{-1} = \begin{pmatrix} q_1 & q_2 & q_3 \\ q_4 & q_5 & q_6 \\ q_7 & q_8 & q_9 \end{pmatrix}$$

of matrix M exists. Moreover, from definition follows that $q_9 = \frac{p_1p_5 - p_2p_4}{\det(M)} \neq 0$.

Now consider the projective function $f' = \mathcal{P}[\mathcal{F}_{\mathrm{p}}](Q)$ given by the parameter point $Q = q_9^{-1} \cdot (q_1, \ldots, q_8)^T$. Firstly, as $\det(M^{-1}) = \det(M)^{-1} \neq 0$ it is true that

$$\frac{q_1}{q_9}\frac{q_5}{q_9} + \frac{q_2}{q_9}\frac{q_6}{q_9}\frac{q_7}{q_9} + \frac{q_3}{q_9}\frac{q_4}{q_9}\frac{q_8}{q_9} \neq \frac{q_1}{q_9}\frac{q_6}{q_9}\frac{q_8}{q_9} + \frac{q_2}{q_9}\frac{q_4}{q_9} + \frac{q_3}{q_9}\frac{q_5}{q_9}\frac{q_7}{q_9}$$

and

$$\frac{q_1}{q_9}\frac{q_5}{q_9} \neq \frac{q_2}{q_9}\frac{q_4}{q_9}.$$

Hence, $f'(x', y') = (x''/z'', y''/z'')$ is a projective function with $(x'', y'', z'') = q_9^{-1} \cdot M^{-1} \cdot (x', y', 1)$. Moreover, f' is the inverse of f by the following argumentation: If $(x', y') = f(x, y)$ then there is $z' \in \mathbb{R}$ such that $(z'x', z'y', z')^T = M \cdot (x, y, 1)^T$. Now let $(x'', y'') = f'(x', y')$. Then there is $z'' \in \mathbb{R}$ such that $(z''x'', z''y'', z'')^T = q_9^{-1} \cdot M^{-1} \cdot (x', y', 1)^T$ and

$$\begin{aligned}
\begin{pmatrix} x'' \\ y'' \\ 1 \end{pmatrix} &= \frac{1}{z''} \cdot \begin{pmatrix} z''x'' \\ z''y'' \\ z'' \end{pmatrix} = \frac{1}{z''q_9} \cdot M^{-1} \cdot \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \frac{1}{z''q_9z'} \cdot M^{-1} \cdot \begin{pmatrix} z'x' \\ z'y' \\ z' \end{pmatrix} \\
&= \frac{1}{z''q_9z'} \cdot M^{-1} \cdot M \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \frac{1}{z''q_9z'} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}.
\end{aligned}$$

Hence, $f'(f(x, y)) = (x, y)$. □

The Proof of Lemma 2.10

Proof. Assume that

$$M = \begin{pmatrix} p_1 & p_2 & p_3 \\ p_4 & p_5 & p_6 \\ p_7 & p_8 & 1 \end{pmatrix}$$

is the projection matrix of f. Then for every $t \in \mathbb{R}$ with $t \neq -\frac{p_7s_x+p_8s_y+1}{p_7v_x+p_8v_y}$ it is true

$$f\left(\left(\begin{smallmatrix} s_x \\ s_y \end{smallmatrix}\right) + t \cdot \left(\begin{smallmatrix} v_x \\ v_y \end{smallmatrix}\right)\right) = \begin{pmatrix} s'_x \\ s'_y \end{pmatrix} + t' \cdot \begin{pmatrix} v'_x \\ v'_y \end{pmatrix}$$

with

$$\begin{aligned}
s'_x &= \frac{p_1v_x + p_2v_y}{p_7v_x + p_8v_y}, \\
s'_y &= \frac{p_4v_x + p_5v_y}{p_7v_x + p_8v_y}, \\
v'_x &= \frac{(p_1p_8 - p_2p_7)(s_xv_y - s_yv_x) + (p_3p_7 - p_1)v_x + (p_3p_8 - p_2)v_y}{p_7v_x + p_8v_y}, \\
v'_y &= \frac{(p_4p_8 - p_5p_7)(s_xv_y - s_yv_x) + (p_6p_7 - p_4)v_x + (p_6p_8 - p_5)v_y}{p_7v_x + p_8v_y} \quad \text{and} \\
t' &= \frac{1}{t(p_7v_x + p_8v_y) + (p_7s_x + p_8s_y + 1)}.
\end{aligned}$$

Hence, every (but one) point (x, y) of L is transformed onto the line

$$L' = \left\{ f(x, y) \;\middle|\; \exists t' \; f\left(\begin{smallmatrix} x \\ y \end{smallmatrix}\right) = \begin{pmatrix} s'_x \\ s'_y \end{pmatrix} + t' \cdot \begin{pmatrix} v'_x \\ v'_y \end{pmatrix} \right\}$$

But also, every point (x', y') on L' is the mapping destination of a point (x, y) on L, i. e., $f(x, y) = (x', y')$. The point (x', y') is defined by a parameter t'. But then

$$t = \frac{t'^{-1} - (p_7 s_x + p_8 s_y + 1)}{p_7 v_x + p_8 v_y}$$

defines the point (x, y) on L. $\square$

The Proof of Lemma 2.12

Proof. Since the elements of the sets are projective transformations their invertibility follows. It remains to show that the inverses remain in the particular set.

In each case except $\mathcal{F}_{\mathtt{t}}$ and $\mathcal{F}_{\mathtt{a}}$ the transformation f has the form $f(x, y) = M \cdot (x, y)$ with M being a two-by-two-matrix. The inverse $f^{-1}(x, y) = M^{-1} \cdot (x, y)^T$ is obtained by M^{-1}, the inverse matrix of M. The following shows for each case that f^{-1} belongs to the specific class.

Scaling: The inverse is

$$f^{-1}(x, y) = \begin{pmatrix} s^{-1} & 0 \\ 0 & s^{-1} \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix},$$

the scaling $f^{-1} = \mathcal{P}[\mathcal{F}_{\mathtt{s}}]\,(s^{-1})$.

Rotation: The inverse is

$$f^{-1}(x, y) = \begin{pmatrix} \cos -\alpha & \sin -\alpha \\ -\sin -\alpha & \cos -\alpha \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix},$$

the rotation $f^{-1} = \mathcal{P}[\mathcal{F}_{\mathtt{r}}]\,(-\alpha)$.

Combinations of scaling and rotation: The inverse is

$$f^{-1}(x, y) = \begin{pmatrix} \frac{p}{p^2+q^2} & \frac{-q}{p^2+q^2} \\ \frac{q}{p^2+q^2} & \frac{p}{p^2+q^2} \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix},$$

the combination of scaling and rotation $f^{-1} = \mathcal{P}[\mathcal{F}_{\mathtt{sr}}]\left(\frac{p}{p^2+q^2}, \frac{-q}{p^2+q^2}\right)$.

Linear transformation: The inverse is

$$f^{-1}(x, y) = \begin{pmatrix} \frac{l_4}{l_1 l_4 - l_2 l_3} & \frac{-l_2}{l_1 l_4 - l_2 l_3} \\ \frac{-l_3}{l_1 l_4 - l_2 l_3} & \frac{l_1}{l_1 l_4 - l_2 l_3} \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix},$$

the linear transformation $f^{-1} = \mathcal{P}[\mathcal{F}_{\ell}]\left(\frac{l_4}{l_1 l_4 - l_2 l_3}, \frac{-l_2}{l_1 l_4 - l_2 l_3}, \frac{-l_3}{l_1 l_4 - l_2 l_3}, \frac{l_1}{l_1 l_4 - l_2 l_3}\right)$.

In the remaining cases, $\mathcal{F}_{\mathtt{t}}$ and $\mathcal{F}_{\mathtt{a}}$, a transformation f has the form $f(x, y) = M \cdot (x, y)^T + t$, where M is a (2×2)-matrix and t is a vector in $\mathbb{R}^2$. If M^{-1} is the inverse of M then $f^{-1}(x, y) = M^{-1} \cdot (x, y)^T - M^{-1} \cdot t$ is the inverse of f.

Translation: The inverse is

$$f^{-1}(x,y) = \begin{pmatrix} 1 & 0 & -t_1 \\ 0 & 1 & -t_2 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix},$$

the translation $f^{-1} = \mathcal{P}[\mathcal{F}_{\mathtt{t}}]\,(-t_1, -t_2)$.

Affine transformation: The inverse is

$$f^{-1}(x,y) = \begin{pmatrix} \frac{a_5}{a_1a_5-a_2a_4} & \frac{-a_2}{a_1a_5-a_2a_4} & \frac{a_6a_2-a_3a_5}{a_1a_5-a_2a_4} \\ \frac{-a_4}{a_1a_5-a_2a_4} & \frac{a_1}{a_1a_5-a_2a_4} & \frac{a_3a_4-a_6a_1}{a_1a_5-a_2a_4} \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix},$$

the affine transformation

$$f^{-1} = \mathcal{P}[\mathcal{F}_{\mathtt{a}}]\left(\frac{a_5}{a_1a_5-a_2a_4}, \frac{-a_2}{a_1a_5-a_2a_4}, \frac{a_6a_2-a_3a_5}{a_1a_5-a_2a_4}, \frac{-a_4}{a_1a_5-a_2a_4}, \frac{a_1}{a_1a_5-a_2a_4}, \frac{a_3a_4-a_6a_1}{a_1a_5-a_2a_4}\right).$$

□

The Proof of Lemma 2.21

Proof. The proof works by induction over w, the number of polynomials in $\mathcal{E} = \{\ell_1, \ldots, \ell_w\}$. If $w = 0$ then $\mathcal{A}_\chi[\mathcal{E}] = \mathcal{H}_\chi[\mathcal{E}] = \{\chi\}$.

Otherwise, if $w \geq 1$ then the statement is true for $\mathcal{E}' = \mathcal{E} \setminus \{\ell_w\}$, i. e., every point P of χ belongs to exactly one face $\varphi' \in \mathcal{A}_\chi[\mathcal{E}']$. Then every face φ in $\mathcal{A}_\chi[\mathcal{E}]$ is an intersection of a face φ' in $\mathcal{A}_\chi[\mathcal{E}']$ and exactly one of the pairwise disjoint spaces $h^+(\ell_w)$, $h^0(\ell_w)$ or $h^-(\ell_w)$. Consequently, if a point P is in φ' then it falls into exactly one subspace $\varphi' \cap h^+(\ell_w)$, $\varphi' \cap h^0(\ell_w)$ or $\varphi' \cap h^-(\ell_w)$. Hence, every point $P \in \chi$ belongs to exactly one face of $\mathcal{A}_\chi[\mathcal{E}]$.

The proof works analogously for $\mathcal{H}_\chi$ and the two disjoint spaces $H^+(\ell_w)$ and $H^-(\ell_w)$. □

The Proof of Lemma 2.24

Proof. If χ is k-dimensional then set $\ell(v) = \emptyset$ for every node v of a k-face $\varphi(v)$.

For $(k-1)$-face nodes v the set $\ell(v)$ is already computed correctly during the construction of $\mathcal{G}_\chi[\mathcal{E}]$. To get $\ell(v)$ for the remaining nodes v insert temporary nodes $w(\ell)$, one for every polynomial ℓ in $\mathcal{E}$. Then add directed edges $w(\ell)v$ if and only if v is a $(k-1)$-face node with $\ell \in \ell(v)$. This takes $O(|\mathcal{E}|^k)$ time since there are at most $O(|\mathcal{E}|^k)$ many $(k-1)$-face nodes.

Every node u, visited in a directed depth first search starting from a temporary node $w(\ell)$, fulfills $\varphi(u) \subseteq h^0(\ell)$. Consequently, the polynomial ℓ belongs to $\ell(u)$. Hence, to compute $\ell(u)$ for every k'-face node u with $k' < k-1$ it is sufficient to perform directed depth first search from every temporary node $w(\ell)$ and add ℓ to the annotation of all visited nodes.

There are $|\mathcal{E}|$ temporary nodes $w(\ell)$. Moreover, by [15, 16] there are at most $O(|\mathcal{E}|^{k-1})$ faces in the sub arrangement $\mathcal{A}_{\chi'}[\mathcal{E}]$ given by the $(k-1)$-dimensional slice $\chi' = \chi \cap h^0(\ell)$. Hence, there are $O(|\mathcal{E}|)$ calls to directed depth first search each visiting $O(|\mathcal{E}|^{k-1})$ nodes and $O(|\mathcal{E}|^{k-1})$ edges. Consequently, the process of computing $\ell(v)$ for all nodes $v \in V$ takes $O(|\mathcal{E}|^k)$ time. $\square$

Chapter 3

Algebraic Parameter Characterization

Projective transformations constitute a parameterized transformation class according to Definition 2.8. This means, e. g., that points P in the uncountable set $\mathbb{R}^8$ describe projective transformations.

On the other hand, the dictionary $\mathcal{D}[m](A)$ of all size m transformations of a digital image A is finite. To obtain the transformed image $f(A)[m]$ for a projective transformation f, every index (i, j) in $z(m)$ is transformed to $(x, y) = f^{-1}(i, j)$ and then interpolation $c = A_I(x, y) = A([x], [y])$ is evaluated to assign the color value $f(A)[m](i, j) = c$. It is plain that every point (x, y) falls into the area of exactly one of the pixels in $Pix[n]$ of image A. Imagine some very small perturbation put on f^{-1} resulting in f'^{-1}. The points $f^{-1}(i, j)$ and $f'^{-1}(i, j)$ differ only slightly for all (i, j) causing that $f^{-1}(i, j)$ and $f'^{-1}(i, j)$ fall into the same pixel of $Pix[n]$ for all $(i, j) \in z(m)$. But then it is still the case that $f(A)[m] = f'(A)[m]$.

Consequently, there are classes of points in the parameter space responsible for the same transformation of A and this chapter describes their structure. The convenient structure of $Pix[n]$, the pixels of A which are simple square areas bounded by vertical and horizontal lines, forms the basis of the parameter space partition. Based on the linear pixel boundary the chapter introduces a set $\mathcal{E}$ of linear polynomials to partition the parameter space $\mathbb{R}^8$ into a finite number of subspaces $\varphi_1, \varphi_2, \ldots, \varphi_r$. Then $\varphi_1, \varphi_2, \ldots, \varphi_r$ are shown to form exactly the mentioned classes of points in the parameter space. In fact, points P and P', which fall into the same subspace φ_k for some $k \in \{1, \ldots, r\}$, correspond to transformations f and f' that transform the image A in the same way, i. e., $f(A)[m] = f'(A)[m]$.

The chapter concludes by providing a number of properties for the classes $\varphi_1, \varphi_2, \ldots, \varphi_r$.

3.1 A Parameter Space Partition

This section describes conditions for projective transformations f^{-1} and f'^{-1} that are responsible for different transformed images $f(A)[m]$ and $f'(A)[m]$. Hence, if there is an index $(i,j) \in z(m)$ with

$$A_I(f^{-1}(i,j)) \neq A_I(f'^{-1}(i,j))$$

the ambition of the section is to explain how the parameters P and P' of the transformations f and f' are differentiated.

A necessary condition for the above inequality is that $f^{-1}(i,j)$ and $f'^{-1}(i,j)$ fall into different pixels $pix(i',j')$ and $pix(i'',j'')$ of $Pix[n]$, where n is the size of A. In that case $A_I(f^{-1}(i,j)) \neq A_I(f'^{-1}(i,j))$ if and only if $A(i',j') \neq A(i'',j'')$, i. e., if the colors of the pixels $pix(i',j')$ and $pix(i'',j'')$ differ. However, the aim is to find a natural property that is independent of specific color values of A and consequently, the focus remains just on the necessary condition:

Question 3.1. *Let $m,n \in \mathbb{N}$ be natural numbers, (i,j) be an index in support $z(m)$ and $pix(i',j') \in Pix[n]$ be a pixel given by (i',j') in the support $z(n)$. Which points $P \in \mathbb{R}^8$ describe an inverse projective transformation $f^{-1} = \mathcal{P}(P)$ with $f^{-1}(i,j) \in pix(i',j')$?*

To answer the question consider for all $i',j' \in \{-n,\dots,n+1\}$

$$\mathtt{v}^+_{i'} = \bigcup_{j=-n}^{n} \bigcup_{i=i'}^{n} pix(i,j) \quad \text{and} \quad \mathtt{v}^-_{i'} = \bigcup_{j=-n}^{n} \bigcup_{i=-n}^{i'-1} pix(i,j),$$

$$\mathtt{h}^+_{j'} = \bigcup_{i=-n}^{n} \bigcup_{j=j'}^{n} pix(i,j) \quad \text{and} \quad \mathtt{h}^-_{j'} = \bigcup_{i=-n}^{n} \bigcup_{j=-n}^{j'-1} pix(i,j),$$

subspaces of $\mathbb{R}^2$ that describe the vertical and horizontal borders of pixels $Pix[n]$. Every space $\mathtt{v}^+_{i'}$ contains all pixels $pix(i,j)$ with $i \geq i'$ and $\mathtt{v}^-_{i'}$ the pixels $pix(i,j)$ with $i < i'$. Accordingly, $\mathtt{h}^+_{j'}$ and $\mathtt{h}^-_{j'}$ describe the horizontal partition of $Pix[n]$ into $pix(i,j)$ with $j \geq j'$ and $j < j'$, respectively. Clearly, it is true that $pix(i,j) = \mathtt{v}^+_i \cap \mathtt{v}^-_{i+1} \cap \mathtt{h}^+_j \cap \mathtt{h}^-_{j+1}$.

The following characterizes parameters $P = (p_1,\dots,p_8)^T$ of inverse projective transformations $f^{-1} = \mathcal{P}(P)$ with respect to the location of transformed points $f^{-1}(i,j)$ within the grid of spaces described by $\mathtt{v}^+_{i'}$, $\mathtt{v}^-_{i'}$ and $\mathtt{h}^+_{j'}$, $\mathtt{h}^-_{j'}$.

But first recall that by Definition 2.8 the mapping $\mathcal{P}[\mathcal{F}]$ is not defined on points $P = (p_1,\dots,p_8)^T$ with $p_1p_5 = p_2p_4$ or $p_1p_5 + p_2p_6p_7 + p_3p_4p_8 = p_1p_6p_8 + p_2p_4 + p_3p_5p_7$. Moreover, some projective transformations are undefined on certain indices $(i,j) \in z(m)$. Particularly, this happens for points $P = (p_1,\dots,p_8)^T$ with $ip_7 + jp_8 + 1 = 0$. Confronted with the subsequent analysis of the parameter space, points that fulfill one of the three conditions exhibit some unpleasant difficulties. Accordingly, such points are called invalid:

Definition 3.2. *A point $P \in \mathbb{R}^8$ is called **invalid** for a support $z(m), m \in \mathbb{N}$ if P is a root of at least one of the polynomials*

$$e_1(p_1,\dots,p_8) = p_1p_5 - p_2p_4,$$

$$e_2(p_1,\dots,p_8) = p_1p_5 - p_1p_6p_8 - p_2p_4 + p_2p_6p_7 + p_3p_4p_8 - p_3p_5p_7$$

or

$$e[i,j](p_1,\dots,p_8) = ip_7 + jp_8 + 1, \quad (i,j) \in z(m),$$

*i. e., if $e_1(P) = 0$ or $e_2(P) = 0$ or $e[i,j](P) = 0$ for some $(i,j) \in z(m)$. Otherwise P is called **valid** for $z(m)$.*

If the support $z(m)$ is clear from the context an (in)valid point for $z(m)$ is called (in)valid for short. Clearly, every invalid point is contained in one of the hypersurface $h^0(e_1)$ or $h^0(e_2)$ or $h^0(e[i,j])$ for some $(i,j) \in z(m)$. For a start, this section disregards such exceptional points to make arguments easier to follow. Later, Section 3.2 demonstrates a convenient approach of handling the difficulties introduced by them.

The following definition provides the set of linear polynomials that define the partition of the parameter space:

Definition 3.3. *For natural numbers $m,n \in \mathbb{N}$ and for all $(i,j) \in z(m)$, all $i' \in \{-n,\dots,n+1\}$, respectively all $j' \in \{-n,\dots,n+1\}$, the linear polynomials*

$$\ell^{\mathtt{v}}_{iji'}(p_1,\dots,p_8) = ip_1 + jp_2 + p_3 + (0.5i - ii')p_7 + (0.5j - ji')p_8 + (0.5 - i') \text{ and}$$
$$\ell^{\mathtt{h}}_{ijj'}(p_1,\dots,p_8) = ip_4 + jp_5 + p_6 + (0.5i - ij')p_7 + (0.5j - jj')p_8 + (0.5 - j')$$

are collected in the set

$$\mathcal{E}[m,n] = \{\ell^{\mathtt{v}}_{iji'} \mid (i,j) \in z(m), -n \le i' \le n+1\} \cup$$
$$\{\ell^{\mathtt{h}}_{ijj'} \mid (i,j) \in z(m), -n \le j' \le n+1\}.$$

For $\star \in \{\mathtt{v},\mathtt{h}\}$ each of these linear polynomials $\ell^{\star}_{ijk} \in \mathcal{E}[m,n]$ describes a partition of the parameter space $\mathbb{R}^8$ into two half spaces $H^+(\ell^{\star}_{ijk})$ and $H^-(\ell^{\star}_{ijk})$. Clearly, every point P in the parameter space $\mathbb{R}^8$ is either in the one or the other subspace for every polynomial. It turns out that these locations describe the complete behavior of f^{-1}, i. e., for all $(i,j) \in z(m)$ they determine the pixel in $Pix[n]$ that contains $f^{-1}(i,j)$. The following lemma makes this connection formal:

Lemma 3.4. *Let $m, n \in \mathbb{N}$ be natural numbers to define the support $z(m)$ and the pixels $Pix[n]$. Furthermore let $P \in \mathbb{R}^8$ be a valid point for support $z(m)$. If (i, j) is an index in $z(m)$ and k_1, k_2 are integers with $-n \leq k_1 < k_2 \leq n+1$ then it is true that the inverse projective transformation $f^{-1} = \mathcal{P}(P) \in \mathcal{F}_{\mathtt{p}}$ represented by P maps (i, j) to a point $f^{-1}(i, j)$ in the vertical pixel band $\mathtt{v}_{k_1}^+ \cap \mathtt{v}_{k_2}^-$ if and only if*

$$P \in \left(H^+(\ell_{ijk_1}^{\mathtt{v}}) \cap H^-(\ell_{ijk_2}^{\mathtt{v}})\right) \cup \left(H^+(-\ell_{ijk_1}^{\mathtt{v}}) \cap H^-(-\ell_{ijk_2}^{\mathtt{v}})\right).$$

Analogously, the point $f^{-1}(i, j)$ falls into the horizontal pixel band $\mathtt{h}_{k_1}^+ \cap \mathtt{h}_{k_2}^-$ if and only if

$$P \in \left(H^+(\ell_{ijk_1}^{\mathtt{h}}) \cap H^-(\ell_{ijk_2}^{\mathtt{h}})\right) \cup \left(H^+(-\ell_{ijk_1}^{\mathtt{h}}) \cap H^-(-\ell_{ijk_2}^{\mathtt{h}})\right).$$

Proof. This proof shows only that $f^{-1}(i, j)$ is a point in the given vertical band of pixels. That $f^{-1}(i, j)$ falls also into the horizontal band works analogously.

If $P = (p_1, \ldots, p_8)^T$ is the parameter of $f^{-1} = \mathcal{P}(P)$ then let $(x, y) = f^{-1}(i, j)$ be the transformed point of (i, j). By Definition 2.8 this means

$$x = \frac{ip_1 + jp_2 + p_3}{ip_7 + jp_8 + 1}.$$

Since P is valid it is true that the denominator $e[i, j](P)$ of the above fraction is not zero and thus, let $q = ip_7 + jp_8 + 1 \neq 0$.

It is true that $x \in \mathtt{v}_{k_1}^+ \cap \mathtt{v}_{k_2}^-$ if and only if $k_1 - 0.5 \leq \frac{ip_1+jp_2+p_3}{q} < k_2 - 0.5$. By multiplying with q this is true if and only if $q > 0$ and

$$q(k_1 - 0.5) \leq ip_1 + jp_2 + p_3 < q(k_2 - 0.5),$$

which is equivalent to

$$\begin{aligned}
&ip_1 + jp_2 + p_3 + (0.5i - ik_1)p_7 + (0.5j - jk_1)p_8 + (0.5 - k_1) \geq 0 \text{ and}\\
&ip_1 + jp_2 + p_3 + (0.5i - ik_2)p_7 + (0.5j - jk_2)p_8 + (0.5 - k_2) < 0,
\end{aligned}$$

or if $q < 0$ and

$$q(k_1 - 0.5) \geq ip_1 + jp_2 + p_3 > q(k_2 - 0.5),$$

which is equivalent to

$$\begin{aligned}
&-ip_1 - jp_2 - p_3 - (0.5i - ik_1)p_7 - (0.5j - jk_1)p_8 - (0.5 - k_1) \geq 0 \text{ and}\\
&-ip_1 - jp_2 - p_3 - (0.5i - ik_2)p_7 - (0.5j - jk_2)p_8 - (0.5 - k_2) < 0.
\end{aligned}$$

In other words, $x \in \mathtt{v}_{k_1}^+ \cap \mathtt{v}_{k_2}^-$ if and only if $q > 0$ and $P \in H_1 = H^+(\ell_{ijk_1}^{\mathtt{v}}) \cap H^-(\ell_{ijk_2}^{\mathtt{v}})$ or if $q < 0$ and $P \in H_2 = H^+(-\ell_{ijk_1}^{\mathtt{v}}) \cap H^-(-\ell_{ijk_2}^{\mathtt{v}})$.

The proof is completed by showing that $q > 0$ if $P \in H_1$ and $q < 0$ if $P \in H_2$. For this reason notice that $\ell^{\mathtt{v}}_{ijk_1}(P) - \ell^{\mathtt{v}}_{ijk_2}(P) = kq$ with $k = k_2 - k_1$ being a positive number. In case of $P \in H_1$, i. e., if $\ell^{\mathtt{v}}_{ijk_1}(P) \geq 0$ and $\ell^{\mathtt{v}}_{ijk_2}(P) < 0$, then $kq > 0$ which implies $q > 0$. Reversely, if P is in H_2 where $\ell^{\mathtt{v}}_{ijk_1}(P) \leq 0$ and $\ell^{\mathtt{v}}_{ijk_2}(P) < 0$, then trivially $q < 0$. □

The lemma helps to answer the question raised earlier in this section. Remember, for every $(i', j') \in z(n)$ the pixel $pix(i', j') \in Pix[n]$ is exactly the intersection

$$pix(i', j') = \mathtt{v}^+_{i'} \cap \mathtt{v}^-_{i'+1} \cap \mathtt{h}^+_{j'} \cap \mathtt{h}^-_{j'+1},$$

the overlapping of a vertical and a horizontal band of pixels. The question, whether $f^{-1}(i, j)$ falls into the pixel $pix(i', j')$ can be answered positively if and only if the point P determining $f^{-1} = \mathcal{P}(P)$ is contained in both intersections

$$\begin{gathered}\bigl(H^+(\ell^{\mathtt{v}}_{iji'}) \cap H^-(\ell^{\mathtt{v}}_{iji'+1})\bigr) \quad \cup \quad \bigl(H^+(-\ell^{\mathtt{v}}_{iji'}) \cap H^-(-\ell^{\mathtt{v}}_{iji'+1})\bigr) \\ \text{and} \\ \bigl(H^+(\ell^{\mathtt{h}}_{ijj'}) \cap H^-(\ell^{\mathtt{h}}_{ijj'+1})\bigr) \quad \cup \quad \bigl(H^+(-\ell^{\mathtt{h}}_{ijj'}) \cap H^-(-\ell^{\mathtt{h}}_{ijj'+1})\bigr).\end{gathered}$$

Next, Lemma 3.4 enables a complete characterization of the parameter space partition into the subspaces $\varphi_1, \ldots, \varphi_r$, such that any two points P and P' in the same subspace $\varphi_k, k \in \{1, \ldots, r\}$ give $f^{-1} = \mathcal{P}(P)$ and $f'^{-1} = \mathcal{P}(P')$ with $f(A)[m] = f'(A)[m]$. In particular, $\varphi_1, \ldots, \varphi_r$ are defined by the polynomials $\mathcal{E}[m, n]$:

Corollary 3.5. *For given natural numbers $m, n \in \mathbb{N}$ let $\mathcal{E}[m, n]$ be the set of linear polynomials of Definition 3.3 and let $\mathcal{H}[\mathcal{E}[m, n]]$ be the corresponding arrangement and $\varphi \in \mathcal{H}[\mathcal{E}[m, n]]$ be any contained face. Furthermore, let P and P' be two valid points in φ. Then it is true that the inverse projective transformations $f^{-1} = \mathcal{P}(P)$ and $f'^{-1} = \mathcal{P}(P')$ fulfill*

$$f^{-1}(i, j) \in pix(i', j') \Leftrightarrow f'^{-1}(i, j) \in pix(i', j').$$

for every $(i, j) \in z(m)$ and all $(i', j') \in z(n)$.

Proof. The statement follows directly from Lemma 3.4. Let P and P' be points in $\mathbb{R}^8$ and let them represent $f^{-1} = \mathcal{P}(P)$ and $f'^{-1} = \mathcal{P}(P')$.

If P and P' are contained in the same face $\varphi \in \mathcal{H}[\mathcal{E}[m, n]]$, then for all $(i, j) \in z(m)$ and all $i' \in \{-n, \ldots, n + 1\}$ (all $j' \in \{-n, \ldots, n + 1\}$) it is true that P and P' are either both contained in $\bigl(H^+(\ell^{\mathtt{v}}_{iji'}) \cap H^-(\ell^{\mathtt{v}}_{iji'+1})\bigr) \cup \bigl(H^+(-\ell^{\mathtt{v}}_{iji'}) \cap H^-(-\ell^{\mathtt{v}}_{iji'+1})\bigr)$ (respectively, in $\bigl(H^+(\ell^{\mathtt{h}}_{ijj'}) \cap H^-(\ell^{\mathtt{h}}_{ijj'+1})\bigr) \cup \bigl(H^+(-\ell^{\mathtt{h}}_{ijj'}) \cap H^-(-\ell^{\mathtt{h}}_{ijj'+1})\bigr)$) or both are not contained in this subspaces.

Particularly, this means that $f^{-1}(i, j)$ falls into the pixel band intersection $pix(i', j') = \mathtt{v}^+_{i'} \cap \mathtt{v}^-_{i'+1} \cap \mathtt{h}^+_{j'} \cap \mathtt{h}^-_{j'+1}$ if and only if $f'^{-1}(i, j)$ does. □

For short, the arrangement $\mathcal{H}[\mathcal{E}[m,n]]$ is simply denoted by $\mathcal{H}[m,n]$. Then, the corollary gives a strong connection between the faces of the arrangement $\mathcal{H}[m,n]$ and the dictionary $\mathcal{D}[m](A)$ for any given image A of size n. It states that two points P and P' coming from the same face correspond to the same image in $\mathcal{D}[m](A)$, because they represent inverse projective transformations f^{-1} and f'^{-1} with $f(A)[m] = f'(A)[m]$. An impression of $\mathcal{H}[m,n]$ can be revealed from Figure 3.1 that shows $\mathcal{H}_\chi[2,2]$, a cut of the arrangement $\mathcal{H}[m,n]$ with $m = n = 2$ in the slice $\chi = \bigcap_{i=1}^{6} h^0(\ell_i)$, the intersection of hyperplanes given by the polynomials

$$\begin{aligned} \ell_1(p_1,\dots,p_8) &= p_3, & \ell_2(p_1,\dots,p_8) &= p_4 + p_2, & \ell_3(p_1,\dots,p_8) &= p_5 - p_1, \\ \ell_4(p_1,\dots,p_8) &= p_6, & \ell_5(p_1,\dots,p_8) &= p_7, & \ell_6(p_1,\dots,p_8) &= p_8. \end{aligned}$$

In fact, χ is formed by all points P describing projective transformations that consist of scaling and rotation only. Hence χ is the same slice as $\chi_{\texttt{sr}}$ defined in the next chapter.

3.2 Properties of Faces

Before the next chapter develops algorithms for image matching by the use of the new parameter space characteristic, this section studies some structural properties of $\mathcal{H}[m,n]$. This means, a bunch of useful properties of the faces in $\mathcal{H}[m,n]$ are discussed to be used in algorithmic approaches to projective image matching. The first part lists a number of face characteristics which imply that every face in $\mathcal{H}[m,n]$ has at least a small amount of *volume* in $\mathbb{R}^8$. Then the second part provides the existence of at least one valid point in every face of $\mathcal{H}[m,n]$. Finally the last part combines the results to provide a very simple manner of enumerating the transformed images in the dictionary $\mathcal{D}[m](A)$ for arbitrary images A of size n.

3.2.1 Faces have Volume

To provide a lower bound on the faces volume the next lemma shows, as an intermediate step, that all faces in $\mathcal{H}[m,n]$ have a positive volume at all. This is not at once clear from the definition. For example, if $\mathcal{E}[m,n]$ contains two polynomials ℓ_1 and ℓ_2 with $\ell_1(P) = -\ell_2(P)$ for all $P \in \mathbb{R}^8$ then the intersection $H^+(\ell_1) \cap H^+(\ell_2)$ coincides with the hyperplane $h^0(\ell_1)$, a space of zero volume. Consequently, all faces of $\mathcal{H}[m,n]$ situated in the intersection $H^+(\ell_1) \cap H^+(\ell_2)$ would have zero volume, too. There are even more complex cases of possible half space intersection that leads to zero volume faces in $\mathcal{H}[m,n]$. The following lemma proves all of them impossible:

Lemma 3.6 (Proof in Section 3.3). *Let m, n be natural numbers, $\mathcal{H}[m,n]$ be the arrangement of parameter space $\mathbb{R}^8$ and let φ be any face in $\mathcal{H}[m,n]$. There is a*

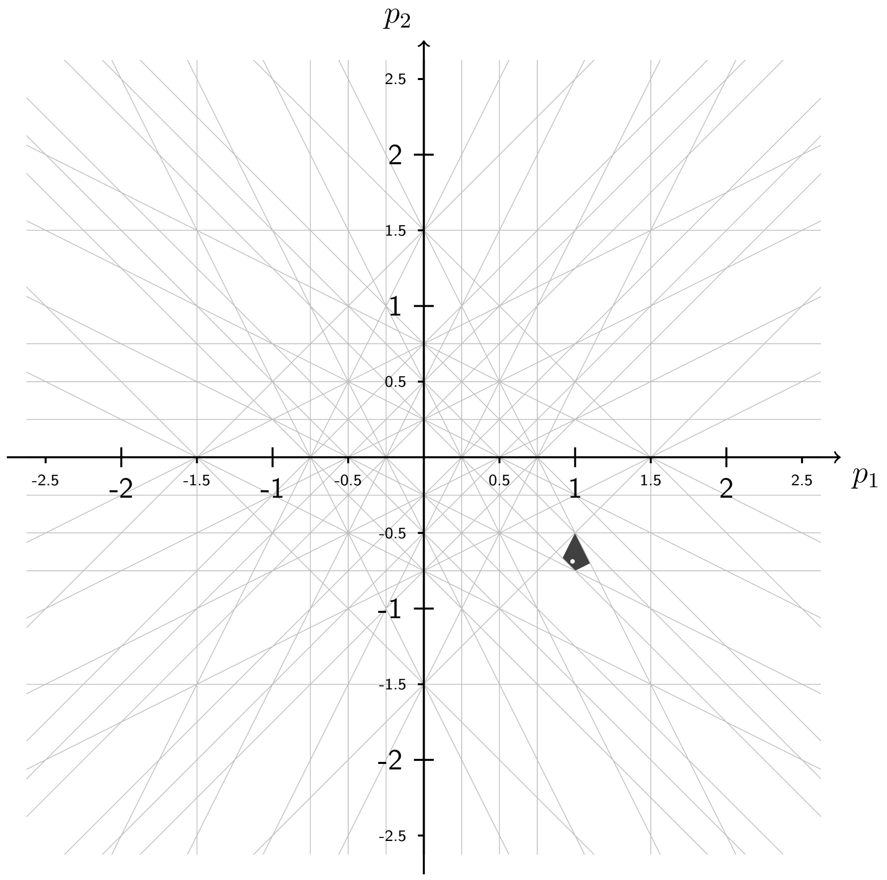


Figure 3.1: A slice χ given by $p_4 = -p_2$, $p_5 = p_1$, $p_3 = p_6 = p_7 = p_8 = 0$ which exactly describes all combinations of scaling and rotation. The figure shows the arrangement $\mathcal{H}_\chi[2,2]$ given by the polynomials $\mathcal{E}[2,2]$. For an image A of size two, like the one in Figure 2.1, each face corresponds to one element of dictionary $\mathcal{D}[\mathcal{F}_{\texttt{sr}}, 2](A)$ shown in Figure 2.2. Particularly, the black face corresponds to the double-framed transformation of A Figure 2.2 because it contains the (white) point representing the inverse of a scaling of 0.83 and a rotations with 35°.

point $P \in \varphi$ and a real radius $r > 0$ such that every point Q with $\|P - Q\| < r$ belongs to φ, too. Consequently, every face in $\mathcal{H}[m,n]$ has a positive volume.

Providing a lower bound on the faces volume still requires a number of further intermediate steps. The following establishes a *bounding box* in $\mathbb{R}^8$ that contains at least a fraction of any face in $\mathcal{H}[m,n]$. The idea is to choose a box that contains all vertices defined by the intersection of hyperplanes given by polynomials in $\mathcal{E}[m,n]$. Every face that is incident to some vertex is then intersected by the box. That such a bounding box intersects every face would follow from the following lemma:

Lemma 3.7 (Proof in Section 3.3)**.** *Let $m, n \in \mathbb{N}$ be natural numbers, $\mathcal{H}[m,n]$ be the arrangement of the parameter space and let φ be any face in $\mathcal{H}[m,n]$. Then there is a set $\{\ell_1, \ldots, \ell_8\}$ of eight linearly independent polynomials in $\mathcal{E}[m,n]$, such that the intersection*

$$h^0(\ell_1) \cap \ldots \cap h^0(\ell_8)$$

is a vertex contained in the boundary $\hat{\varphi}$ of φ.

Because all faces are incident to some vertex defined by hyperplane intersection, there exists a hypercube that serves as a bounding box, i. e., which intersects every face of $\mathcal{H}[m,n]$. The hypercube just makes sure that it contains a little more than every vertex. This gives the first structural property of $\mathcal{E}[m,n]$:

Theorem 3.8 (**Bounding-Box Property**, Proof in Section 3.3)**.** *Let $m, n \in \mathbb{N}$ be natural numbers and $\mathcal{H}[m,n]$ be the arrangement of parameter space $\mathbb{R}^8$ given by m and n. If $r = 1.5 \cdot 10^7 m^6 n^3$ then the hypercube $\mathcal{Q}_r$ of radius r centered at the origin contains at least one point from every face in $\mathcal{H}[m,n]$.*

By the theorem it is reasonable to restrict further constructions to the hypercube, which is subsequently denoted by $\mathcal{Q}_{m,n}$, i. e., $\mathcal{Q}_{m,n} = \mathcal{Q}_{1.5 \cdot 10^7 m^6 n^3}$. Accordingly, it is possible to consider a bounded version of the arrangement:

$$\mathcal{H}_{\mathcal{Q}_{m,n}}[m,n] = \{\varphi \cap \mathcal{Q}_{m,n} \mid \varphi \in \mathcal{H}[m,n]\}.$$

The set $\mathcal{H}_{\mathcal{Q}_{m,n}}[m,n]$ contains at least a fraction of every face in $\mathcal{H}[m,n]$. However, all faces of $\mathcal{H}_{\mathcal{Q}_{m,n}}[m,n]$ are of finite volume because $\mathcal{Q}_{m,n}$ has finite volume. That means they must be bounded by hyperplanes from all sides. The faces that have been of infinite volume in $\mathcal{H}[m,n]$ are now limited by the hyperplanes which define $\mathcal{Q}_{m,n}$. This fact implies a certain regular structure of the faces in $\mathcal{H}_{\mathcal{Q}_{m,n}}[m,n]$ which will be used to infer the lower bound on the faces volume. In particular, the extremal points of every face are vertices. Because of the dimension of $\mathbb{R}^8$ this in turn means that the boundary of every face has a minimum number of nine vertices:

Lemma 3.9 (Proof in Section 3.3)**.** *Let $m, n \in \mathbb{N}$ be natural numbers, $\mathcal{H}_{\mathcal{Q}_{m,n}}[m,n]$ be the bounded arrangement and φ be any face in $\mathcal{H}_{\mathcal{Q}_{m,n}}[m,n]$. There are at least nine linearly independent vertices $V_1, \ldots, V_9$ defined by the intersection of hyperplanes given by the polynomials in $\mathcal{E}[m,n]$ and the polynomials that describe $\mathcal{Q}_{m,n}$ such that the boundary $\hat{\varphi}$ of φ contains V_1 to V_9.*

The following proves a lower positive bound on the volume of the faces contained in $\mathcal{H}_{\mathcal{Q}_{m,n}}[m,n]$:

Lemma 3.10 (Proof in Section 3.3)**.** *Let $m, n \in \mathbb{N}$ be natural numbers, $\mathcal{H}_{\mathcal{Q}_{m,n}}[m,n]$ be the bounded arrangement and φ be any face in $\mathcal{H}_{\mathcal{Q}_{m,n}}[m,n]$. Then φ contains an 8-ball with radius*

$$r \geq \frac{1}{4.5 \cdot 10^8 m^7 n^3}.$$

Since the faces in $\mathcal{H}_{\mathcal{Q}_{m,n}}[m,n]$ are subspaces of those in $\mathcal{H}[m,n]$ the above bound can simply be transfered to the faces of $\mathcal{H}[m,n]$. This gives the second fundamental structural property:

Corollary 3.11 (**Volume Property**, Without proof)**.** *Let $m, n \in \mathbb{N}$ be natural numbers, $\mathcal{H}[m,n]$ be the arrangement of the parameter space and φ be any face in $\mathcal{H}[m,n]$. Then φ contains an 8-ball with radius*

$$r \geq \frac{1}{4.5 \cdot 10^8 m^7 n^3}.$$

3.2.2 Faces have Valid Points

The question in focus of this part is which faces of $\mathcal{H}[m,n]$ are relevant for the dictionary $\mathcal{D}[m](A)$ of projective image transformations of a given image A. Theoretically it would be possible that there are faces φ which contain only points $P = (p_1, \ldots, p_8)^T$ that are invalid according to Definition 3.2. Such a face φ would not contribute a meaningful projective transformation of A.

This part of the section shows that each face in $\mathcal{H}[m,n]$ has a valid point P for support $z(m)$. The idea behind that statement is simple. Remember that every $\varphi \in \mathcal{H}[m,n]$ has a positive volume. But any surface in $\mathbb{R}^8$ has zero volume. Consequently, a finite number of surfaces can never eat up the whole volume of φ and there remain points which fulfill neither $e_1(P) = 0$ nor $e_2(P) = 0$ nor $e[i,j](P) = 0$ for any $(i,j) \in z(m)$. The following lemma states this formal:

Lemma 3.12 (Proof in Section 3.3)**.** *Let $m, n \in \mathbb{N}$ be natural numbers, $\mathcal{H}[m,n]$ be the arrangement of the parameter space and φ be any face in $\mathcal{H}[m,n]$. Moreover, let $\{q_1, \ldots, q_u\}$ be a finite set of arbitrary (not necessarily linear) polynomials. Then there is a point $P \in \varphi$ such that $P \notin h^0(q_k)$ for all $k \in \{1, \ldots, u\}$.*

The general condition stated in the lemma can now be used to conclude the third basic structural property of faces in $\mathcal{H}[m,n]$:

Corollary 3.13 (**Valid-Points Property**, Without proof)**.** *Let $m, n \in \mathbb{N}$ be natural numbers, $\mathcal{H}[m,n]$ be the arrangement of the parameter space and φ be any face in $\mathcal{H}[m,n]$. Then there is a valid point $P \in \varphi$.*

The corollary means that every face contains a point P which represents an inverse projective transformation f^{-1} that is defined on all indices $(i,j) \in z(m)$. This means that there is a natural connection between the faces in $\mathcal{H}[m,n]$ and the dictionary $\mathcal{D}[m](A)$ of image transformations for a given image A of size n. In the next chapter this connection is formalized and applied to design polynomial time projective image matching algorithms. Figure 3.2 (a) shows valid points for all faces in a small subspace of $\mathbb{R}^8$.

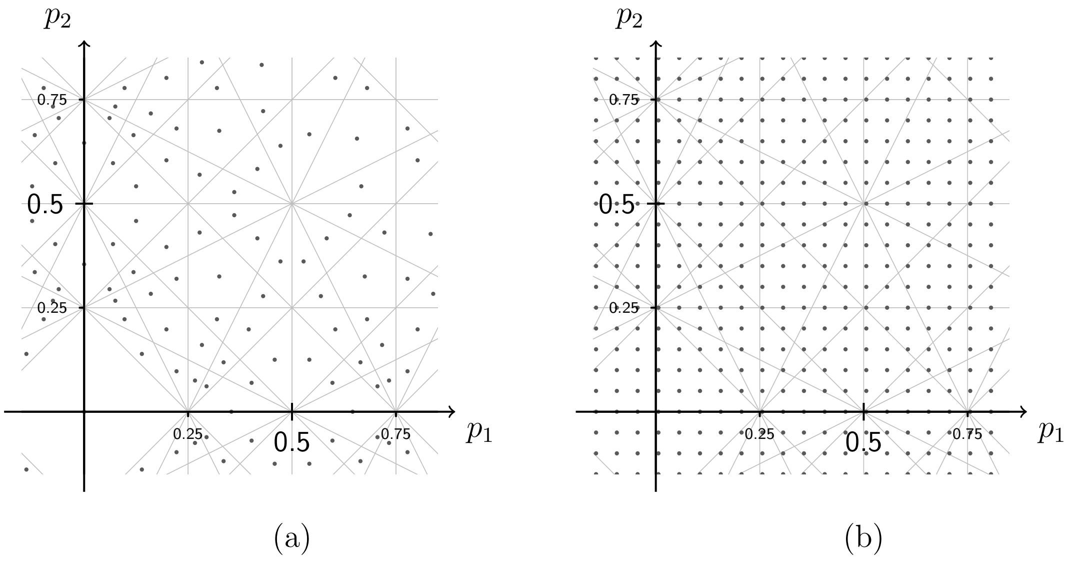


Figure 3.2: (a) Valid points for all faces of $\mathcal{H}_\chi[2,2]$ in the slice χ of Figure 3.1. (b) The lattice $\mathcal{L}[2,2]$ restricted to χ. Despite the regular structure, every face is hit by at least one lattice point. For visual convenience, parameters have been changed to $L = 1$ and $d = 0.05$.

3.2.3 Sampling Representative Points

The last part of this section affiliates the previous three structural properties of faces in $\mathcal{H}[m,n]$ to design a mathematical tool to be applied in projective image matching algorithms. It provides a lattice $\mathcal{L}$ such that every face $\varphi \in \mathcal{H}[m,n]$ is hit by at least one valid point $P \in \mathcal{L}$. The structure of the lattice is demonstrated in Figure 3.2 (b).

Definition 3.14. *For given natural numbers $m, n \in \mathbb{N}$ let $L = 3 \cdot 10^{16} m^{13} n^6$ and $d = (1.5 \cdot 10^9 m^7 n^3 + 0.5)^{-1}$. Then $\mathcal{L}[m,n]$ is the lattice that exactly contains all points*

$$\begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \\ p_7 \\ p_8 \end{pmatrix} = d \cdot \begin{pmatrix} t_1 + 0.125L^{-2} \\ t_2 \\ t_3 \\ t_4 \\ t_5 + 0.5 \\ t_6 \\ t_7 \\ t_8 \end{pmatrix}$$

for combinations of integers $t_1, \dots, t_8 \in \{-L, \dots, L\}$. For all such integer combinations $t_1, \dots, t_8$ let $\mathcal{L}[m,n](t_1, \dots, t_8)$ denote the point P given by $t_1, \dots, t_8$.

The points in $\mathcal{L}[m,n]$ have the following properties:

Theorem 3.15 (**Lattice Property**, Proof in Section 3.3)**.** *Let $m, n \in \mathbb{N}$. Then*

1. *for every face $\varphi \in \mathcal{H}[m,n]$ there is a point $P \in \mathcal{L}[m,n]$ with $P \in \varphi$,*
2. *all points $P \in \mathcal{L}[m,n]$ are valid for support $z(m)$ and*
3. *the cardinality of $\mathcal{L}[m,n]$ is less than $10^{132} \cdot m^{104} n^{48}$.*

3.3 Technical Proofs

The Proof of Lemma 3.6

Proof. Begin with the following property: For every point $P \in \varphi$ there is a vector $V(P) = (0,0,r(P),0,0,s(P),0,0)$ with $r(P), s(P) \in \mathbb{R}^+$ such that for all $t \in [0,1)$ every point

$$Q = P + t \cdot V(P)$$

belongs to φ.

By definition φ is the intersection $H^{s_1}(\ell_1) \cap \ldots \cap H^{s_w}(\ell_w)$ of the half spaces defined by the polynomials $\ell_1, \ldots, \ell_w \in \mathcal{E}[m,n]$ and the signs $s_1, \ldots, s_w \in \{+,-\}$. For every $k \in \{0, \ldots, w\}$ let φ_k be the face which is given by the intersection $H^{s_1}(\ell_1) \cap \ldots \cap H^{s_k}(\ell_k)$ of only k hyperplanes. Clearly, $\varphi \subseteq \varphi_k$ for all $k \in \{0, \ldots, w\}$. Then, the above statement is shown via induction over the number k:

$k = 0$: Then $\varphi_0 = \mathbb{R}^8$ is the whole parameter space and thus, choose for every point P simply $r_0(P) = s_0(P) = 1$.

$k > 0$: Assume the validity of the argument for the face φ_{k-1}. Hence, for every point P in φ_{k-1} there are values $r_{k-1}(P), s_{k-1}(P) > 0$ such that all points Q defined as above are also in φ_{k-1}.

Adding the half space $H^{s_k}(\ell_k)$ gives $\varphi_k = \varphi_{k-1} \cap H^{s_k}(\ell_k)$ by excluding a (possibly empty) set of points from φ_{k-1}. But since $\varphi \neq \emptyset$ and $\varphi = \varphi_w \subseteq \varphi_k$ there are points in φ_k. The proof shows for every point P in φ_k that the claimed condition still holds. So assume that $P = (p_1, \ldots, p_8)^T$ and that $r_{k-1}(P)$ is the value defined for $k-1$ hyperplanes.

First assume that $\ell_k = \ell^{\mathrm{v}}_{iji'}$.

$s_k = +$: Then $P \in H^+(\ell_k)$ which means

$$0 \leq ip_1 + jp_2 + p_3 + (0.5i - ii')p_7 + (0.5j - ji')p_8 + (0.5 - i').$$

But since every point $Q = P + t \cdot V(P)$ has an even larger p_3-component it follows that $Q \in H^+(\ell_k)$ as well and subsequently it follows that $Q \in \varphi_k$. Consequently, let $r_k(P) = r_{k-1}(P)$ and $s_k(P) = s_{k-1}(P)$.

$s_k = -$: It is true that $P \in H^-(\ell_k)$ which means

$$0 > ip_1 + jp_2 + p_3 + (0.5i - ii')p_7 + (0.5j - ji')p_8 + (0.5 - i').$$

In this case define

$$\begin{aligned}&r_k(P) = \min\{r_{k-1}(P),\\ &- ip_1 - jp_2 - p_3 - (0.5i - ii')p_7 - (0.5j - ji')p_8 - (0.5 - i')\}\end{aligned}$$

and let $s_k(P) = s_{k-1}(P)$. Clearly, $r_k(P) > 0$ and all points Q obtained by the new $r_k(P)$ and $s_k(P)$ are also in $H^-(\ell_k)$ and by that in φ_k.

The case of $\ell_k = \ell^{\mathtt{h}}_{ijj'}$ works analogously with $s_k(P)$.

The claim holds also for $\varphi = \varphi_w$. Since φ is not empty, consider any contained point P. If there is a number $r > 0$ such that every point Q with $\|P - Q\| < r$ is in φ, then the argument is complete.

Otherwise consider

$$\begin{aligned}Q_1 &= P + 0.5 \cdot V(P) \quad \text{and}\\ Q_2 &= P + 0.75 \cdot V(P).\end{aligned}$$

By the above argumentation it is true that $Q_1, Q_2 \in \varphi$. Assume that even for Q_1 there is no number $r > 0$ such that all points Q with $\|Q_1 - Q\| < r$ are in φ. Then, there must be a polynomial ℓ_k with $k \in \{1, \ldots, w\}$ such that $\ell_k(Q_1) = 0$.

Assume again that $\ell_k = \ell^{\mathtt{v}}_{iji'}$ for some $(i, j) \in z(m)$ and $-n \le i' \le n + 1$. Then

$$0 = ip_1 + jp_2 + (p_3 + 0.5r(P)) + (0.5i - ii')p_7 + (0.5j - ji')p_8 + (0.5 - i')$$

and thus, $\ell_k(P) < 0$ and $\ell_k(Q_2) > 0$. This means $P \in H^-(\ell_k)$ and $Q_2 \in H^+(\ell_k)$. But then P and Q_2 are in different half spaces with respect to ℓ_k although they both belong to φ. This is a contradiction. Analogously, one can show that ℓ_k cannot be $\ell^{\mathtt{h}}_{ijj'}$ for any $(i, j) \in z(m)$ and $-n \le j' \le n + 1$. Thus, an environment of positive volume around Q_1 belongs to φ. □

The Proof of Lemma 3.7

Proof. Assume the contrary, i. e., that there is a face φ, which has no such vertex in its boundary. The face φ is separated from other faces by hyperplanes. Consider χ, an intersection of a maximum number k of linearly independent hyperplanes in the boundary of φ. According to Definition 2.17 $\chi = h^0(\ell_1) \cap \ldots \cap h^0(\ell_k)$ is a $(8 - k)$-slice. The slice χ is at least a single hyperplane if $k = 1$. Because χ is not a vertex it follows also that $k < 8$. This means that χ is either a line, a plane, or a slice of higher dimension.

If there is a hyperplane $h^0(\ell)$ linearly independent from $h^0(\ell_1), \dots, h^0(\ell_k)$ then $h^0(\ell)$ intersects χ. Even if $h^0(\ell)$ does not belong to the boundary of φ its existence implies another linearly independent hyperplane $h^0(\ell')$ which is a boundary hyperplane of φ. This in turn would imply that k is not the maximum cardinality and thus a contradiction to the assumption that $\hat{\varphi}$ contains no vertex.

The following will show the existence of $h^0(\ell)$. Because $1 \leq k < 8$ it is true that the intersection χ contains a point $S = (s_1, \dots, s_8)^T$ and a line $L = \{P \mid P = S + tV, t \in R\}$ that starts at S and follows some non-zero vector V. If there exists a hyperplane $h^0(\ell)$ defined by $\ell \in \mathcal{E}[m, n]$ that intersects the line L in exactly one point then $h^0(\ell)$ is linearly independent to $h^0(\ell_1), \dots, h^0(\ell_k)$.

Let $V = (v_1, \dots, v_8)^T$. If at least one of v_1, v_2, v_3, v_7, v_8 is non-zero, then consider the hyperplane $h^0(\ell)$ defined by $\ell = \ell^{\mathtt{v}}_{iji'}$ with

$$\ell^{\mathtt{v}}_{iji'} = ip_1 + jp_2 + p_3 + (0.5i - ii')p_7 + (0.5j - ji')p_8 + (0.5 - i')$$

for random $-m \leq i, j < m$ and $-n \leq i' \leq n + 1$. The intersection between L and $h^0(\ell^{\mathtt{v}}_{iji'})$ is defined by t with

$$t = -\frac{N}{D} = -\frac{is_1 + js_2 + s_3(0.5i - ii')s_7 + (0.5j - ji')s_8 + (0.5 - i')}{iv_1 + jv_2 + v_3 + (0.5i - ii')v_7 + (0.5j - ji')v_8}.$$

If this fraction has a non-zero denominator D and thus, determines t to one value of $\mathbb{R}$, then t describes the intersection point $P = S + tV$. This is certainly the case if only $v_3 \neq 0$.

Otherwise, if $D = 0$ then L is either a subspace of the hyperplane $h^0(\ell)$ or they do not intersect at all. However, in that case at least one v_1, v_2, v_7 or v_8 is non-zero and then there is another hyperplane which intersects L. Particularly, if

- $v_1 \neq 0$ or $v_7 \neq 0$ then $h^0(\ell^{\mathtt{v}}_{(i+1)ji'})]$ or if
- $v_2 \neq 0$ or $v_8 \neq 0$ then $h^0(\ell^{\mathtt{v}}_{i(j+1)i'})]$

represents a hyperplane which surely intersects L in a unique point.

If all, $v_1 = v_2 = v_3 = v_7 = v_8 = 0$, then at least one v_4, v_5 or v_6 is not zero. Consider the polynomials

$$\ell = \ell^{\mathtt{h}}_{ijj'} = ip_4 + jp_5 + p_6 + (0.5i - ij')p_7 + (0.5j - jj')p_8 + (0.5 - j')$$

for random $-m \leq i, j < m$ and $-n \leq j' \leq n + 1$. The intersection points are determined by

$$t = -\frac{N}{D} = -\frac{is_4 + js_5 + s_6 + (0.5i - ij')s_7 + (0.5j - jj')s_8 + (0.5 - j')}{iv_4 + jv_5 + v_6}.$$

Again, if $D \neq 0$ then the fraction determines t. Otherwise, if D is zero, then at least one v_4 or v_5 is not zero. Consequently, either $h^0(\ell^{\mathtt{h}}_{(i+1)jj'})$ or $h^0(\ell^{\mathtt{h}}_{i(j+1)j'})$ gives an intersecting hyperplane. □

The Proof of Theorem 3.8

Proof. Let $\varphi \in \mathcal{H}[m,n]$. By Lemma 3.7 it is true that φ's boundary contains at least one vertex V defined by hyperplane intersection. If V is not part of φ, then every ϵ-environment around V with $\epsilon > 0$ intersects φ. Hence, if the hypercube of the lemma contains only an ϵ more than every vertex defined by the intersection of hyperplanes given by polynomials in $\mathcal{E}[m,n]$, than the lemma is true.

According to Definition 2.17 every vertex $V = (p_1, \ldots, p_8)^T$ is the intersection of eight linearly independent hyperplanes $h^0(\ell_1), \ldots, h^0(\ell_8)$ of polynomials $\ell_1, \ldots, \ell_8 \in \mathcal{E}[m,n]$. Consequently, $\ell_k(V) = 0$ for all $1 \leq k \leq 8$. This remains true if multiplied by two, i.e., $2\ell_k(V) = 0$. Hence, if $\ell_k = \ell^{\mathrm{v}}_{iji'}$ for some $(i,j) \in z(m)$ and $i' \in \{-n, \ldots, n+1\}$ then

$$0 = 2ip_1 + 2jp_2 + 2p_3 + (i - 2ii')p_7 + (j - 2ji')p_8 + (1 - 2i').$$

Clearly, the above equation has the form

$$m_9 = m_1p_1 + m_2p_2 + m_3p_3 + m_4p_4 + m_5p_5 + m_6p_6 + m_7p_7 + m_8p_8$$

for some integer coefficients m_1 to m_9. The same holds if $\ell_k = \ell^{\mathrm{h}}_{ijj'}$ for some $(i,j) \in z(m)$ and $j' \in \{-n, \ldots, n+1\}$. Hence, for all $k \in \{1, \ldots, 8\}$ there are integers $m_{k,1}, \ldots, m_{k,9}$ such that it is true

$$m_{k,9} = m_{k,1}p_1 + m_{k,2}p_2 + m_{k,3}p_3 + m_{k,4}p_4 + m_{k,5}p_5 + m_{k,6}p_6 + m_{k,7}p_7 + m_{k,8}p_8.$$

Consequently, V fulfills

$$\begin{pmatrix} m_{1,9} \\ m_{2,9} \\ m_{3,9} \\ m_{4,9} \\ m_{5,9} \\ m_{6,9} \\ m_{7,9} \\ m_{8,9} \end{pmatrix} = \begin{pmatrix} m_{1,1} & m_{1,2} & m_{1,3} & m_{1,4} & m_{1,5} & m_{1,6} & m_{1,7} & m_{1,8} \\ m_{2,1} & m_{2,2} & m_{2,3} & m_{2,4} & m_{2,5} & m_{2,6} & m_{2,7} & m_{2,8} \\ m_{3,1} & m_{3,2} & m_{3,3} & m_{3,4} & m_{3,5} & m_{3,6} & m_{3,7} & m_{3,8} \\ m_{4,1} & m_{4,2} & m_{4,3} & m_{4,4} & m_{4,5} & m_{4,6} & m_{4,7} & m_{4,8} \\ m_{5,1} & m_{5,2} & m_{5,3} & m_{5,4} & m_{5,5} & m_{5,6} & m_{5,7} & m_{5,8} \\ m_{6,1} & m_{6,2} & m_{6,3} & m_{6,4} & m_{6,5} & m_{6,6} & m_{6,7} & m_{6,8} \\ m_{7,1} & m_{7,2} & m_{7,3} & m_{7,4} & m_{7,5} & m_{7,6} & m_{7,7} & m_{7,8} \\ m_{8,1} & m_{8,2} & m_{8,3} & m_{8,4} & m_{8,5} & m_{8,6} & m_{8,7} & m_{8,8} \end{pmatrix} \cdot \begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \\ p_7 \\ p_8 \end{pmatrix},$$

hence $N = M \cdot V$ with integer vector N and integer matrix M. Solving this system of equations gives that $p_1 = \frac{u_1}{\det(M)}, \ldots, p_8 = \frac{u_8}{\det(M)}$ are fractions of integers u_1 to u_8 and integer $\det(M)$, the determinant of M. For all $1 \leq k \leq 8$ the form of u_k is

$$u_k = m_{1,9}\tilde{m}_{k,1} + \ldots + m_{8,9}\tilde{m}_{k,8}$$

where every $\tilde{m}$ is an entry of M's adjunct and by definition a seven by seven sub determinant of M. Consequently, $\tilde{m}$ is a sum of 5040 products of each seven integer entries of M. Thereby each product fixes seven of M's eight columns and selects one entry from each fixed column. Thus, for all $k \in \{0, \dots, 8\}$ it is true that

$$\begin{aligned} |m_{k,1}| &\le 2m, \\ |m_{k,2}| &\le 2m, \\ |m_{k,3}| &\le 2, \\ |m_{k,4}| &\le 2m, \\ |m_{k,5}| &\le 2m, \\ |m_{k,6}| &\le 2, \\ |m_{k,7}| &\le m(2n+1) \quad \text{and} \\ |m_{k,7}| &\le m(2n+1). \end{aligned}$$

For $n \ge 4$ every $\tilde{m}$ is an integer with absolute value bounded from above by $10^6 m^6 n^2$. Because $|m_{k,9}| \le 2(n+1)$ it follows that $|u_k| \le 1.5 \cdot 10^7 m^6 n^3$.

The absolute value of V's coordinates is not larger than $1.5 \cdot 10^7 m^6 n^3$ because (1) the coordinates of V are $\frac{u_k}{\det(M)}$, (2) an upper bound on $|u_k|$ is known and (3) $\det(M) \ge 1$. □

The Proof of Lemma 3.9

Proof. By the Bounding-Box Property 3.8 the volume of φ is finite. It is also not zero by Lemma 3.6. Since φ is an eight-dimensional polyhedron the lemma is implied by the following simple property: A two-dimensional polyhedron is a polygon, e. g., it is bounded by at least three lines which share at least three vertices. A three-dimensional polyhedron is bounded by at least four two-dimensional polyhedron which share at least four vertices. The continued argumentation gives that an eight-dimensional polyhedron is bounded by at least nine seven-dimensional polyhedron which share nine vertices.

Alternatively, notice that any combination of eight linearly independent vertices $V_1, \dots, V_8$ are contained in an eight-dimensional hyperplane $h^0(\ell)$ given by a polynomial

$$\ell(p_1, \dots, p_8) = m_1 p_1 + \ldots + m_8 p_8 + m_9$$

with m_1 to m_9 being the solution of the following system of linear equations

$$\begin{pmatrix} -m_9 \\ -m_9 \\ -m_9 \\ -m_9 \\ -m_9 \\ -m_9 \\ -m_9 \\ -m_9 \end{pmatrix} = \begin{pmatrix} p_{1,1} & p_{1,2} & p_{1,3} & p_{1,4} & p_{1,5} & p_{1,6} & p_{1,7} & p_{1,8} \\ p_{2,1} & p_{2,2} & p_{2,3} & p_{2,4} & p_{2,5} & p_{2,6} & p_{2,7} & p_{2,8} \\ p_{3,1} & p_{3,2} & p_{3,3} & p_{3,4} & p_{3,5} & p_{3,6} & p_{3,7} & p_{3,8} \\ p_{4,1} & p_{4,2} & p_{4,3} & p_{4,4} & p_{4,5} & p_{4,6} & p_{4,7} & p_{4,8} \\ p_{5,1} & p_{5,2} & p_{5,3} & p_{5,4} & p_{5,5} & p_{5,6} & p_{5,7} & p_{5,8} \\ p_{6,1} & p_{6,2} & p_{6,3} & p_{6,4} & p_{6,5} & p_{6,6} & p_{6,7} & p_{6,8} \\ p_{7,1} & p_{7,2} & p_{7,3} & p_{7,4} & p_{7,5} & p_{7,6} & p_{7,7} & p_{7,8} \\ p_{8,1} & p_{8,2} & p_{8,3} & p_{8,4} & p_{8,5} & p_{8,6} & p_{8,7} & p_{8,8} \end{pmatrix} \cdot \begin{pmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ m_5 \\ m_6 \\ m_7 \\ m_8 \end{pmatrix},$$

where $V_k = (p_{k,1}, \ldots, p_{k,8})$ for all $k \in \{1, \ldots, 8\}$. Hence, if the boundary of φ had at most eight or less linearly independent vertices it would have zero or infinite volume. □

The Proof of Lemma 3.10

Proof. Let $V_1, \ldots, V_9$ be a subset of linearly independent vertices in φ's boundary $\hat{\varphi}$ which exist by Lemma 3.9. According to Definition 2.17 this subset spans an eight-dimensional space. Every vertex V_k with $1 \leq k \leq 9$ is on eight boundary hyperplanes of φ defined by some linear polynomials $\ell_1, \ldots, \ell_8$ which are either in $\mathcal{E}[m,n]$ or belong to the hypercube $\mathcal{Q}_{m,n}$. In either case it is true for all $k' \in \{1, \ldots, 8\}$ that $2\ell_{k'}(V_k) = 0$. By a similar argumentation as in the proof of the Bounding-Box Property 3.8 it follows that $V_k = (p_{k,1}, \ldots, p_{k,8})$ fulfills

$$\begin{pmatrix} m_{k,1,9} \\ m_{k,2,9} \\ m_{k,3,9} \\ m_{k,4,9} \\ m_{k,5,9} \\ m_{k,6,9} \\ m_{k,7,9} \\ m_{k,8,9} \end{pmatrix} = \begin{pmatrix} m_{k,1,1} & m_{k,1,2} & m_{k,1,3} & m_{k,1,4} & m_{k,1,5} & m_{k,1,6} & m_{k,1,7} & m_{k,1,8} \\ m_{k,2,1} & m_{k,2,2} & m_{k,2,3} & m_{k,2,4} & m_{k,2,5} & m_{k,2,6} & m_{k,2,7} & m_{k,2,8} \\ m_{k,3,1} & m_{k,3,2} & m_{k,3,3} & m_{k,3,4} & m_{k,3,5} & m_{k,3,6} & m_{k,3,7} & m_{k,3,8} \\ m_{k,4,1} & m_{k,4,2} & m_{k,4,3} & m_{k,4,4} & m_{k,4,5} & m_{k,4,6} & m_{k,4,7} & m_{k,4,8} \\ m_{k,5,1} & m_{k,5,2} & m_{k,5,3} & m_{k,5,4} & m_{k,5,5} & m_{k,5,6} & m_{k,5,7} & m_{k,5,8} \\ m_{k,6,1} & m_{k,6,2} & m_{k,6,3} & m_{k,6,4} & m_{k,6,5} & m_{k,6,6} & m_{k,6,7} & m_{k,6,8} \\ m_{k,7,1} & m_{k,7,2} & m_{k,7,3} & m_{k,7,4} & m_{k,7,5} & m_{k,7,6} & m_{k,7,7} & m_{k,7,8} \\ m_{k,8,1} & m_{k,8,2} & m_{k,8,3} & m_{k,8,4} & m_{k,8,5} & m_{k,8,6} & m_{k,8,7} & m_{k,8,8} \end{pmatrix} \cdot \begin{pmatrix} p_{k,1} \\ p_{k,2} \\ p_{k,3} \\ p_{k,4} \\ p_{k,5} \\ p_{k,6} \\ p_{k,7} \\ p_{k,8} \end{pmatrix},$$

i. e., $N_k = M_k \cdot V_k$ with integer vector N_k and integer matrix M_k. Solving this system of equations gives that $p_{k,1} = \frac{u_{k,1}}{\det(M_k)}, \ldots, p_{k,8} = \frac{u_{k,8}}{\det(M_k)}$ fractions of integers $u_{k,1}$ to $u_{k,8}$ and determinant $\det(M_k)$.

Consider $V = \frac{V_1 + \ldots + V_9}{9}$, which is a point of φ. Moreover, consider any linear polynomial $\ell \in \mathcal{E}[m,n]$ and the corresponding hyperplane $h^0(\ell)$. Again 2ℓ is given by an integer vector $N = (n_1, \ldots, n_8)^T$ and some integer number n_9 and still it describes the same hyperplane $h^0(\ell) = h^0(2\ell)$. The distance z between $V = (p_1, \ldots, p_8)^T$ and $h^0(\ell)$ can be measured by

$$z = \left| \frac{p_1 n_1 + \ldots + p_8 n_8 + n_9}{\|N\|} \right|.$$

Since V is the weighted sum of V_1 to V_9 it follows

$$\begin{aligned} z &= \left| \frac{\left(\frac{u_{1,1}}{\det(M_1)} + \ldots + \frac{u_{9,1}}{\det(M_9)} \right) n_1 + \ldots + \left(\frac{u_{1,8}}{\det(M_1)} + \ldots + \frac{u_{9,8}}{\det(M_9)} \right) n_8 + n_9}{9\|N\|} \right| \\ &\leq \left| \frac{(u_{1,1} + \ldots + u_{9,1}) n_1 + \ldots + (u_{1,8} + \ldots + u_{9,8}) n_8 + \det(M_{\max}) n_9}{9 \cdot \det(M_{\max}) \cdot \|N\|} \right| \end{aligned}$$

where $\det(M_{\max})$ is the determinant of $\det(M_1)$ to $\det(M_9)$ with largest absolute value.

The numerator of the fraction is obviously integer. But since V is not situated on any hyperplane, it cannot be zero. Hence, the smallest possible value for the numerator is one.

The denominator is $9 \cdot \det(M_{\max}) \cdot \|N\|$ with

$$\|N\| = \sqrt{n_1^2 + n_2^2 + n_3^2 + n_4^2 + n_5^2 + n_6^2 + n_7^2 + n_8^2} \quad \text{and}$$
$$\det(M_{\max}) = \sum_{\phi \in \Phi(8)} \left(sgn(\phi) \prod_{k=1}^{8} m_{\max,k,\phi(k)} \right)$$

where $\Phi(8)$ contains all permutations of the numbers $\{1, \ldots, 8\}$ and $m_{\max,k,\phi(k)}$ are the entries of the matrix $M_{\max}$.

Consider the following cases: If ℓ is a boundary of $\mathcal{Q}_{m,n}$ then $\|N\| = 4$ because all but one of the values from $\{n_1, \ldots, n_8\}$ are zero and the remaining one is two. Moreover, if $M_{\max}$ contains a row $m_{\max,k,1}, \ldots, m_{\max,k,8}$ representing a linear polynomial given by the boundary of $\mathcal{Q}_{m,n}$, then $|\det(M_{\max})| \leq 289m^6n^2$ because all but one value from $\{m_{\max,k,1}, \ldots, m_{\max,k,8}\}$ are zero which implies that only one of the 40320 elements of sum in $\det(M_{\max})$ is non-zero. Both cases give a relatively low estimate. Consequently, it is sound to neglect them for the upper bound on the denominator.

For all $k \in \{1, \ldots, 9\}$ and all $k' \in \{1, \ldots, 8\}$ it is true that

$$\begin{aligned}
|n_1|, |m_{k,k',1}| &\leq 2m, \\
|n_2|, |m_{k,k',2}| &\leq 2m, \\
|n_3|, |m_{k,k',3}| &\leq 2, \\
|n_4|, |m_{k,k',4}| &\leq 2m, \\
|n_5|, |m_{k,k',5}| &\leq 2m, \\
|n_6|, |m_{k,k',6}| &\leq 2, \\
|n_7|, |m_{k,k',7}| &\leq m(2n+1) \quad \text{and} \\
|n_8|, |m_{k,k',7}| &\leq m(2n+1).
\end{aligned}$$

As a consequence, one easily obtains that

$$\begin{aligned}
\|N\| &\leq \sqrt{16m^2 + 2m^2(2n+1)^2 + 8} \\
&\leq 4m + 3mn \leq 4mn \qquad \text{for } n \geq 4 \text{ and} \\
|\det(M_{\max})| &\leq 40320 \cdot 64m^6 \cdot (2n+1)^2 \\
&\leq 40320 \cdot 289m^6n^2 \qquad \text{for } n \geq 8 \\
&\leq 1.2 \cdot 10^7 \cdot m^6n^2.
\end{aligned}$$

The product gives an upper bound on the denominator's absolute value. This implies a lower bound on the minimum distance of V to any hyperplane and thus, to the boundary of φ. Hence, every face of $\mathcal{H}_{\mathcal{Q}_{m,n}}[m,n]$ contains an 8-ball with radius

$$r \geq \frac{1}{4.32 \cdot 10^8 m^7 n^3}.$$

□

The Proof of Lemma 3.12

Proof. The face φ is the intersection $H^{s_1}(\ell_1) \cap \ldots \cap H^{s_w}(\ell_w)$ of the hyperplanes defined by the polynomials $\ell_1, \ldots, \ell_w$ and the signs $s_1, \ldots, s_w$. By Lemma 3.6 there is a point $P \in \varphi$ and a radius $r > 0$ such that all points Q with $\|P - Q\| < r$ are also in φ.

Induction over z, the number of intersecting surfaces, shows the existence of B_z, a volumetric subset of φ, solely built of points not part of any surface defined by the polynomials $q_1, \ldots, q_z$. For $z = 0$ one obtains $B_0 = \varphi$. Then, for $z > 0$ assume the existence of a valid subspace $B_{z-1} \subseteq \varphi$ for $z - 1$ surfaces. Consider the surface defined by q_z. If there is no intersection between $h^0(q_z)$ and B_{z-1} then keep $B_z = B_{z-1}$ because all contained point are even not on $h^0(q_z)$. Otherwise, if B_{z-1} is intersected by $h^0(q_z)$ then at least one of the sets $(h^+(q_z) \cap B_{z-1})$ or $(h^-(q_z) \cap B_{z-1})$ contains a smaller volumetric subspace. Set B_z to this set and obtain a volume of valid points. □

The Proof of Theorem 3.15

Proof. 1. Let $\varphi \in \mathcal{H}[m,n]$ be any face. By the Bounding-Box Property 3.8 φ is at least partially contained in the hypercube $\mathcal{Q}_{m,n}$ with lateral length $2 \cdot 10^7 m^6 n^3$. By definition it is true that $\mathcal{L}[m,n]$ samples the complete hypercube with points such that in any of the eight dimensions two points have a distance of d. Consequently, for every point $Q \in \mathcal{Q}_{m,n}$ there is a point $P \in \mathcal{L}[m,n]$ such that $\|P - Q\| < \sqrt{8}d$.

By Lemma 3.10 the face φ contains an 8-ball of radius $r \geq (4.5 \cdot 10^8 m^7 n^3)^{-1}$ which is also entirely contained in $\mathcal{Q}_{m,n}$. If Q is the center point of this ball, then a lattice point $P \in \mathcal{L}[m,n]$ within distance less than $\sqrt{8}d$ is found. Because $\sqrt{8}d < r$, this lattice point belongs to φ.

2. Let $P = \mathcal{L}[m,n](t_1, \ldots, t_8)$ be the lattice point for some $t_1, \ldots, t_8 \in \{-L, \ldots, L\}$. The following argues that $P = (d(t_1 + 0.125L^{-2}), dt_2, dt_3, dt_4, d(t_5 + 0.5), dt_6, dt_7, dt_8)^T$ does not belong to the forbidden surfaces: Firstly, it is true that

$$e_1(P) = (2t_1t_5 - 2t_2t_4 + t_1) \cdot 0.5d^2 + (t_5 + 0.5) \cdot 0.125L^{-2}d^2.$$

Hence, the value $e_1(P)$ is the sum of an integer multiple of $0.5d^2$ and $d' = (t_5 + 0.5) \cdot 0.125L^{-2}d^2$. Obviously, $d' \neq 0$, because t_5 is integer. Moreover,

$|t_5 + 0.5| < L^2$ implies that $|d'| < 0.125d^2$. Consequently, $e_1(P)$ cannot be zero because an integer multiple of $0.5d^2$ cannot compensate the much smaller amount d'.

Then, it is true that

$$\begin{aligned} e_2(P) = {} & (-2t_1t_6t_8 + 2t_2t_6t_7 + 2t_3t_4t_8 - 2t_3t_5t_7 - t_3t_7) \cdot 0.5d^3 + \\ & (2t_1t_5 - 2t_2t_4 + t_1) \cdot 0.5d^2 + ((t_5 + 0.5)d^{-1} - t_6t_8) \cdot 0.125L^{-2}d^3. \end{aligned}$$

This is the sum of an integer multiple of $0.5d^3$, an integer multiple of $0.5d^2$ and $d' = ((t_5 + 0.5)d^{-1} - t_6t_8) \cdot 0.125L^{-2}d^3$. However, d' cannot be zero, because t_6t_8 is always integer and $(t_5+0.5)d^{-1}$ never. Nevertheless, it is the case that $|d'| < 0.25d^3$ because $|(t_5 + 0.5)d^{-1} - t_6t_8| \leq 2L^2$. Consequently $e_2(P) \neq 0$ by a similar argumentation as above. The sum of integer multiples of $0.5d^3$ and $0.5d^2$ cannot compensate a number with absolute value as small as d'.

Finally, for all $(i, j) \in z(m)$ it is true that

$$e[i, j](P) = d(it_7 + jt_8) + 1.$$

In case of P on $h^0(e[i, j])$, i. e., when $e[i, j](P) = 0$, one obviously finds that

$$(it_7 + jt_8) = \frac{-1}{d} = -1.5 \cdot 10^9 m^7 n^3 + 0.5,$$

which would imply that the integer $(it_7 + jt_8)$ equals the rational number on the right side. Thus, $e[i, j](P) \neq 0$.

3. The bound is trivially obtained by the choice of $t_1, \ldots, t_8$ in $\{-L, \ldots, L\}$.

□

Chapter 4

Generic Image Matching

This chapter presents the main result of the thesis – a new image matching technique under projective transformations and nearest neighbor interpolation. The foundation of the new technique is the discretization of $\mathcal{F}_{\mathtt{p}}$ provided by the previous chapter.

Particularly, to solve the image matching problem under $\mathcal{F}_{\mathtt{p}}$ for given images A and B and distortion measure ω, algorithms can now be constructed to compute representative transformations f_1 to f_r, one for every possible image transformation of A. Determining f_1 to f_r can basically be done by selecting a set of valid points $P_1, \ldots, P_r$ from the faces $\varphi_1, \ldots, \varphi_r$ in the arrangement $\mathcal{H}[m, n]$. Using these transformations, images in the dictionary $\mathcal{D}[m](A)$ can be searched and the distortions $\omega(f_1(A)[m], B), \ldots, \omega(f_r(A)[m], B)$ can be computed to finally find an optimal solution.

Such an approach is eligible not only for projective image matching. This chapter introduces a generic image matching algorithm applicable for many subclasses of projective transformations as well, in particular for $\mathcal{F}_{\mathtt{s}}$, $\mathcal{F}_{\mathtt{sr}}$, $\mathcal{F}_{\mathtt{t}}$, $\mathcal{F}_{\ell}$ and $\mathcal{F}_{\mathtt{a}}$. To build such a general framework, the next section shows how slices in the parameter space $\mathbb{R}^8$ can be used as a way to describe parameter space partitions for the particular subclasses.

4.1 Parameter Spaces of Projective Subclasses

Section 3.1 provided a partition of $\mathbb{R}^8$ in terms of the arrangement $\mathcal{H}[m, n]$. This partition induces for any pair of points $P, P' \in \mathbb{R}^8$ whether the described projective transformations $f^{-1} = \mathcal{P}(P)$ and $f'^{-1} = \mathcal{P}(P')$ lead to the same transformation $f(A)[m] = f'(A)[m]$ of a given image A of size n.

Recall that the classes $\mathcal{F}_{\mathtt{s}}$, $\mathcal{F}_{\mathtt{r}}$, $\mathcal{F}_{\mathtt{sr}}$, $\mathcal{F}_{\mathtt{t}}$, $\mathcal{F}_{\ell}$ and $\mathcal{F}_{\mathtt{a}}$ are included in projective transformations. Consequently, for each of these classes there exists a subspace of $\mathbb{R}^8$ containing exactly the points representing the transformations in this subclass. It is clear that the property of the space partition remains the same even for subspaces of $\mathbb{R}^8$. Hence, according to Corollary 3.5, points P, P' representing two transformations from the same subclass, correspond to the same transformation of A if they belong to the same face in $\mathcal{H}[m,n]$.

The parameter subspaces for the classes $\mathcal{F}_{\mathtt{s}}$, $\mathcal{F}_{\mathtt{r}}$, $\mathcal{F}_{\mathtt{sr}}$, $\mathcal{F}_{\mathtt{t}}$, $\mathcal{F}_{\ell}$ and $\mathcal{F}_{\mathtt{a}}$ are not very complex. In all but one case they can be described as one of the slices in $\mathbb{R}^8$ introduced in the definition below:

Definition 4.1. *Let $\ell_1, \ldots, \ell_8$ be the linear polynomials*

$$\begin{array}{llllll}
\ell_1(p_1,\ldots,p_8) & = p_1 - 1, & \ell_2(p_1,\ldots,p_8) & = p_2, & \ell_3(p_1,\ldots,p_8) & = p_3, \\
\ell_4(p_1,\ldots,p_8) & = p_4 + p_2, & \ell_5(p_1,\ldots,p_8) & = p_5 - p_1, & \ell_6(p_1,\ldots,p_8) & = p_6, \\
\ell_7(p_1,\ldots,p_8) & = p_7, & & \textit{and} & \ell_8(p_1,\ldots,p_8) & = p_8.
\end{array}$$

Based on the polynomials let

$$\begin{aligned}
\chi_{\mathtt{a}} &= h^0(\ell_7) \cap h^0(\ell_8), \\
\chi_{\ell} &= h^0(\ell_3) \cap h^0(\ell_6) \cap h^0(\ell_7) \cap h^0(\ell_8), \\
\chi_{\mathtt{t}} &= h^0(\ell_1) \cap h^0(\ell_2) \cap h^0(\ell_4) \cap h^0(\ell_5) \cap h^0(\ell_7) \cap h^0(\ell_8), \\
\chi_{\mathtt{sr}} &= h^0(\ell_3) \cap h^0(\ell_4) \cap h^0(\ell_5) \cap h^0(\ell_6) \cap h^0(\ell_7) \cap h^0(\ell_8) \qquad \textit{and} \\
\chi_{\mathtt{s}} &= h^0(\ell_2) \cap h^0(\ell_3) \cap h^0(\ell_4) \cap h^0(\ell_5) \cap h^0(\ell_6) \cap h^0(\ell_7) \cap h^0(\ell_8)
\end{aligned}$$

be slices in $\mathbb{R}^8$.

Each slice consists of points P in the parameter space $\mathbb{R}^8$ representing inverse projective transformations $f^{-1} = \mathcal{P}(P)$. But as a slice is only a subspace of $\mathbb{R}^8$ it follows that the subsumption of all such points define a certain subclass of projective transformations. The following lemma connects the slices with the respective transformation classes:

Lemma 4.2 (Proof in Section 4.5)**.**

$$\begin{array}{ll}
\mathcal{F}_{\mathtt{a}} & = \{f \mid \exists P \in \chi_{\mathtt{a}} \textit{ such that } f^{-1} = \mathcal{P}(P)\}, \\
\mathcal{F}_{\ell} & = \{f \mid \exists P \in \chi_{\ell} \textit{ such that } f^{-1} = \mathcal{P}(P)\}, \\
\mathcal{F}_{\mathtt{t}} & = \{f \mid \exists P \in \chi_{\mathtt{t}} \textit{ such that } f^{-1} = \mathcal{P}(P)\}, \\
\mathcal{F}_{\mathtt{sr}} & = \{f \mid \exists P \in \chi_{\mathtt{sr}} \textit{ such that } f^{-1} = \mathcal{P}(P)\}, \\
\mathcal{F}_{\mathtt{s}} & = \{f \mid \exists P \in \chi_{\mathtt{s}} \textit{ such that } f^{-1} = \mathcal{P}(P)\}.
\end{array}$$

Hence, there is a close connection between natural subclasses of projective transformations and slices in $\mathbb{R}^8$. To make this concept more general consider the following definition:

Definition 4.3. *If χ is a slice in $\mathbb{R}^8$ then*

$$\mathcal{F}_{\chi} = \left\{ f \in \mathcal{F}_{\mathtt{p}} \;\middle|\; \exists P \in \chi \textit{ such that } f^{-1} = \mathcal{P}(P) \right\},$$

contains all projective transformations corresponding to a point in χ.

Hence, $\mathcal{F}_{\mathtt{a}} = \mathcal{F}_{\chi_{\mathtt{a}}}$, $\mathcal{F}_{\ell} = \mathcal{F}_{\chi_{\ell}}$ and so forth. Notice that $\mathcal{F}_{\chi}$ is not a transformation class for every slice χ in $\mathbb{R}^8$, e. g., if $\mathcal{F}_{\chi}$ is not closed under taking inverses.

One exception that does not fit into this framework are rotations. Although the parameter points in $\mathbb{R}^8$ that represent rotations form a subspace, it is not possible to describe them simply by a slice. The reason for this is a non-linear subspace shape. In particular, it is true for the quadratical polynomial

$$\ell(p_1, \ldots, p_8) = p_1^2 + p_2^2 - 1$$

that the parameter subspace of rotations can be described by

$$\lambda = h^0(\ell) \cap \chi_{\mathtt{sr}}.$$

Naturally, as rotations are a subset of $\mathcal{F}_{\mathtt{sr}}$ their parameter subspace is a one-dimensional subset of the two-dimensional plane $\chi_{\mathtt{sr}}$, i. e., a circle of unit radius.

Although the parameter space cannot be described as a slice, most of the results in this chapter work for λ or can be modified to fit. In [25, 26] the author, Maciej Liśkiewicz and Ragnar Nevries describe the exact proceeding to enable the generic algorithm of Section 4.3 to work for image matching under rotations. However, in the given framework the case of rotations is omitted for the generality of presentation. Only the conclusion of this chapter gives some statements about the complexity of image matching under $\mathcal{F}_{\mathtt{r}}$.

4.2 A First Polynomial Time Approach

This section develops the *polynomial time image matching algorithm* for subsets of projective transformations. This provides the first known polynomial time algorithm for projective, affine, and linear transformations. The correctness of the presented approach bases on the following theorem which formulates the announced connection between $\mathcal{H}[m, n]$ and $\mathcal{D}[m](A)$. In fact, the mentioned correspondence between the two sets can be expressed for many subclasses of projective transformations:

Theorem 4.4. *Let $m, n \in \mathbb{N}$ be natural numbers and A be an image of size n. Moreover, let χ be any slice in $\mathbb{R}^8$ that describes a subset (not necessarily class) $\mathcal{F}_{\chi}$ of projective transformations. Then there is a surjective mapping*

$$\Gamma_{\chi} : \mathcal{H}_{\chi}[m, n] \to \mathcal{D}[\mathcal{F}_{\chi}, m](A).$$

Proof. If $\mathcal{D}[\mathcal{F}_{\chi}, m](A)$ is empty, then Γ_{χ} is undefined for all arguments. Otherwise let A_0 be any image contained in $\mathcal{D}[\mathcal{F}_{\chi}, m](A)$. Now consider for all faces $\varphi \in \mathcal{H}_{\chi}[m, n]$ a valid point $P_{\varphi} \in \varphi$ if such a point exists. It is straightforward that $P_{\varphi} \in \chi$ since $\varphi \subseteq \chi$. Moreover, by definition it is true that $(\mathcal{P}(P_{\varphi}))^{-1} \in \mathcal{F}_{\chi}$. Then let $f_{\varphi}^{-1} = \mathcal{P}(P_{\varphi})$ and f_{φ} be the projective transformation with inverse f_{φ}^{-1}.

For all images A of size n let Γ_χ be the following mapping:

$$\forall \varphi \in \mathcal{H}_\chi[m,n] : \ \Gamma_\chi(\varphi) = \begin{cases} f_\varphi(A)[m] & : \text{ if } P_\varphi \text{ exists,} \\ A_0 & : \text{ otherwise.} \end{cases}$$

It remains to show that such a Γ_χ is surjective. For that assume the contrary. Then there is a transformation $f \in \mathcal{F}_\chi$ such that no face φ gives $\Gamma_\chi(\varphi) = f(A)[m]$. However, f^{-1} is a projective transformation represented by a point P in χ and by Lemma 2.21 and by definition there must be a face $\varphi \in \mathcal{H}_\chi[m,n]$ containing that point. Thus, the face φ contains both P and P_φ, which implies that the points are not separated by any hyperplane given by the polynomials in $\mathcal{E}[m,n]$.

Under the given assumption that $f(A)[m] \neq f_\varphi(A)[m]$ there exists $(i,j) \in z(m)$ with different colors $f(A)[m](i,j)$ and $f_\varphi(A)[m](i,j)$.

- Firstly, if neither the point $f^{-1}(i,j)$ nor the point $f_\varphi^{-1}(i,j)$ hit a pixel in $Pix[n]$, then both $f(A)[m](i,j) = 0$ and $f_\varphi(A)[m](i,j) = 0$. Consequently, at least one of the points must hit a pixel.

- Secondly, if $f^{-1}(i,j) \in pix(i',j')$ with $pix(i',j') \in Pix[n]$ consider the four polynomials
$$\ell^{\mathtt{v}}_{iji'} \quad \ell^{\mathtt{v}}_{ij(i'+1)} \quad \ell^{\mathtt{h}}_{ijj'} \quad \text{and} \quad \ell^{\mathtt{h}}_{ij(j'+1)}.$$
By Lemma 3.4 it must be the case that P is either in the space $H^+(\ell^{\mathtt{v}}_{iji'}) \cap H^-(\ell^{\mathtt{v}}_{ij(i'+1)})$ or in $H^+(-\ell^{\mathtt{v}}_{iji'}) \cap H^-(-\ell^{\mathtt{v}}_{ij(i'+1)})$. Since P and P_φ are not separated by hyperplanes it follows that P_φ has to be in one of these subspaces, too. But then Lemma 3.4 implies that $f_\varphi^{-1}(i,j) \in \mathtt{v}^+_{i'} \cap \mathtt{v}^-_{i'+1}$.

  Analogously, since P is in $H^+(\ell^{\mathtt{h}}_{ijj'}) \cap H^-(\ell^{\mathtt{h}}_{ij(j'+1)})$ or in $H^+(-\ell^{\mathtt{h}}_{ijj'}) \cap H^-(-\ell^{\mathtt{h}}_{ij(j'+1)})$ it must be that P_φ is in these, too. Again Lemma 3.4 implies that $f_\varphi^{-1}(i,j) \in \mathtt{h}^+_{j'} \cap \mathtt{h}^-_{j'+1}$. Hence, $f_\varphi^{-1}(i,j) \in pix(i',j')$ which means that both $f^{-1}(i,j)$ and $f_\varphi^{-1}(i,j)$ fall into the same pixel of $Pix[n]$. Thus, again $f(A)[m](i,j) = f_\varphi(A)[m](i,j)$.

- Finally, the case when $f_\varphi^{-1}(i,j)$ hits a pixel $pix(i',j')$ in $Pix[n]$ works analogously.

Thus, the assumption fails and Γ_χ has to be surjective. □

The theorem is applicable to a variety of subsets of projective transformations. In particular, it can be applied for nearly all cases considered in Section 2.3:

Corollary 4.5 (Without proof). *Let $m, n \in \mathbb{N}$ be natural numbers and A be an image of size n. Then there is a surjective mapping*

$$\Gamma : \mathcal{H}[m,n] \to \mathcal{D}[m](A)$$

for projective transformations. Moreover, for all $\chi \in \{\chi_{\mathtt{a}}, \chi_{\ell}, \chi_{\mathtt{t}}, \chi_{\mathtt{sr}}, \chi_{\mathtt{s}}\}$ *there is a surjective mapping*

$$\Gamma_\chi : \mathcal{H}_\chi[m, n] \to \mathcal{D}[\mathcal{F}_\chi, m](A)$$

for the subclass $\mathcal{F}_\chi$ *of projective transformations.*

The central result presented in Theorem 4.4 can be applied as a toolkit for an image matching algorithm under subsets of projective transformations. For example, assume that images A and B of size n respectively m are given. Then Theorem 4.4 enables a very convenient way to compute the whole dictionary $\mathcal{D}[\mathcal{F}_\chi, m](A)$. Instead of trying every transformation in $\mathcal{F}_\chi$, which is impossible, an algorithm can simply enumerate all faces $\varphi \in \mathcal{H}_\chi[m, n]$ to obtain $A' = \Gamma_\chi(\varphi)$. Following this procedure, the algorithm yields the whole set $\mathcal{D}[\mathcal{F}_\chi, m](A)$. Finally, holding $\mathcal{D}[\mathcal{F}_\chi, m](A)$ enables image matching simply by computing $\delta(A', B)$ for every image A' in $\mathcal{D}[\mathcal{F}_\chi, m](A)$. This generic approach to the image matching Problem 2.13 is provided by the abstract algorithm in Figure 4.1, which exploits the surjectivity of mapping Γ_χ. The problems with the generic approach are the

Algorithm `Abstract Generic Image Matching`
Input: Image A of size n and image B of size m.
Parameter: A slice χ in $\mathbb{R}^8$, a distortion measure $\omega : \mathcal{C} \times \mathcal{C} \to \mathbb{N}$.
Output: $P \in \chi$ with $f^{-1} = \mathcal{P}[\mathcal{F}_{\mathtt{p}}](P)$ such that $\delta(f(A)[m], B)$ is the minimum over all $f \in \mathcal{F}_\chi$.

```
 1. Initialize $P$ and $\delta = \infty$;
 2. for all faces $\varphi \in \mathcal{H}_\chi[m, n]$ do
 3.     if $\varphi$ contains a valid point then
 4.         compute a valid point $P_\varphi \in \varphi$;
 5.         get the inverse projective transformation $f_\varphi^{-1} = \mathcal{P}(P_\varphi)$;
 6.         obtain the transformed image $A' = f_\varphi(A)[m]$ by $f_\varphi^{-1}$;
 7.         if $\delta > \delta(A', B)$ then update $P = P_\varphi$ and $\delta = \delta(A', B)$;
 8.     end if
 9. end for
10. return parameter $P$;
```

Figure 4.1: An abstract generic image matching algorithm.

unhandled algorithmic challenges to compute exactly all faces φ in $\mathcal{H}_\chi[m, n]$ and then to obtain valid points P_φ which are in φ. The next section shows how the faces and its representative points can be found algorithmically. For an example Figure 3.2 presents a set of points eligible for the subspace $\chi_{\mathtt{sr}}$ of $\mathbb{R}^8$ that contains all combinations of scaling and rotation.

For the case of projective transformations there is, however, a very convenient way to avoid the challenge of computing faces and contained valid points. Particularly, by the existence of the lattice $\mathcal{L}[m, n]$ it is not necessary to deal with faces in such an exact fashion. It is also possible to guarantee the processing of at least one valid point from every face by sampling the parameter space $\mathbb{R}^8$ with the points in $\mathcal{L}[m,n]$. See Figure 3.2 for an impression of the lattice $\mathcal{L}[m,n]$. Then, according to Lattice Property 3.15, the abstract algorithm can be modified to a concrete projective image matching algorithm in the fashion presented in Figure 4.2. The new algorithm is a polynomial time approach to projective

Algorithm `Exhaustive Search Projective Image Matching`
Input: Image A of size n and image B of size m.
Parameter: A distortion measure $\omega : \mathcal{C} \times \mathcal{C} \to \mathbb{N}$.
Output: $P \in \mathbb{R}^8$ with $f^{-1} = \mathcal{P}(P)$ such that $\delta(f(A)[m], B)$ is the minimum over all $f \in \mathcal{F}_{\mathtt{p}}$.

```
1. Initialize $P$ and $\delta = \infty$;
2. for all integer combinations $t = (t_1, \ldots, t_8)^T \in \{-L, \ldots, L\}^8$ do
3.     compute $P_t = d \cdot (t_1 + 0.125L^{-2}, t_2, t_3, t_4, t_5 + 0.5, t_6, t_7, t_8)^T$;
4.     get the inverse projective transformation $f^{-1} = \mathcal{P}(P_t)$;
5.     obtain the transformed image $A' = f(A)[m]$ by $f^{-1}$;
6.     if $\delta > \delta(A', B)$ then update $P = P_t$ and $\delta = \delta(A', B)$;
7. end for
8. return parameter $P$;
```

Figure 4.2: A projective image matching algorithm searching exhaustively the set $\mathcal{L}[m,n]$. Here $L = 3 \cdot 10^{16} m^{13} n^6$ and $d = (1.5 \cdot 10^9 m^7 n^3 + 0.5)^{-1}$ are defined as in Section 3.2.

image matching. Note, however, that the algorithm works for projective transformations and it is not obvious how it can be modified e. g. for affine or linear transformations.

Theorem 4.6. *The image matching Problem 2.13 under projective transformations can be solved in polynomial time.*

Proof. Problem 2.13 under projective transformations can be solved by the use of the algorithm in Figure 4.2. Its correctness follows trivially from the Lattice Property 3.15 and Corollary 4.5. It remains to show that it works in polynomial time with respect to n and m, the sizes of given images A and B.

By definition the **for**-loop iterates $(2L+1)^8$ times which is roughly $O(m^{104} n^{48})$. The time complexity of the steps within the loop is dominated by the computation of A' and $\delta(A', B)$ and thus, it is $O(m^2) \cdot T$, where T is the time needed for the computation of the function $\omega : \mathcal{C} \times \mathcal{C} \to \mathbb{R}$. However, since $\mathcal{C}$ is a finite set T is a constant. Consequently, the time complexity of computing projective image matching is in $O(m^{106} n^{48})$ which is polynomial in m and n. $\square$

Since the time complexity of the algorithm in Figure 4.2 is very high, the next section will return to the abstract generic approach to image matching. Instead of avoiding the algorithmic challenges, the next algorithm is equipped with a sophisticated data structure for $\mathcal{H}_\chi[m,n]$ that makes it possible to rigorously improve the running time.

4.3 A Generic Algorithm

This section presents a new generic polynomial time algorithm for image matching under specific subsets of projective transformations. For the class of projective transformations themselves the algorithm runs in $O(m^{16}n^8)$ time. This massive acceleration with respect to the given $O(m^{106}n^{48})$ time algorithm of the previous section can be achieved by the help of a data structure for $\mathcal{H}_\chi[m,n]$ that is built in a preprocessing step. After this preprocessing, the algorithm uses the data structure to simply enumerate all faces in the parameter space. For every face φ obtained by this procedure the algorithm computes the image $A' = \Gamma_\chi(\varphi)$ and compares it to B by the given distance measure ω. This approach corresponds roughly to the abstract algorithm given in Figure 4.1 of Section 4.2.

4.3.1 Data Structure

Before going into the details, this section presents the data structure for $\mathcal{H}_\chi[m,n]$. Research in combinatorial geometry focuses on the arrangement $\mathcal{A}_\chi[\mathcal{E}[m,n]]$ rather then on $\mathcal{H}_\chi[m,n]$ and thus, provides an incidence graph $\mathcal{G}_\chi[\mathcal{E}[m,n]]$ only for $\mathcal{A}_\chi[\mathcal{E}[m,n]]$. For a short notation the set $\mathcal{A}_\chi[\mathcal{E}[m,n]]$ and the graph $\mathcal{G}_\chi[\mathcal{E}[m,n]]$ are subsequently denoted by $\mathcal{A}_\chi[m,n]$ and $\mathcal{G}_\chi[m,n]$, respectively.

The following develops a notion that makes it possible to apply the incidence graph $\mathcal{G}_\chi[m,n]$ of $\mathcal{A}_\chi[m,n]$ to use it as a data structure for $\mathcal{H}_\chi[m,n]$. Because the nodes of $\mathcal{G}_\chi[m,n]$ correspond to faces in $\mathcal{A}_\chi[m,n]$, its application for $\mathcal{H}_\chi[m,n]$ needs a translation mechanism between the elements of the two arrangements. It is comprehensible that the translation needs a certain overhead and thus, implies a loss in efficiency. On the other hand, in this way it is not necessary to develop a new data structure and the image matching algorithm can access well understood subroutines for $\mathcal{G}_\chi[m,n]$.

The following introduces a *translation* method, denoted here as τ, that converts faces from $\mathcal{A}_\chi[m,n]$ into faces of $\mathcal{H}_\chi[m,n]$. The basic idea is to make sure that the geometric properties of a given face $\varphi \in \mathcal{A}_\chi[m,n]$ are kept as well as possible by $\tau(\varphi)$. Hence, to roughly preserve the location of φ the translation τ selects the face $\tau(\varphi)$ in $\mathcal{H}_\chi[m,n]$ that completely contains φ:

Definition 4.7. *Let $m, n \in \mathbb{N}$ be natural numbers and let $\mathcal{E}[m,n] = \{\ell_1, \ldots, \ell_w\}$ be the set of linear polynomials of Definition 3.3. Moreover, let χ be any slice* *in $\mathbb{R}^8$. Then $\tau_\chi : \mathcal{A}_\chi[m,n] \to \mathcal{H}_\chi[m,n]$ is the following mapping between the respective arrangements: For every face*

$$\varphi = \chi \cap \left(\bigcap_{k=1}^{w} h^{s_k}(\ell_k) \right) \neq \emptyset,$$

which is given by a specific choice of signs $s_1, \ldots, s_w \in \{+, 0, -\}$, let

$$\tau_\chi(\varphi) = \chi \cap \left(\bigcap_{k=1}^{w} H^{s'_k}(\ell_k) \right)$$

where for all $k \in \{1, \ldots, w\}$ the new sign $s'_k \in \{+, -\}$ is defined as

$$s'_k = \begin{cases} + & : \text{ if } s_k = +, \\ + & : \text{ if } s_k = 0, \\ - & : \text{ if } s_k = -. \end{cases}$$

For $\chi = \mathbb{R}^8$, let for convenience, τ be an abbreviation for $\tau_{\mathbb{R}^8}$.

For an example for the translation of faces in $\mathcal{A}_\chi[m,n]$ into faces of $\mathcal{H}_\chi[m,n]$ see Figure 2.4 (a) and the corresponding description below the figure. Particularly, in the given example it would be the case that e. g. $\tau_\chi(\varphi_1) = \varphi'_1$ and $\tau_\chi(\varphi_{14}) = \varphi'_4$.

The translation τ_χ has the following useful property:

Lemma 4.8 (Proof in Section 4.5). *Let $m, n \in \mathbb{N}$ be natural numbers and χ be any slice in $\mathbb{R}^8$. Then τ_χ is a surjective mapping such that for all faces $\varphi \in \mathcal{A}_\chi[m,n]$ it is true $\varphi \subseteq \tau_\chi(\varphi)$.*

Via the translation method, $\mathcal{G}_\chi[m,n]$ can be used as a data structure for $\mathcal{H}_\chi[m,n]$. By traversing all nodes of $\mathcal{G}_\chi[m,n]$ one enumerates all faces in $\mathcal{A}_\chi[m,n]$. This provides one face $\varphi' = \tau_\chi(\varphi)$ of $\mathcal{H}_\chi[m,n]$ for every encountered face φ. Since τ_χ is surjective this process enumerates all faces of $\mathcal{H}_\chi[m,n]$, too. Consequently, for given image A of size n and image B of size m the approach of the previous section can be reformulated as in Figure 4.3. Although the new algorithm introduces a particular way of enumerating the faces in $\mathcal{H}_\chi[m,n]$ it remains abstract because it still does not solve the challenge of finding valid points.

However, first impressions on time complexity can be found. In fact, the time needed to build the graph $\mathcal{G}_\chi[m,n]$ and also the corresponding space consumption depend on the cardinality of $\mathcal{E}[m,n]$ and the dimension k of slice χ. Building on Edelsbrunner et al. [15, 16] Section 2.5 shows that $\mathcal{G}_\chi[m,n]$ can be computed in $O(|\mathcal{E}[m,n]|^k)$ time. Hence, the following lemma gives an upper bound on $|\mathcal{A}_\chi[m,n]|$.

Lemma 4.9 (Proof in Section 4.5). *Let $m, n \in \mathbb{N}$ be natural numbers and χ be any k-slice in $\mathbb{R}^8$ with $0 \leq k \leq 8$. The cardinality of $\mathcal{A}_\chi[m,n]$ grows by $O(m^{2k}n^k)$. This implies that $|\mathcal{H}_\chi[m,n]|$ is in $O(m^{2k}n^k)$, too.*

Algorithm `Abstract Generic Depth First Search Image Matching`
Input: Image A of size n and image B of size m.
Parameter: A slice χ in $\mathbb{R}^8$, a distortion measure $\omega : \mathcal{C} \times \mathcal{C} \to \mathbb{N}$.
Output: $P \in \chi$ with $f^{-1} = \mathcal{P}[\mathcal{F}_{\mathtt{p}}]\,(P)$ such that $\delta(f(A)[m], B)$ is the minimum over all $f \in \mathcal{F}_\chi$.

```
1.  Initialize $P$ and $\delta = \infty$;
2.  Compute $\mathcal{G}_\chi[m,n]$ and traverse it with undirected depth first search;
3.  **for** every encountered node $v$ **do**
4.      determine $\varphi' = \tau_\chi(\varphi(v))$;
5.      **if** $\varphi'$ contains a valid point **then**
6.          compute a valid point $P_{\varphi'} \in \varphi'$;
7.          get the inverse projective transformation $f_{\varphi'}^{-1} = \mathcal{P}\,(P_{\varphi'})$;
8.          obtain the transformed image $A' = f_{\varphi'}(A)[m]$ by $f_{\varphi'}^{-1}$;
9.          **if** $\delta > \delta(A', B)$ **then** update $P = P_\varphi$ and $\delta = \delta(A', B)$;
10.     **end if**
11. **end for**
12. **return** parameter $P$;
```

Figure 4.3: An abstract generic image matching algorithm traversing all faces in $\mathcal{H}_\chi[m,n]$ by depth first searching the incidence graph $\mathcal{G}_\chi[m,n]$ of $\mathcal{A}_\chi[m,n]$.

Consequently, the time complexity of the algorithm in Figure 4.3 is $O(m^{2k}n^k)$ times $O(\max\{m^2, T\})$, where $O(m^2)$ describes the cost of computing $A' = \Gamma_\chi(\varphi)$ and $\delta(A', B)$ in the **for**-loop and $O(T)$ the maximum time to find a valid point in a visited face.

This means that, provided that the computation of valid points can be done fast, the use of the incidence graph implies a crucial speed up. But there is still a bit potential left to decrease time complexity. In fact, the rest of this section shows how to omit the $O(m^2)$ time factor needed to compute $A' = \Gamma_\chi(\varphi)$ and $\delta(A', B)$.

On the other hand, it is left as an open problem to find a *generic* approach of determining valid points of visited faces. Consequently, it is necessary to give a specific technique for every special case of given slice χ. Section 4.4 introduces constant time methods of computing valid points for all considered subclasses of projective transformations. Hence, for $\mathcal{F}_{\mathtt{s}}$, $\mathcal{F}_{\mathtt{sr}}$, $\mathcal{F}_{\mathtt{t}}$, $\mathcal{F}_\ell$, $\mathcal{F}_{\mathtt{a}}$ and $\mathcal{F}_{\mathtt{p}}$ image matching is shown to be solvable in $O(m^{2k}n^k)$ time where k is the dimension of the respective slice $\chi_{\mathtt{s}}, \chi_{\mathtt{sr}}, \chi_{\mathtt{t}}, \chi_\ell, \chi_{\mathtt{a}}$ or $\mathbb{R}^8$, respectively.

4.3.2 Incremental Enumeration of $\mathcal{D}[\mathcal{F}_\chi, m](A)$

So far the algorithm in Figure 4.3 computes $A' = \Gamma(\varphi)$ from scratch in every loop iteration. In this way, a lot of time is wasted because every A' can also be computed by very few update operations. Before giving the structural property needed for this speed-up, consider another connection between $\mathcal{H}_\chi[m,n]$ and $\mathcal{A}_\chi[m,n]$:

Theorem 4.10 (Proof in Section 4.5)**.** *Let $m, n \in \mathbb{N}$ be natural numbers and consider $\mathcal{E}[m,n] = \{\ell_1, \dots, \ell_w\}$, the set of linear polynomials of Definition 3.3. Moreover, let χ be any slice in $\mathbb{R}^8$. Take two incident faces*

$$\varphi_1 = \chi \cap \left(\bigcap_{k=1}^{w} h^{s_k}(\ell_k) \right) \neq \emptyset \qquad \textit{and} \qquad \varphi_2 = \chi \cap \left(\bigcap_{k=1}^{w} h^{S_k}(\ell_k) \right) \neq \emptyset$$

of $\mathcal{A}_\chi[m,n]$, which are determined by the specific choice of signs $s_1, \dots, s_w, S_1, \dots, S_w \in \{+, 0, -\}$.

Respectively, consider the faces

$$\varphi_1' = \tau_\chi(\varphi_1) \qquad \textit{and} \qquad \varphi_2' = \tau_\chi(\varphi_2)$$

of $\mathcal{H}_\chi[m,n]$.

If there are valid points $P_1 \in \varphi_1'$ and $P_2 \in \varphi_2'$, then it is true for $f_1^{-1} = \mathcal{P}(P_1)$ and $f_2^{-1} = \mathcal{P}(P_2)$ and for all $(i,j) \in z(m)$: If $f_1^{-1}(i,j)$ and $f_2^{-1}(i,j)$ fall into different pixels of $Pix[n]$ then there is an index $q \in \{1, \dots, w\}$ with $s_q \neq S_q$ and $\ell_q = \ell^{\mathtt{v}}_{ijk'}$ or $\ell_q = \ell^{\mathtt{h}}_{ijk'}$ for some $k' \in \{-n, \dots, n+1\}$.

The value of Theorem 4.10 depends heavily on the existence of valid points P_1 and P_2. Although there are slices for which their existence cannot be guaranteed, the Valid-Points Property 3.13 states that for $\chi = \mathbb{R}^8$ every face of $\mathcal{H}_\chi[m,n]$ contains a valid point. Section 4.4 shows that for the considered subclasses of projective transformations valid points exist for all faces in the restricted arrangements $\mathcal{H}_\chi[m,n]$.

Thus, the theorem states that two faces φ_1 and φ_2, which are adjacent in the parameter space, give $\varphi_1' = \tau_\chi(\varphi_1)$ and $\varphi_2' = \tau_\chi(\varphi_2)$ that correspond to similar images $A_1 = \Gamma_\chi(\varphi_1')$ and $A_2 = \Gamma_\chi(\varphi_2')$. Particularly, if φ_1 and φ_2 are incident, then A_1 and A_2 differ at an index $(i,j) \in z(m)$ only if the separating hyperplane is given by a polynomial $\ell^{\star}_{ijk'}$ in $\mathcal{E}[m,n]$ defined by the index (i,j) and for an arbitrary $\star \in \{\mathtt{v}, \mathtt{h}\}$ and arbitrary $k' \in \{-n, \dots, n+1\}$.

This means, e. g., if two nodes u and v of $\mathcal{G}_\chi[m,n]$ are connected by an edge uv then they represent two incident faces $\varphi(u)$ and $\varphi(v)$. In that case it may happen that the undirected depth first search on $\mathcal{G}_\chi[m,n]$ has just finished processing of the node u corresponding to a face $\varphi_1' = \tau_\chi(\varphi(u))$ and thus, the algorithm has the image $A_1 = \Gamma_\chi(\varphi_1')$ in memory. Traversing towards node v leads to a face $\varphi_2' = \tau_\chi(\varphi(v))$ which is either the same $\varphi_1' = \varphi_2'$ by Lemma 4.8 or an adjacent face of φ_1'. In the second case, Theorem 4.10 states that A_1 and $A_2 = \Gamma_\chi(\varphi_2')$ differ at most at indices (i,j) given by hyperplanes separating $\varphi(u)$ and $\varphi(v)$. Hence, it is sufficient to update A_1 on such indices to get A_2. Altogether, this bears the possibility to save $O(m^2)$ time consumption.

Recall the sets $\ell(u)$ and $\ell(v)$ introduced in Definition 2.23. To find the indices (i, j) to be updated, it is sufficient to trace the polynomials $\ell(u)$ and $\ell(v)$. In particular, the following defines *Update*(uv) labels for edges uv containing all information needed to compute A_2 from A_1:

Definition 4.11. *Let $m, n \in \mathbb{N}$ be natural numbers and $\mathcal{E}[m, n]$ be the set of linear polynomials in Definition 3.3. Moreover, let χ be any slice in $\mathbb{R}^8$, let $\mathcal{G}_\chi[m, n] = (V, E)$ be the incidence graph of the arrangement $\mathcal{A}_\chi[m, n]$, let $uv \in E$ be any edge with $\varphi(u)$ and $\varphi(v)$ being two incident faces of $\mathcal{A}_\chi[m, n]$ and let $\varphi_1 = \tau_\chi(\varphi(u))$ and $\varphi_2 = \tau_\chi(\varphi(u))$ be the corresponding (not necessarily different) faces of $\mathcal{H}_\chi[m, n]$.*

If $P_1 \in \varphi_1$ and $P_2 \in \varphi_2$ are valid points then

$$\mathit{Update}(uv) = \left\{(i, j, i', j') \;\middle|\; \ell^\star_{ijk} \in \ell(u) \cup \ell(v) \text{ such that } [f_2^{-1}(i, j)] = (i', j')\right\}$$

and

$$\mathit{Update}(vu) = \left\{(i, j, i', j') \;\middle|\; \ell^\star_{ijk} \in \ell(u) \cup \ell(v) \text{ such that } [f_1^{-1}(i, j)] = (i', j')\right\},$$

are Update sets where $\star \in \{\mathtt{v}, \mathtt{h}\}$ and $f_1^{-1} = \mathcal{P}(P_1)$ and $f_2^{-1} = \mathcal{P}(P_2)$.

Again, the definition of *Update*(uv) and *Update*(vu) works only if φ_1 and φ_2 contain valid points P_1 and P_2. However, the functionality of *Update*(uv) is simple: If $A_1 = \Gamma_\chi(\tau_\chi(\varphi(u)))$ is in memory and the algorithm traverses to v, then *Update*(uv) consists of tuples (i, j, i', j') to tell the algorithm to update the color at index (i, j) by setting its value to $A(i', j')$. The following lemma states the correctness of Definition 4.11, i. e., that processing all tuples in *Update*(uv) changes A_1 to the $A_2 = \Gamma_\chi(\tau_\chi(\varphi(v)))$:

Lemma 4.12 (Proof in Section 4.5)**.** *Let $m, n \in \mathbb{N}$ be natural numbers, $\mathcal{E}[m, n] = \{\ell_1, \dots, \ell_w\}$ be the set of linear polynomials in Definition 3.3, let χ be any slice in $\mathbb{R}^8$ and $\mathcal{G}_\chi[m, n] = (V, E)$ be the incidence graph of the arrangement $\mathcal{A}_\chi[m, n]$. Moreover, let $uv \in E$ be an edge that connects the nodes u and v of the incident faces*

$$\varphi(u) = \chi \cap \left(\bigcap_{k=1}^{w} h^{s_k}(\ell_k)\right) \qquad \text{and} \qquad \varphi(v) = \chi \cap \left(\bigcap_{k=1}^{w} h^{S_k}(\ell_k)\right),$$

which are determined by the specific choice of signs $s_1, \dots, s_w$, $S_1, \dots, S_w \in \{+, 0, -\}$.

If there is a valid point $P \in \tau_\chi(\varphi(v))$ and if the signs $s_q \neq S_q$ differ for the polynomial $\ell_q = \ell^\star_{ijk'}$, then $(i, j, i', j') \in \mathit{Update}(uv)$ with $(i', j') = [\mathcal{P}(P)(i, j)]$.

Hence, the set *Update*(uv) covers at least all indices (i, j, i', j') which belong to separating hyperplanes. According to Theorem 4.10 it follows that the complete update can be performed by considering such indices. It happens for several reasons that *Update*(uv) identifies (i, j, i', j') to be updated although $A_1(i, j) = A_2(i, j)$. However, although an update can be superfluous, it cannot cause an error.

4.3.3 Preprocessing

The preprocessing step computes $\mathcal{G}_\chi[m,n] = (V,E)$, the incidence graph of $\mathcal{A}_\chi[m,n]$, to serve as a data structure for $\mathcal{H}_\chi[m,n]$. According to Section 2.5 it is known from Theorem 7.6 in [15] that the graph can be constructed in $O(m^{2k}n^k)$ time for k-slices χ since $\mathcal{E}[m,n]$ contains $2(2m+1)^2(2n+1) = O(m^2n)$ polynomials.

Beside computing $\mathcal{G}_\chi[m,n]$, the preprocessing is also used to label the nodes and edges of the graph with additional information needed for the matching process. Particularly, according to [15, 16] the computation of $\mathcal{G}_\chi[m,n]$ provides already a label $P(v)$ for every node v. Each label $P(v)$ stands for a point in $\mathbb{R}^8$ which is contained in $\varphi(v)$. But this point is not necessarily valid. Consequently, an additional preprocessing step has to compute *valid points* $P'(v)$ for all nodes of the incidence graph.

Definition 4.13. *Let $m,n \in \mathbb{N}$ be natural numbers, let χ be any k-dimensional slice in $\mathbb{R}^8$ and consider $\mathcal{G}_\chi[m,n] = (V,E)$, the incidence graph of $\mathcal{A}_\chi[m,n]$. Then $P' : V \to \mathbb{R}^8$ is a node label assigning every node $v \in V$ a valid point $P'(v) \in \tau_\chi(\varphi(v))$, if such a point exists.*

As stated above, a generic approach to this challenge has not been found yet, and thus, this particular step of preprocessing has to be solved individually for every given slice χ. Section 4.4 provides methods to obtain valid points $P'(v)$ for every node v when χ is one of the slices $\chi_{\mathtt{s}}, \chi_{\mathtt{sr}}, \chi_{\mathtt{t}}, \chi_\ell, \chi_{\mathtt{a}}$ or $\mathbb{R}^8$ representing subclasses of projective transformation.

Every edge $uv \in E$ should be labeled by *Update*(uv), the set that contains all information on which indices are to be updated when traversing from face $\varphi(u)$ to face $\varphi(v)$. The difficulty of computing *Update*(uv) for all edge $uv \in E$ is that it probably consumes too much time. Although the node degree in $\mathcal{G}_\chi[m,n]$ is constant on average, there may be nodes with an excessive amount of neighbors. Especially for such nodes v the incident edges uv have large *Update*(uv) sets. Whether it is possible to compute *Update*(uv) for all edges uv in $O(|\mathcal{E}[m,n]|^k)$ time if χ is a k-slice remains open.

However, the use of $\mathcal{G}_\chi[m,n]$ is to perform depth first search in the set of faces. The following shows that it is enough to preprocess *Update*(uv) only for a small subset of edges making sure that the spanned graph remains connected. This task is in particular accomplishable in constant time for every edge of the reduced graph. The proof of the following lemma presents a convenient approach.

Lemma 4.14. *Let $m,n \in \mathbb{N}$ be natural numbers, let χ be any k-dimensional slice in $\mathbb{R}^8$ and consider $\mathcal{G}_\chi[m,n] = (V,E)$ being the incidence graph of $\mathcal{A}_\chi[m,n]$. Moreover, let $e \geq 1$ be a constant. The computation of a set $E^e_{in}(v) \subseteq E_{in}(v)$ and $E^e_{out}(v) \subseteq E_{out}(v)$ for every node $v \in V$ such that*

1. *$|E^e_{in}(v)| \leq e$ and*

2. *the edge set* $E^e = \bigcup_{v \in V} \bigcup_{u \in E^e_{in}(v)} \{uv\}$ *spans a connected subgraph of* $\mathcal{G}_\chi[m,n]$ *that connects every pair of nodes in* V

can be accomplished in $O(|\mathcal{E}[m,n]|^k)$ *time. If every node* $v \in V$ *corresponds to a face* $\varphi = \tau_\chi(\varphi(v))$ *that contains a valid point* P*, then the computation of* Update(uv) *for all* $uv \in E^e$ *can be done in* $O(|\mathcal{E}[m,n]|^k) \cdot T$ *time, where* T *is the worst case time to compute* P*.*

Proof. The set E^e is constructed in a recursive fashion. Firstly, since χ is k-dimensional, the set of k-face nodes u will have $E^e_{in}(u) = \emptyset$. Because $(k-1)$-face nodes v have exactly two superface nodes, such nodes have simply $E^e_{in}(v) = E_{in}(v)$. It is clear that the subgraph built only on k-face nodes u and $(k-1)$-face nodes v is connected only due to the edges uv given by $E^e_{in}(v)$.

The rest of the construction idea is to attach every remaining layer of k'-face nodes with $k' < k-1$ to e superfaces nodes. Then it follows from the recursive building scheme that the spanned graph is connected. This aim is achieved by a directed depth first search from every $(k-1)$-face node w. For every visited node u the search continues to a subface node v only as long as the set $E^e_{in}(v)$ has not reached the limit of e elements. In this case the node u is added to $E^e_{in}(v)$.

Since every node is visited at most e times, the process finishes after $O(|V| + |E|)$ operations. This quantity is at most $O(|\mathcal{E}[m,n]|^k)$ by Theorem 7.6 in [15] and thus, it is true that $E^e_{in}(v)$ and $E^e_{out}(v)$ can be computed in $O(|\mathcal{E}[m,n]|^k)$ time.

Every set $E^e_{in}(v)$ describe a number of edges directing from nodes $u \in V$ to v. This defines implicitly a set $E^e_{out}(u)$ for every node $u \in V$ containing exactly all edges directing from u to another node $v \in V$ that has $uv \in E^e_{in}(v)$. Obviously, $E^e_{out}(u)$ can be computed along with $E^e_{in}(v)$ without additional effort.

To compute the *Update* sets, the remaining edges of $\mathcal{G}_\chi[m,n]$ are enumerated once more. Whenever an edge uv is in $E^e_{in}(v)$, the set *Update*(uv) is computed according to its definition. Computing all *Update* sets takes time proportional to $e \cdot T \cdot \sum_{v \in V} |\ell(v)|$, since every node is processed at most e times. Since e is a constant and because it is possible to compute $\ell(v)$ for all nodes $v \in V$ in $O(|\mathcal{E}[m,n]|^k)$ time according to Lemma 2.24, this lemma is true. □

The choice of e is discretionary. In fact $e = 1$ would give a tree-like subgraph of $\mathcal{G}_\chi[m,n]$. Larger e lead just to more alternatives for traversal in depth first search. Moreover, the cardinality of the sets $E^e_{out}(v)$ are not necessarily bounded by a constant. However, for the time complexity of the algorithm it is sufficient that the sets $E^e_{in}(v)$ cannot be arbitrarily large. The algorithm in the next part performs undirected depth first search on E^e, the edge set

$$E^e = \bigcup_{v \in V} \bigcup_{u \in E^e_{in}(v)} \{uv\} = \bigcup_{u \in V} \bigcup_{v \in E^e_{out}(u)} \{uv\} \subseteq E.$$

Concluding, the preprocessing is made up from three steps, the computation of (1) the incidence graph $\mathcal{G}_\chi[m,n]$, (2) the $P'(v)$-labels for nodes and (3) the *Update* labels for edges. The following sections show that, based on the *Update* labels, the algorithm is able to save an $O(m^2)$ factor in its time complexity because it processes every label at most twice.

4.3.4 Implementation

The following introduces the generic image matching algorithm in detail. In principle, the algorithm sticks to the simple depth first search approach of the algorithm in Figure 4.3, but traverses along edges in E^e instead of E. The main difference, however, is the procedure of incrementally computing $A_2 = \Gamma_\chi(\varphi')$ during the processing of a node v with $\varphi' = \tau_\chi(\varphi(v))$ in the depth first search. Instead of constructing A_2 from scratch by the use of $f_{\varphi'}^{-1}$, the algorithm applies $A_1 = \Gamma_\chi(\tau_\chi(\varphi(u)))$, the image which belongs to the predecessor node u of v in the depth first traversal. By the help of the update information *Update*(uv) stored in $\mathcal{G}_\chi[m,n]$ for the edge $uv \in E^e$, the algorithm performs a number of update operations on A_1 to obtain the correct image A_2 for node v. The same update operations are also performed on the distortion measure δ. In this fashion the algorithm obtains A_2 and the distortion $\delta(A_2, B)$ by essentially fewer operations. Figure 4.4 lists a detailed description of the new projective image matching algorithm. The algorithm starts by performing all necessary preprocessing steps described in the previous subsection. As discussed earlier, the success of preprocessing, especially the computation of valid points, depends heavily on χ. For the considered subclasses of projective image matching the computation of valid points is demonstrated in Section 4.4.

For the traversal of an edge uv during the undirected depth first search the algorithm fetches update sets (i,j,i',j') from *Update*(uv) to recognize which indices (i,j) have to be updated. But instead of using a valid point $P \in \tau_\chi(\varphi(v))$ to compute $[\mathcal{P}(P)(i,j)]$ to obtain (i',j') the algorithm gets (i',j') directly from *Update*. This makes the graph searching component of the algorithm independent of the actual transformations represented in the subspace χ. Instead the application of transformations is restricted to the preprocessing. In this way the algorithm becomes generic and applicable for image matching under different sets of transformations given by slices χ in $\mathbb{R}^8$.

The same concept of reusability is aspired by the use of a RETURNVALUE procedure which keeps the algorithm robust against small changes in the problem specification. For example, if the algorithm should solve Problem 2.13 then the procedure RETURNVALUE returns $P'(v_{opt})$, the valid parameter point of the optimal face $\tau_\chi(\varphi(v_{opt}))$ with $f^{-1} = \mathcal{P}(P'(v_{opt}))$ that minimizes $\delta(f(A)[m], B)$. By exchanging the RETURNVALUE procedure to a simple Yes/No-output it is possible to solve also the decision Problem 2.14.

Algorithm `Generic Image Matching`
Input: Image A of size n and image B of size m.
Parameter: A slice χ in $\mathbb{R}^8$, a distortion measure $\omega : \mathcal{C} \times \mathcal{C} \to \mathbb{N}$ and a constant $e \geq 1$.
Output: $P \in \chi$ with $f^{-1} = \mathcal{P}[\mathcal{F}_{\mathtt{p}}]\,(P)$ such that $\delta(f(A)[m], B)$ is the minimum over all $f \in \mathcal{F}_{\chi}$.

```
1.  Procedure SEARCH($u, v$)                          /* undirected depth first searching */
2.      mark $v$ as seen;
3.      for all $(i, j, i', j')$ in Update($uv$) do            /* compute $\Gamma(\tau_\chi(\varphi(v)))$ */
4.          $\delta = \delta - \omega(A'(i,j), B(i,j))$;
5.          $A'(i,j) = A(i',j')$;
6.          $\delta = \delta + \omega(A'(i,j), B(i,j))$;
7.      end for
8.      if $\delta < \delta_{opt}$ then                          /* memorize better image match */
9.          $\delta_{opt} = \delta$;
10.         $v_{opt} = v$;
11.     end if
12.     for all $vw \in E^e$ do
13.         if $w$ is unseen then call SEARCH($v, w$);                  /* continue DFS */
14.     end for
15.     for all $(i, j, i', j')$ in Update($vu$) do            /* restore $\Gamma(\tau_\chi(\varphi(u)))$ */
16.         $\delta = \delta - \omega(A'(i,j), B(i,j))$;
17.         $A'(i,j) = A(i',j')$;
18.         $\delta = \delta + \omega(A'(i,j), B(i,j))$;
19.     end for

20. Procedure MAIN()                                             /* Main Program */
21.     call PREPROCESS($m, n, e$);   /* construct $\mathcal{G}_\chi[m,n]$, $E^e$, labels $P'$ and Update */
22.     $v_{opt} = v_{id}$;                               /* select identity transformation */
23.     mark $v_{id}$ as seen;
24.     for all $(i, j) \in z(m)$ do          /* prepare $A'$ as identity transformation of $A$ */
25.         $A'(i,j) = A(i,j)$;
26.     end for
27.     $\delta_{opt} = \delta = \delta(A', B)$;
28.     for all $v_{id}w \in E^e$ do
29.         if $w$ is unseen then call SEARCH($v_{id}, w$);                  /* start DFS */
30.     end for
31.     return RETURNVALUE($v_{opt}$);              /* compute return value ($P'(v_{opt})$) */
```

Figure 4.4: The generic image matching algorithm. The main procedure prepares the undirected depth first search on E^e. The search itself is realized recursively by the SEARCH procedure. With each call one face φ becomes *seen*, that means it is processed by updating indices and estimating the new distortion.

4.3.5 Analysis

The last step for this section is the analysis of the new image matching algorithm's performance:

Theorem 4.15. *Let A be an image of size n, B be an image of size m and let χ be any k-dimensional slice in $\mathbb{R}^8$. If every face φ in $\mathcal{H}_\chi[m,n]$ contains a valid point P, then the algorithm in Figure 4.4 determines in $O(|\mathcal{E}[m,n]|^k) \cdot T$ time a point $P \in \chi$ such that $f^{-1} = \mathcal{P}(P)$ minimizes $\delta(f(A)[m], B)$ over all $f \in \mathcal{F}_\chi$, where T is the worst case time to compute P.*

Proof. The correctness of the algorithm in Figure 4.4 follows straight from

- Lemma 4.8, because traversing all nodes of $\mathcal{G}_\chi[m,n]$ enumerates all faces φ of $\mathcal{A}_\chi[m,n]$ and consequently also all faces $\varphi' = \tau_\chi(\varphi)$ of $\mathcal{H}_\chi[m,n]$, and from
- Theorem 4.4, because for $\mathcal{F}_\chi$ the computation $\Gamma_\chi : \mathcal{H}_\chi[m,n] \to \mathcal{D}[\mathcal{F}_\chi, m](A)$ in the traversal supplies all elements of the dictionary $\mathcal{D}[\mathcal{F}_\chi, m](A)$.

For the running time of the algorithm consider first the preprocessing of the incidence graph $\mathcal{G}_\chi[m,n]$ with edges E^e for some $e \geq 1$. According to Lemma 4.14 the preprocessing needs $O(|\mathcal{E}[m,n]|^k) \cdot T$ time.

The worst case running time of the core algorithm is in $O(|\mathcal{E}[m,n]|^k)$. For that notice that the MAIN procedure runs in $O(m^2)$ time if the preprocessing and the call to SEARCH are ignored. Hence, in comparison to $O(|\mathcal{E}[m,n]|^k)$, this time amount is negligible. Then, because $\mathcal{G}_\chi[m,n]$ has $O(|\mathcal{E}[m,n]|^k)$ nodes (according to Theorem 7.6 in [15]), the procedure SEARCH is called $O(|\mathcal{E}[m,n]|^k)$ times. Moreover, since the SEARCH procedure applies the *Update*(uv) set of every edge in E^e at most two times, once for forward traversing and once for backtracking, it follows that the subsumption of all calls to SEARCH consumes running time proportional to $\sum_{uv \in E^e} |\mathit{Update}(uv)| + |\mathit{Update}(vu)|$. According to Lemma 4.14 it is possible to compute all *Update* sets in $O(|\mathcal{E}[m,n]|^k)$ time and thus, $\sum_{uv \in E^e} |\mathit{Update}(uv)| + |\mathit{Update}(vu)| \leq O(|\mathcal{E}[m,n]|^k)$. This completes the proof. □

4.4 Specific Transformation Classes and Valid Points

This section concludes Chapter 4 by applying the generic algorithm in Figure 4.4 to solve image matching for projective transformations and the specific subclasses $\mathcal{F}_{\mathtt{s}}$, $\mathcal{F}_{\mathtt{sr}}$, $\mathcal{F}_{\mathtt{t}}$, $\mathcal{F}_\ell$ and $\mathcal{F}_{\mathtt{a}}$. This needs in particular specific techniques to compute valid points for all faces in the respective arrangements.

4.4.1 Projective Image Matching

For projective transformations it is known from the Valid-Points Property 3.13 that every face in $\mathcal{H}[m,n]$ has a valid point. The efficient computation of valid points for the faces $\varphi = \tau(\varphi(v))$ of all nodes v in the incidence graph $\mathcal{G}[m,n]$ is technically involved but possible, as the following theorem states. The idea behind the theorem is to select points that are in φ and at the same time in $\mathcal{L}[m,n]$.

Theorem 4.16 (Proof in Section 4.5)**.** *Let $m,n \in \mathbb{N}$ be natural numbers and $\mathcal{G}[m,n] = (V,E)$ be the incidence graph of $\mathcal{A}[m,n]$. The computation of a valid point $P'(v) \in \tau(\varphi(v))$ for all nodes v of $\mathcal{G}[m,n]$ can be done in a total time of $O(m^{16}n^8)$.*

The proof of Theorem 4.16 works on basis of the idea to use the P labels computed by the algorithm of Edelsbrunner et al. [15, 16] to find nine linearly independent vertices in the boundary of $\varphi = \tau(\varphi(v))$ for every node v. These vertices, in turn, can be used to obtain a point which is roughly in the center of φ and thus, has a certain minimum distance to its boundary. The computation of $P'(v)$ is realized by rounding the center point in a certain fashion. Then $P'(v)$ is a point in $\mathcal{L}[m,n]$ and thus, it can be represented very efficiently, i. e., by $O(\log m + n)$ bits.

Based on the computation of valid points the generic image matching algorithm can be applied for projective transformations. The following corollary shows its running time for this particular case:

Corollary 4.17. *The image matching Problem 2.13 under projective transformations can be solved in $O(m^{16}n^8)$ time for given image A of size n and image B of size m.*

Proof. For the 8-slice $\chi = \mathbb{R}^8$ the computation of valid points $P'(v)$ can be done in $O(m^{16}n^8)$ time for all nodes v of $\mathcal{G}[m,n]$ according to Theorem 4.16. Since $\mathcal{G}[m,n]$ has $O(m^{16}n^8)$ nodes and edges this means that on average every valid point is computed in constant time T. Then, by Theorem 4.15 the running time of the whole algorithm is $O(m^{16}n^8)$, too. □

4.4.2 Image Matching for Projective Subclasses

Again, providing techniques to compute valid points basically solves the problem for all subclasses $\mathcal{F}_{\mathtt{s}}$, $\mathcal{F}_{\mathtt{sr}}$, $\mathcal{F}_{\mathtt{t}}$, $\mathcal{F}_{\ell}$ and $\mathcal{F}_{\mathtt{a}}$. The class of scalings represents the only exception which needs a little more effort. In fact, one problem with the use of Edelsbrunner's algorithm [15, 16] to compute the incidence graph is that it works only in k-dimensional slices with $k \geq 2$. Hence, for the slices $\chi_{\mathtt{a}}, \chi_{\ell}, \chi_{\mathtt{t}}$ and $\chi_{\mathtt{sr}}$ the application of the construction algorithm to compute $\mathcal{G}_{\chi}[m,n]$ in $O(|\mathcal{E}[m,n]|^k)$ time is valid since the dimension k of each slice is at least two. But

$\chi_{\mathtt{s}}$ has only dimension one. However, the input set of polynomials $\mathcal{E}[m,n]$ is not arbitrary and thus, $\mathcal{G}_{\chi_{\mathtt{s}}}[m,n]$ can be computed in output linear time.

Lemma 4.18 (Proof in Section 4.5). *Let $m, n \in \mathbb{N}$ be natural numbers. The graph $\mathcal{G}_{\chi_{\mathtt{s}}}[m,n] = (V, E)$ can be computed in $O(|\mathcal{E}[m,n]|)$ time.*

The main issue with the generic approach applied to projective subclasses is that, contrary to projective transformations, it is not at once clear whether every face of the respective arrangement $\mathcal{H}_\chi[m,n]$ contains a valid point. But the preprocessing depends on this property and thus, the following theorem provides this necessary condition as well as constant time techniques to compute all valid points.

Theorem 4.19 (Proof in Section 4.5). *Let $m, n \in \mathbb{N}$ be natural numbers and χ be any slice in $\{\chi_{\mathtt{t}}, \chi_{\mathtt{sr}}, \chi_{\mathtt{s}}\}$.*

1. *All faces in $\mathcal{A}_\chi[m,n]$ contain a valid point and consequently, all faces in $\mathcal{H}_\chi[m,n]$ contain a valid point.*
2. *For $k \geq 1$ all k-faces $\varphi \in \mathcal{A}_{\chi_\ell}[m,n]$ contain a valid point and consequently, all k-faces $\varphi' \in \mathcal{H}_{\chi_\ell}[m,n]$ contain a valid point.*
3. *All faces in $\mathcal{H}_{\chi_{\mathtt{a}}}[m,n]$ contain a valid point.*

Let $\mathcal{G}_\chi[m,n] = (V, E)$ be the incidence graph of $\mathcal{A}_\chi[m,n]$ for any slice $\chi \in \{\chi_{\mathtt{a}}, \chi_\ell, \chi_{\mathtt{t}}, \chi_{\mathtt{sr}}, \chi_{\mathtt{s}}\}$. Moreover let k be the dimension of slice χ. The computation of valid points $P'(v)$ for all nodes $v \in V$ representing faces $\varphi = \tau_\chi(\varphi(v)) \in \mathcal{H}_\chi[m,n]$ containing valid points can be done in $O(|\mathcal{E}[m,n]|^k)$ time.

Notice that Theorem 4.15 demands the existence of a valid point $P'(v)$ for every node v in the incidence graph. As Theorem 4.19 states, this is not necessarily the case in the slice χ_ℓ. However, the only exceptions are some 0-face nodes. Such nodes v can be tested in constant time, simply by checking if $P(v)$ is valid. In case not, it is easy to prune v from $\mathcal{G}_{\chi_\ell}[m,n]$ such that it is never visited in the generic depth first traversal.

As the labels $P'(v)$ can be computed in $O(|\mathcal{E}[m,n]|^k)$ time with k the dimension of χ, it is possible to apply the generic algorithm in Figure 4.4 to solve the image matching problem for all the transformation classes $\mathcal{F}_{\mathtt{s}}$, $\mathcal{F}_{\mathtt{sr}}$, $\mathcal{F}_{\mathtt{t}}$, $\mathcal{F}_\ell$ and $\mathcal{F}_{\mathtt{a}}$. In particular, the Theorems 4.15 and 4.19 imply the following:

Corollary 4.20 (Without proof). *Let A be an image of size n, B be an image of size m and let χ be a slice in $\{\chi_{\mathtt{a}}, \chi_\ell, \chi_{\mathtt{t}}, \chi_{\mathtt{sr}}, \chi_{\mathtt{s}}\}$. The algorithm in Figure 4.4 determines in $O(|\mathcal{E}[m,n]|^k)$ time a point $P \in \chi$ such that $f^{-1} = \mathcal{P}(P)$ minimizes $\delta(f(A)[m], B)$ over all $f \in \mathcal{F}_\chi$, where k is the dimension of χ.*

Since $\chi_{\mathtt{a}}$ has six dimensions, χ_ℓ has four dimensions, $\chi_{\mathtt{t}}$ and $\chi_{\mathtt{sr}}$ have two dimensions and $\chi_{\mathtt{s}}$ has one dimensions the corollary has the following implication:

Corollary 4.21 (Without proof). *For given image A of size n and image B of size m the image matching Problem 2.13 can be solved in*

- *$O(m^{12}n^6)$ time under affine transformations,*
- *$O(m^8n^4)$ time under linear transformations,*
- *$O(m^4n^2)$ time under translations,*
- *$O(m^4n^2)$ time under combinations of scalings and rotations and*
- *$O(m^2n)$ time under solely scalings.*

Corollary 4.20 gives an impression on the general applicability of the image matching algorithm proposed in this chapter and the techniques used therein. However, there are natural subclasses of projective transformations that are not covered by the given approach. Rotations $\mathcal{F}_{\mathtt{r}}$ are a basic example for this. As mentioned before, this is basically caused by keeping the notion of slices simple, which makes them strictly linear. In fact, it is easy to transfer the essential ideas to the quadratical subspace $\lambda \subset \mathbb{R}^8$ representing rotations. The generic algorithm applied for rotations yields:

Theorem 4.22 (Christian Hundt and Maciej Liśkiewicz [24, 25]). *For given image A of size n and image B of size m the image matching Problem 2.14 can be solved in $O(n^3)$ time under solely rotations.*

Notice that the theorem changes the image matching problem to the decision version. Although it is true that all presented techniques work for λ, i. e., there is a subdivision $\mathcal{H}_\lambda[m,n]$ of the subspace λ into faces φ containing valid points and an incidence graph $\mathcal{G}_\lambda[m,n] = (V,E)$ representing this subdivision, it is not generally possible to compute the valid points $P'(v)$ for all nodes $v \in V$. In particular, there are faces φ in $\mathcal{H}_\lambda[m,n]$ containing only one point, which, although being valid, has irrational components and thus, cannot be computed explicitly. Thus, if φ leads to the optimal solution, then it would not be possible to return a proper parameter point $P \in \varphi$. However, using the presented techniques, the algorithm can decide if there is a point P that represents a rotation f such that $\delta(f(A)[m], B) \leq \mathcal{T}$ for a given threshold $\mathcal{T}$.

4.5 Technical Proofs

The Proof of Lemma 4.2

Proof. The proof works similarly on all cases and the following considers only the first case: affine transformations. Let $P = (p_1, \ldots, p_8)^T$ be a point in $\chi_{\mathtt{a}}$. Then, since P is in $h^0(\ell_7)$ it follows that $p_7 = 0$ and from $P \in h^0(\ell_8)$ follows $p_8 = 0$.

This means for $f = \mathcal{P}(P)$ with $f(x,y) = \left(\frac{x'}{z'}, \frac{y'}{z'}\right)$ and $\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} p_1 & p_2 & p_3 \\ p_4 & p_5 & p_6 \\ p_7 & p_8 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$ that $z' = 1$ and thus, it follows that $f(x,y) = \begin{pmatrix} p_1 & p_2 & p_3 \\ p_4 & p_5 & p_6 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}$, which is an affine transformation by Definition 2.11. □

The Proof of Lemma 4.8

Proof. Let

$$\varphi' = \chi \cap \left(\bigcap_{k=1}^{w} H^{s'_k}(\ell_k) \right) \neq \emptyset$$

be any face in $\mathcal{H}_\chi[m,n]$ given by a specific choice of $s'_1, \ldots, s'_w \in \{+,-\}$. Then

$$\downarrow\!\varphi' = \chi \cap \left(\bigcap_{k=1}^{w} h^{s'_k}(\ell_k) \right)$$

is in $\mathcal{A}_\chi[m,n]$ and trivially $\tau_\chi(\downarrow\!\varphi') = \varphi'$. Thus, τ_χ is surjective.

Now consider any face

$$\varphi = \chi \cap \left(\bigcap_{k=1}^{w} h^{s_k}(\ell_k) \right) \neq \emptyset$$

in $\mathcal{A}_\chi[m,n]$ given by $s_1, \ldots, s_w \in \{+,0,-\}$. That $\varphi \subseteq \tau_\chi(\varphi)$ is obtained by showing that even for all $z \in \{0, \ldots, w\}$ the face

$$\varphi_z = \chi \cap \left(\bigcap_{k=1}^{r} h^{s_k}(\ell_k) \right) \subseteq \tau_\chi(\varphi_z).$$

This is clearly true for $z = 0$ where $\varphi_0 = \chi$ and $\tau_\chi(\varphi_0) = \chi$.

For every $z > 0$ the argument is true for φ_{z-1} by induction hypothesis. Hence, $\varphi_{z-1} \subseteq \tau_\chi(\varphi_{z-1})$. Then $\varphi_z = \varphi_{z-1} \cap h^{s_z}(\ell_z)$. Consequently, it is true $\varphi_z \subseteq \tau_\chi(\varphi_{z-1})$ and $\varphi_z \subseteq h^{s'_z}(\ell_z)$ which implies that φ_z is a subset of $\tau_\chi(\varphi_z) = \tau_\chi(\varphi_{z-1}) \cap h^{s'_z}(\ell_z)$. □

The Proof of Lemma 4.9

Proof. This challenge is studied in combinatorial geometry. Bounding this cardinalities needs the value of $|\mathcal{E}[m,n]|$ which is trivially $2(2m+1)^2(2n+1) = O(m^2 n)$. Consequently, from Theorem 1.3 in [15] follows that $\mathcal{A}_\chi[m,n] \in O(m^2 n)^k = O(m^{2k} n^k)$. From Lemma 4.8 follows $|\mathcal{H}_\chi[m,n]| \leq |\mathcal{A}_\chi[m,n]|$ and thus, the statement follows. □

The Proof of Theorem 4.10

Proof. The faces

$$\varphi_1' = \tau_\chi(\varphi_1) = \chi \cap \left(\bigcap_{k=1}^{w} H^{s_k'}(\ell_k) \right) \qquad \text{and} \qquad \varphi_2' = \tau_\chi(\varphi_2) = \chi \cap \left(\bigcap_{k=1}^{w} H^{S_k'}(\ell_k) \right)$$

are given by the signs $s_1', \dots, s_w', S_1', \dots, S_w' \in \{+, -\}$ according to Lemma 4.8. If there are valid points $P_1 \in \varphi_1'$ and $P_2 \in \varphi_2'$ then assume for $f_1^{-1} = \mathcal{P}(P_1)$ and $f_2^{-1} = \mathcal{P}(P_2)$ and some $(i,j) \in z(m)$ that the points $f_1^{-1}(i,j)$ and $f_1^{-2}(i,j)$ fall into different pixels $pix(i_1', j_1')$ and $pix(i_2', j_2')$ of $Pix[n]$.

If $i_1' \neq i_2'$ then there exists $i' \in \{-n, \dots, n+1\}$ such that without loss of generality $f_1^{-1}(i,j) \in \mathtt{v}_{i'}^{+} \cap \mathtt{v}_{(n+1)}^{-}$ and $f_1^{-1}(i,j) \in \mathtt{v}_{(-n)}^{+} \cap \mathtt{v}_{i'}^{-}$. Let

1. $H_{1,1} = H^{+}(\ell_{iji'}^{\mathtt{v}}) \cap H^{-}(\ell_{ij(n+1)}^{\mathtt{v}})$,
2. $H_{1,2} = H^{+}(-\ell_{iji'}^{\mathtt{v}}) \cap H^{-}(-\ell_{ij(n+1)}^{\mathtt{v}})$,
3. $H_{2,1} = H^{+}(\ell_{ij(-n)}^{\mathtt{v}}) \cap H^{-}(\ell_{iji'}^{\mathtt{v}})$ and
4. $H_{2,2} = H^{+}(-\ell_{ij(-n)}^{\mathtt{v}}) \cap H^{-}(-\ell_{iji'}^{\mathtt{v}})$.

By Lemma 3.4 it is the case that either $P_1 \in H_{1,1}$ or $P_1 \in H_{1,2}$ and either $P_2 \in H_{2,1}$ or $P_2 \in H_{2,2}$.

If $P_1 \in H_{1,1}$ and $P_2 \in H_{2,1}$ or $P_1 \in H_{1,2}$ and $P_2 \in H_{2,2}$, then the hyperplane defined by polynomial $\ell_{iji'}^{\mathtt{v}}$ separates P_1 and P_2. This implies $s_q' \neq S_q'$ for the index q of the polynomial $\ell_q = \ell_{iji'}^{\mathtt{v}}$. It follows that $s_q \neq S_q$ because s_q' and S_q' are defined on the basis of s_q and S_q.

Now let $P_1 \in H_{1,1}$ and $P_2 \in H_{2,2}$. Clearly, by definition, it is true $\ell_{ij-n}^{\mathtt{v}}(P_2) \leq 0$ and $\ell_{iji'}^{\mathtt{v}}(P_2) > 0$. If $P_2 = (p_1, \dots, p_8)^T$ it follows from $\frac{2n+1}{n+i'} > 0$ and $\frac{n+1-i'}{2n+1} > 0$ that

$$\begin{aligned} 0 &< \left(\frac{2n+1}{n+i'}\right) \cdot \left(\ell_{iji'}^{\mathtt{v}}(P_2) - \left(\frac{n+1-i'}{2n+1}\right) \cdot \ell_{ij-n}^{\mathtt{v}}(P_2) \right) \\ &= ip_1 + jp_2 + p_3 + (0.5i - i(n+1))p_7 + (0.5j - j(n+1))p_8 + (0.5 - (n+1)). \end{aligned}$$

Hence, $\ell_{ij(n+1)}^{\mathtt{v}}(P_2) > 0$. Since $\ell_{ij(n+1)}^{\mathtt{v}}(P_1) < 0$ it follows that the hyperplane $h^0(\ell_{ij(n+1)}^{\mathtt{v}})$ separates P_1 and P_2. Then $\ell_q = \ell_{ij(n+1)}^{\mathtt{v}}$ and again $s_q' \neq S_q'$ which implies $s_q \neq S_q$.

Finally, let $P_1 \in H_{1,2}$ and $P_2 \in H_{2,1}$. Since $\ell_{ij-n}^{\mathtt{v}}(P_2) \geq 0$ and $\ell_{iji'}^{\mathtt{v}}(P_2) < 0$ it follows in a similar fashion as above that $\ell_{ij(n+1)}^{\mathtt{v}}(P_2) < 0$. Because this time $\ell_{ij(n+1)}^{\mathtt{v}}(P_1) > 0$ the hyperplane defined by polynomial $\ell_q = \ell_{ij(n+1)}^{\mathtt{v}}$ separates P_1 and P_2 again. Consequently, $s_q \neq S_q$. $\square$

The Proof of Lemma 4.12

Proof. *Update*(uv) is defined properly only if the point P exists in $\tau_\chi(\varphi(v))$. Assume that $s_q \neq S_q$ for some polynomial $\ell_q = \ell^\star_{ijk'}$ and that $[f^{-1}(i,j)] = (i',j')$. Without loss of generality it is true that $\varphi(v)$ is a subface of $\varphi(u)$ and consequently $\varphi(u)$ is either subset of $h^+(\ell_q)$ or of $h^-(\ell_q)$ whereas $\varphi(v)$ is subset of $h^0(\ell_q)$. Consequently, $\varphi(v)$ is a subset of $h^0(\ell^\star_{ijk'})$. By definition *Update*(uv) contains (i,j,i',j') as $\ell^\star_{ijk'} \in \ell(v)$. □

The Proof of Theorem 4.16

Proof. Let v be any node in V and $\varphi' = \tau(\varphi(v))$. There is an 8-face φ in $\mathcal{A}[m,n]$ such that $\downarrow\varphi' = \varphi$. To compute $P'(v)$ the given approach determines nine linearly independent vertices in the boundary of φ and takes their average to obtain an interior point which is then also an interior point of φ'. Then this point is rounded to a lattice point in $\mathcal{L}[m,n]$ that remains in φ'.

The approach requires the computation of $\downarrow\tau(\varphi(v))$ for all $v \in V$. To realize this the whole node set of the incidence graph is labeled with $\Lambda : V \to V$ where for all $u, v \in V$ it is true that $\Lambda(v) = u$ if and only if $\downarrow\tau(\varphi(v)) = \varphi(u)$, i. e., $\varphi(u)$ is the same 8-face as specified by $\downarrow\tau(\varphi(v))$.

Clearly, for all nodes $v \in V$ representing an 8-face $\varphi(v)$ it is simply the case that $\Lambda(v) = v$. Moreover, for 7-face nodes v there are only two superfaces $\varphi(u_1)$ and $\varphi(u_2)$ incident to $\varphi(v)$ and the decision if $\Lambda(v) = u_1$ or $\Lambda(v) = u_2$ can be done in constant time by the help of $\ell(v)$. Consequently, computing Λ for all 7-face and all 8-face nodes is done in $O(m^{16}n^8)$ time.

For k-face nodes with $k < 7$ the computation of Λ is technically involved. The difficulty is that a k-face may have a large number of superface nodes v and it is complicated to decide which $\Lambda(v)$ contains the correct link.

To overcome these difficulties, the process computes additional directed edge sets $E_1, \ldots, E_7$ for the incidence graph such that for all $k \in \{1, \ldots, 7\}$ it is true that E_k is a set of edges between k-face nodes. Particularly, $uv \in E_k$ for some $k \in \{1, \ldots, 7\}$ if and only if there is a $(k+1)$-face $\varphi \subseteq \tau(\varphi(u))$ such that $\varphi(u)$ and $\varphi(v)$ are incident to φ. Notice that $uv \in E_k$ does not mean that $\tau(\varphi(u)) = \tau(\varphi(v))$. This happens if and only if $uv \in E_k$ and $vu \in E_k$.

The sets E_k and the labels Λ are computed recursively. Particularly, for all $k \in \{7, 6, \ldots, 1\}$ the label $\Lambda(v)$ on k-face nodes v is used to first compute the set E_k and secondly, based on E_k, the labels $\Lambda(u)$ for $(k-1)$-nodes are determined.

To be precise, E_k is obtained, by taking every k-face node u and adding an edge uv to all k-face node v which have a common superface node w with u that fulfills $\Lambda(u) = \Lambda(w)$. Traversing to all superface nodes of k-face nodes touches every edge in E between $(k+1)$-face nodes and k-faces nodes at most once. To find all edges of E_k the edges in E between $(k+1)$-face nodes and k-faces nodes are touched once again. Hence, E_k is computed in $O(|E|)$ time and $|E_k| \leq |E|$.

To compute $\Lambda(v)$ for all $(k-1)$-faces involves the computation of directed cycles in E_k. Consider the set neighborhood $E_{in}(v)$ of superface nodes of v. Then, $\Lambda(v) = w$ if and only if $u_1, \dots, u_r \in E_{in}(v)$ is a directed cycle, i. e., $u_r u_1 \in E_k$ and $u_i u_{i+1} \in E_k$ for all $i \in \{1, \dots, r-1\}$, and $\Lambda(u_1) = \dots = \Lambda(u_r) = w$.

This can be seen as follows: If $\Lambda(v) = w$, consider all superface nodes $u_1, \dots, u_r \in E_{in}(v)$ with $\varphi(u_i) \subseteq \tau(\varphi(v)), i \in \{1, \dots, r\}$. Clearly, $\Lambda(u_i) = w$ as $\tau(\varphi(u_i)) = \tau(\varphi(v))$. Without loss of generality, $\varphi(u_r)$ and $\varphi(u_1)$ as well as $\varphi(u_i)$ and $\varphi(u_{i+1})$ for all $i \in \{1, \dots, r-1\}$ are adjacent. The common superface φ of two adjacent faces $\varphi(u_i)$ and $\varphi(u_j)$ for $i, j \in \{1, \dots, r\}$ has the property $\varphi \in \tau(\varphi(v))$ and consequently, $u_i u_j \in E_k$ and $u_j u_i \in E_k$. This yields the directed cycle $u_1, \dots, u_r$.

Reversely, if $u_1, \dots, u_r \in E_{in}(v)$ is a directed cycle then $\varphi' = \tau(\varphi(u_1)) = \dots = \tau(\varphi(u_r))$. If $\tau(\varphi(v)) \neq \varphi'$ then there is a polynomial $\ell \in \ell(v)$ such that $\varphi' \subseteq H^+(\ell)$ and $\varphi(v) \subseteq H^-(\ell)$. But since $h^0(\ell)$ does not intersect φ' and because $\varphi(v)$ is in the boundary of φ' it follows that $\varphi(v)$ is not an 8-face. A contradiction to the Volume Property 3.11.

Finding all directed cycles in E_k can be done in $O(|E_k|)$ time using Tarjan's algorithm [44] for the strongly connected components in E_k. The result are labels i for k-face nodes u identifying that u in the ith cycle. Then, Λ can be computed in $O(V)$ time for all $(k-1)$-face nodes v by using the labels to search for the cycle in $E_{in}(v)$.

Consequently, computing $\Lambda(v)$ for all nodes $v \in V$ takes $O(|V| + |E|)$ time. Hence, if every 8-face v had a valid point $P'(v)$ then computing $P'(u)$ for all remaining nodes would work simply by taking $P'(\Lambda(u))$.

The computation of a valid point for every 8-face can be done in $O(|V| + |E|)$ time by depth first search, providing a set $\mathcal{V}(v)$ of nine linearly independent vertices in the boundary of every 8-face $\varphi(v)$. For $\mathcal{V}(v) = \{P_1, \dots, P_9\}$ a point $P = \frac{1}{9}(P_1 + \dots + P_9)$ can be computed in constant time. According to the proof of Lemma 3.10 the point P is the center of an 8-ball with radius $r \geq \frac{1}{4.5 \cdot 10^8 m^7 n^3}$ entirely contained in φ. To obtain $P'(v)$ the point $P = (v_1, \dots, v_8)^T$ is rounded to a lattice point of $\mathcal{L}[m, n]$. The computation of

$$\begin{pmatrix} t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \\ t_7 \\ t_8 \end{pmatrix} = \left[d^{-1} \cdot \begin{pmatrix} v_1 - 0.125 d L^{-2} \\ v_2 \\ v_3 \\ v_4 \\ v_5 - 0.5d \\ v_6 \\ v_7 \\ v_8 \end{pmatrix} \right].$$

for $L = 3 \cdot 10^{16} m^{13} n^6$ and $d^{-1} = 1.5 \cdot 10^9 m^7 n^3 + 0.5$ takes constant time. Then $\mathcal{L}[m, n](t_1, \dots, t_8)$ is the lattice point closest to P. Incident lattice points have a mutual distance of less than $\sqrt{8}d < r$ and thus, $P'(v) = \mathcal{L}[m,n](t_1,\ldots,t_8)$ is in φ. Since $\mathcal{G}[m,n]$ has $O(m^{16}n^8)$ nodes and edges it follows that $P'(v)$ can be computed in constant time for every node v and thus, the whole process takes $O(m^{16}n^8)$ time. □

The Proof of Lemma 4.18

Proof. All points $P = (p_1,\ldots,p_8)^T$ in the slice χ_{s} have $p_1 = p_5$ and $p_2 = p_3 = p_4 = p_6 = p_7 = p_8 = 0$. Consequently, for χ_{s} the polynomials of $\mathcal{E}[m,n]$ have a simpler form, i. e., $\mathcal{E}[m,n]$ contains

$$\ell^{\mathrm{v}}_{iji'}(p_1,\ldots,p_8) = ip_1 + (0.5 - i') \text{ and } \ell^{\mathrm{h}}_{ijj'}(p_1,\ldots,p_8) = jp_5 + (0.5 - j')$$

for all $(i,j) \in z(m)$ and all $i',j' \in \{-n,\ldots,n+1\}$.

Then, $\mathcal{G}_{\chi_{\mathsf{s}}}[m,n]$ is simply a chain $u_1v_1u_2v_2\ldots u_rv_ru_{r+1}$ of nodes where $u_1,\ldots,u_{r+1}$ represent 1-faces and $v_1,\ldots,v_r$ represent 0-faces. Clearly, every 0-face $\varphi(v)$ is an intersection of χ with $h^0(\ell)$ for some polynomial $\ell \in \mathcal{E}[m,n]$, i. e., $\varphi(v)$ contains only the point

$$P(v) = \left(\frac{i'-0.5}{i}, 0, 0, 0, \frac{i'-0.5}{i}, 0, 0, 0\right)^T$$

for some $i' \in \{-n,\ldots,n+1\}$ and some $i \in \{-m,\ldots,-1,1,\ldots,m\}$. In this case $\ell(v)$ contains $\ell^{\mathrm{v}}_{iji'}$ and $\ell^{\mathrm{h}}_{jii'}$ for all $j \in \{-m,\ldots,m\}$.

Then, since for all $k \in \{1,\ldots,r-1\}$ it is true that the nodes v_k and v_{k+1} have points $P(v_k) = (p_1,\ldots)^T$ and $P(v_{k+1}) = (p'_1,\ldots)^T$ with $p_1 < p'_1$, it follows that the computation of $\mathcal{G}_{\chi_{\mathsf{s}}}[m,n]$ needs to determine and sort all possible fractions $\frac{i'-0.5}{i}$ for $i' \in \{-n,\ldots,n+1\}$ and $i \in \{-m,\ldots,-1,1,\ldots,m\}$.

By the limits of i' and i, there are $O(mn)$ fractions and sorting them takes $O(mn \log mn)$ time. This is less than $O(|\mathcal{E}[m,n]|) = O(m^2n)$ time unless n is exponentially larger than m.

In the second case notice that points $P(v_k) = (p_1,\ldots)^T$ and $P(v_{k+1}) = (p'_1,\ldots)^T$ of nodes v_k and v_{k+1} for all $k \in \{1,\ldots,r-1\}$ have $\frac{1}{2m^2} \leq |p_1 - p'_1| \leq 2mn$. Hence, just to decide the relation between two fractions it is enough to represent them as a fractional binary number of $2\log(m) + 1 + \log(mn) + 1 \leq 5\log(mn)$ bits. Based on this representation it is possible to bucket sort the fractions using mn buckets. According to the approximate representation this takes only five iterations and thus, the whole sorting runs in $O(mn)$ time.

By the observation about $\ell(v)$, the computation of this set takes $O(m)$ time for every combination of i' and i. Since $\ell(u) = \emptyset$ for all 1-face nodes u, it follows, that the sets $\ell(v)$ for all nodes $v \in V$ can be computed in $O(m^2n)$ time.

Finally, after the order of faces has been determined, $P(v)$ can be computed in full precision for all 0-face nodes v in $O(mn)$ time. Afterwards, the computation of $P(u)$ for 1-face nodes u is simply taking the mean of $P(v)$ for the two neighbor

0-face nodes. This takes $O(mn)$ time, too. For the extremal nodes u_1 and u_{r+1} the points $P(u_1) = P(v_1) - 1$ and $P(u_{r+1}) = P(v_r) + 1$ are computed in constant time. □

The Proof of Theorem 4.19

Proof. Notice first that for all slices $\chi \in \{\chi_{\mathtt{a}}, \chi_\ell, \chi_{\mathtt{t}}, \chi_{\mathtt{sr}}, \chi_{\mathtt{s}}\}$ it is true that all contained points $P = (p_1, \ldots, p_8)^T \in \chi$ have $p_7 = p_8 = 0$. It follows that $h^0(e_1) \cap \chi = h^0(e_2) \cap \chi$ and $h^0(e[i,j]) \cap \chi = \emptyset$ for all $(i,j) \in z(m)$. Consequently, all invalid points to be considered are in $h^0(e_1) \cap \chi$.

It is straightforward that both $\chi_{\mathtt{sr}}$ and $\chi_{\mathtt{s}}$ contain only one invalid point $P_0 = (0, \ldots, 0)^T$ and that $\chi_{\mathtt{t}}$ does not contain invalid points at all. Hence, if there was a face in $\mathcal{A}_{\chi_{\mathtt{sr}}}[m,n]$ or in $\mathcal{H}_{\chi_{\mathtt{s}}}[m,n]$ without valid points then it must be a 0-face with P_0 being the only contained point. However, since P_0 does not belong to the hyperplane $h^0(\ell)$ for any $\ell \in \mathcal{E}[m,n]$ this is impossible.

This means that for $\chi \in \{\chi_{\mathtt{t}}, \chi_{\mathtt{sr}}, \chi_{\mathtt{s}}\}$ it is possible to compute $P'(v)$ simply by computing the center point of $\varphi(v)$. This is clearly a point contained in $\tau_\chi(\varphi(v))$. If it happens that the center is P_0, then let $r = \max\{m,n\}$. It is convenient to select the point $P'(v)$ in a very close environment around P_0, say in a distance of $\frac{1}{r^{1000}}$. This means that the computation of $P'(v)$ for all $v \in V$ can be done in $O(|\mathcal{A}_\chi[m,n]|) = O(|\mathcal{E}[m,n]|^k)$ time.

Now consider the 4-slice χ_ℓ of linear transformations. It is easy to find 0-faces in $\mathcal{A}_{\chi_\ell}[m,n]$ where the only contained point is part of $h^0(e_1)$. Consequently, there may also be 0-faces in $\mathcal{H}_{\chi_\ell}[m,n]$ with an invalid point. Hence, for every 0-face node $v \in V$ it easy to check whether $e_1(P(v)) \neq 0$ and in this case one sets $P'(v) = P(v)$.

Every 1-face $\varphi \in \mathcal{A}_{\chi_\ell}[m,n]$ is contained in the intersection of three linearly independent hyperplanes $h^0(\ell_1)$, $h^0(\ell_2)$ and $h^0(\ell_3)$ defined by three polynomials ℓ_1, ℓ_2 and ℓ_3 in $\mathcal{E}[m,n]$. Notice that the polynomials become simpler since every point $P = (p_1, \ldots, p_8)^T$ in χ_ℓ has $p_3 = p_6 = p_7 = p_8 = 0$. Then there remain two kinds of polynomials, one with non-zero coefficients for p_1 and p_2 and one with non-zero coefficients for p_4 and p_5. Since every choice of three polynomials from one kind are linearly dependent, it follows without loss of generality that

$$\begin{aligned}
\ell_1(p_1, \ldots, p_8) &= i_1 p_1 + j_1 p_2 + (0.5 - i'_1), \\
\ell_2(p_1, \ldots, p_8) &= i_2 p_1 + j_2 p_2 + (0.5 - i'_2) \text{ and} \\
\ell_3(p_1, \ldots, p_8) &= i_3 p_4 + j_3 p_5 + (0.5 - j')
\end{aligned}$$

for some $(i_1, j_1), (i_2, j_2), (i_3, j_3) \in z(m)$ and $i'_1, i'_2, j' \in \{-n, \ldots, n+1\}$. If $i_3 = j_3 = 0$ then ℓ_3 would not define a hyperplane $h^0(\ell_3)$ and thus, without loss of generality let $i_3 \neq 0$. From the linear independence it follows that $i_1 j_2 - i_2 j_1 \neq 0$

and every point $P = (p_1, \dots, p_8)^T$ in φ has

$$p_1 = \frac{j_2(i'_1 - 0.5) - j_1(i'_2 - 0.5)}{i_1 j_2 - i_2 j_1},$$
$$p_2 = \frac{i_1(i'_2 - 0.5) - i_2(i'_1 - 0.5)}{i_1 j_2 - i_2 j_1} \text{ and}$$
$$p_4 = \frac{(j'_1 - 0.5) - j_3 p_5}{i_3},$$

i. e., the coordinates p_1 and p_2 are fixed and the coordinates p_4 and p_5 are variable, but depend on each other.

Now assume that φ does not contain a single valid point, hence, for all $P = (p_1, \dots, p_8)^T \in \varphi$ it is true that $p_1 p_5 - p_2 p_4 = 0$. Since

$$\begin{aligned} 0 &= p_1 p_5 - p_2 p_4 \\ &= p_5 \cdot \frac{i_3(j_2(i'_1 - 0.5) - j_1(i'_2 - 0.5)) + j_3(i_1(i'_2 - 0.5) - i_2(i'_1 - 0.5))}{i_3(i_1 j_2 - i_2 j_1)} + \\ & \frac{(j' - 0.5)(i_2(i'_1 - 0.5) - i_1(i'_2 - 0.5)}{i_3(i_1 j_2 - i_2 j_1)} \end{aligned}$$

it follows that

$$p_5 = \frac{(j' - 0.5)(i_1(i'_2 - 0.5) - i_2(i'_1 - 0.5)}{i_3(j_2(i'_1 - 0.5) - j_1(i'_2 - 0.5)) + j_3(i_1(i'_2 - 0.5) - i_2(i'_1 - 0.5))}.$$

If the denominator of the fraction was not zero then p_4 and p_5 would be determined. This implies that only one point in the intersection of $\chi_\ell \cap h^0(\ell_1) \cap h^0(\ell_2) \cap h^0(\ell_3)$ is invalid. Since φ contains more than one point this is a contradiction. Consequently, the denominator is zero and by multiplying $i_1 j_2 - i_2 j_1$ one obtains that

$$\begin{aligned} 0 &= i_3(j_2(i'_1 - 0.5) - j_1(i'_2 - 0.5)) + j_3(i_1(i'_2 - 0.5) - i_2(i'_1 - 0.5)) \\ &= i_3 p_1 + j_3 p_2. \end{aligned}$$

This implies that

$$\begin{aligned} 0 &= p_1 p_5 - p_2 p_4 \\ &= -\frac{j_3}{i_3} p_2 p_5 - p_2 \frac{(j'_1 - 0.5) - j_3 p_5}{i_3} \\ &= -p_2 \frac{j'_1 - 0.5}{i_3} \end{aligned}$$

which can be true only if $p_2 = 0$. Since $p_1 p_5 = p_2 p_4$ it follows that at least one of p_1 and p_5 is zero, too. If $p_5 = 0$ then p_4 is fixed and again, only one point in the intersection of $\chi_\ell \cap h^0(\ell_1) \cap h^0(\ell_2) \cap h^0(\ell_3)$ is invalid. This is a contradiction to the assumption of all points in φ being invalid.

Hence, p_1 must be zero. But since $p_2 = 0$ it follows that $j_2(i'_1 - 0.5) - j_1(i'_2 - 0.5) = i_1(i'_2 - 0.5) - i_2(i'_1 - 0.5) = 0$ which implies $i_1 j_2 - i_2 j_1 = 0$. This is impossible by the linear independence of $h^0(\ell_1)$ and $h^0(\ell_2)$ and consequently, the first assumption of φ containing only invalid points must be false. Consequently, if v is the 1-face node then $P'(v)$ is simply the center point of $\varphi(v)$. Only if the center is invalid $P(v')$ is chosen in the near environment by slightly modifying p_5 and thus, p_4.

In the rest of cases a valid point is chosen similarly. For a first attempt $P'(v)$ is set to the center $P = (p_1, \ldots, p_8)^T$ of $\varphi(v)$. If this point is invalid then it is altered slightly in the following fashion:

For 2-faces $\varphi(v)$, which are contained in the intersection of two linearly independent hyperplanes $h^0(\ell_1)$ and $h^0(\ell_2)$ of two polynomials from different kind it is true that

$$i_1 p_1 + j_1 p_2 = i' - 0.5 \quad \text{and} \quad i_2 p_4 + j_2 p_5 = j' - 0.5$$

for some $(i_1, j_1), (i_2, j_2) \in z(m)$ and $i', j' \in \{-n, \ldots, n+1\}$. Clearly, every point $(p'_1, \ldots, p'_8)^T \in \varphi(v)$ is defined by

$$\begin{aligned}
p'_1 &= p_1 + w_1 j_1,\\
p'_2 &= p_2 + w_1(-i_1),\\
p'_3 &= p_3 = 0,\\
p'_4 &= p_4 + w_2 j_2,\\
p'_5 &= p_5 + w_2(-i_2),\\
p'_6 &= p_6 = p'_7 = p_7 = p'_8 = p_8 = 0
\end{aligned}$$

for some w_1, w_2 which implies that

$$p'_1 p'_5 - p'_2 p'_4 = w_1(i_1 p_4 + j_1 p_5) - w_2(i_2 p_1 + j_2 p_2) + w_1 w_2 (i_1 j_2 - i_2 j_1).$$

The point $P'(v)$ is found by choosing w_1 and w_2 appropriately. If $(i_1 p_4 + j_1 p_5) \neq 0$ set w_1 to a small value, say $\frac{1}{r^{1000}}$ and let $w_2 = 0$. Otherwise, if $(i_2 p_1 + j_2 p_2) \neq 0$ then let $w_1 = 0$ and w_2 be the small value. If both, $(i_1 p_4 + j_1 p_5)$ and $(i_2 p_1 + j_2 p_2)$ are zero but $(i_1 j_2 - i_2 j_1) \neq 0$ then set both w_1 and w_2 to small values. Finally, if all $(i_1 p_4 + j_1 p_5)$, $(i_2 p_1 + j_2 p_2)$ and $(i_1 j_2 - i_2 j_1)$ are zero, then $p_1 p_4 + p_2 p_5 = 0$ follows. Since $p_1 p_5 - p_2 p_4 = 0$ this works only if $p_1 = p_2 = 0$ or $p_4 = p_5 = 0$ which is impossible by $(p_1, \ldots, p_8)^T$ being part of $h^0(\ell_1)$ and $h^0(\ell_2)$.

If $h^0(\ell_1)$ and $h^0(\ell_2)$ are of the same kind or if v stands for a 3- or 4-face, then without loss of generality p_1 and p_2 may be fixed but p_4 and p_5 are free without depending on each other. In this case an invalid center point P on $h^0(e_1)$ can be fixed by choosing $P'(v) = (p'_1, \ldots, p'_8)$ by

$$\begin{aligned}
&p'_1 = p_1, p'_2 = p_2, p'_3 = p_3 = 0,\\
&p'_4 = p_4 + \epsilon p_2,\\
&p'_5 = p_5 + \epsilon p_1,\\
&p'_6 = p_6 = p'_7 = p_7 = p'_8 = p_8 = 0
\end{aligned}$$

for some small number ϵ, say $\epsilon = \frac{1}{3r^{1001}}$. Then $P'(v)$ remains in $\varphi(v)$ since $\|P - P'(v)\| = \epsilon\sqrt{p_1^2 + p_2^2} \le \frac{1}{r^{1000}}$ and $p'_1p'_5 - p'_2p'_4 = \epsilon^2(p_1^2 + p_2^2) > 0$.

Hence, all nodes v of $\mathcal{G}_{\chi_\ell}[m,n]$ can be $P'(v)$-labeled in constant time. This means that the whole operation takes $O(|\mathcal{A}_{\chi_\ell}[m,n]|) = O(|\mathcal{E}[m,n]|^4)$ time.

Finally, for the 6-slice χ_{a} of affine transformations the proof works just as for projective transformations. In particular the author [21] shows that every face in $\mathcal{H}_{\chi_{\mathsf{a}}}[m,n]$ has a positive volume and that χ_{a} admits a lattice $\mathcal{L}_{\mathsf{a}}[m,n]$ of valid points that hit every face in $\mathcal{H}_{\chi_{\mathsf{a}}}[m,n]$. Then, the computation of $P'(v)$ for all nodes v of the incidence graph $\mathcal{G}_{\chi_{\mathsf{a}}}[m,n]$ can be done analogously as in the proof of Theorem 4.16. This works in $O(|\mathcal{A}_{\chi_{\mathsf{a}}}[m,n]|) = O(|\mathcal{E}[m,n]|^6)$ time. $\square$

Chapter 5

Complexity Aspects and Parallelism

Chapter 3 analyzes the structure of projective image transformations. The central property revealed is a partition of the parameter space into polygonal faces such that points within each face correspond to equal projective transformations. This structure is applied to find a sequential polynomial time algorithm to enumerate all projective transformations $\mathcal{D}[m](A)$ of a given image A of size n. Then, the dictionary $\mathcal{D}[m](A)$ is used to compute the best match against another given image B of size m.

The aim of this chapter is to locate the projective image matching problem in the structure of complexity classes. This does not mean the search for a better time bound but instead it aims at locating the problem relative to the hierarchy of complexity classes

$$AC_O \subset TC_O \subseteq NC_1 \subseteq LOGSPACE \subseteq \ldots \subseteq PTIME,$$

inside *PTIME*, the set of problems decidable in polynomial time. According to Chapter 4 it is clear that the decision Problem 2.14 of projective image matching is in *PTIME*.

However, the other classes in the hierarchy imply structural computational advantages against the hardness of *PTIME*. If, e. g., the image matching problem was in *LOGSPACE* then it would by solvable by the use of just a logarithmic amount of space. The time improved algorithm in Figure 4.4 of the previous chapter involves the preprocessing of an enormously big data structure, $\mathcal{G}[m,n]$, which cannot be established in logarithmic space in a straightforward fashion. But the simple algorithm in Figure 4.2 is independent of $\mathcal{G}[m,n]$ and can in fact be regarded as a deterministic logarithmic space algorithm for projective image matching.

This chapter provides an even more general result. The relation of the image matching problem to the classes AC_O, TC_O, NC_1 would also mean that it could be solved very efficiently in a parallel fashion, i. e., in sub-polynomial time by the usage of at most polynomially many processors. Particularly, AC_O and TC_O contain problems that can be decided in constant parallel running time by unbounded fan-in processors and NC_1 consists of decision problems solvable in poly-logarithmic time by processors with at most two input ports. But the computation of $\mathcal{G}[m, n]$ seems to be a challenge also for a parallel model of computation. Moreover, it is not clear how to transform the sequential algorithm in Figure 4.2 into a parallel one.

In this chapter the decision Problem 2.14 of projective image matching is shown to be TC_O-complete. This result has no direct effect for the complexity of the search version but it is also shown that a best transformation can be computed by equal means of computational expense. Interestingly, TC_O is a very natural complexity class, because it exactly expresses the complexity of a variety of most basic problems in computation, such as integer addition, multiplication, comparison and division. Even sorting integers is a TC_O-complete problem.

However, regarding the practical importance of the results in this chapter it is indispensable to mention that they rather have to be seen as an attempt to gain insight into the structural properties of the problem. The presented approach reveals a circuit that is far too big to be realized in a practical setting which is partially caused by the use of a very weak model of TC_O. Although being a natural choice for complexity analysis more powerful models produce much smaller circuits.

The next section starts with an introduction to the necessary concepts of parallel computation and a useful connection between the complexity classes AC_O, TC_O and logic. Secondly, Section 5.2 presents a parallel approach of solving the search version of projective image matching and in this way it shows also that the decision version is in TC_O. Then, Section 5.3 substantiates that TC_O is the natural complexity class for image matching by showing the TC_O-completeness of the decision problem.

5.1 Concepts of TC_O

The complexity class U_D-TC_O was originally described as a generalization of U_D-AC_O, a class of decision problems that can be computed by circuits. A *circuit* N can be imagined as a *directed acyclic graph* where nodes, also called *gates*, compute *Boolean functions*. Gates gain input *truth values* from predecessor gates and distribute computation results to all their successor gates. If N has n sources and m sinks, then it computes a function $g : \{0,1\}^n \to \{0,1\}^m$, i. e., N computes for every input string of length n an output string of length m. This makes circuits weaker than other computational models, which can compute functions $\{0,1\}^* \to \{0,1\}^*$. Consequently one considers families $\mathcal{N} = \{N_0, N_1, N_2, \ldots\}$ of circuits to compute f for every input length n with an individual circuit N_n. On the other hand, such families can be surprisingly powerful because they may not necessarily be finitely describable in the traditional sense. A usual workaround is a *uniformity model* which demands that every circuit N_n of $\mathcal{N}$ can be described by a *Turing machine* $\mathcal{M}_{\mathcal{N}}$ with resource bounds related to n. Usually $\mathcal{M}_{\mathcal{N}}$ is chosen much weaker than the computational power of $\mathcal{N}$ to avoid the obscuration of $\mathcal{N}$'s complexity. This chapter considers only *DLOGTIME-uniform families* $\mathcal{N}$ where $\mathcal{M}_{\mathcal{N}}$ has to verify in $O(\log n)$ time whether N_n fulfills a given structural property. This means, the machine can answer in $O(\log n)$ time questions like, e. g., *Gate i computes the* AND*-function* or *Gate i is a predecessor of gate j.* The following gives a short definition of the complexity classes AC_O and TC_O. For a substantial overview, see the text book of Vollmer [45].

Definition 5.1. *The class **DLOGTIME-uniform** FAC_O, denoted as U_D-FAC_O, contains all functions $\{0,1\}^* \to \{0,1\}^*$ which can be computed by **constant-depth, polynomial-size families $\mathcal{N}$ of DLOGTIME-uniform circuits**, i. e., where (1) every gate computes an* AND-, OR- *or* NOT*-function, (2) all circuits N_n can be verified by a Turing machine $\mathcal{M}_{\mathcal{N}}$ that runs in $O(\log n)$-time, (3) the number of gates in N_n grows only polynomially in n and (4) regardless of n, the length of any path in N_n from input to output is not longer than a constant.*

The function class U_D-FAC_O induces the class U_D-AC_O of decision problems. It contains all problems Π which can be specified by a function $g \in U_D$-FAC_O, i. e., $g : \{0,1\}^* \to \{0,1\}$ is a function with $g(s) = 1 \Leftrightarrow s \in \Pi$.

*If gates can also compute the **threshold-functions** T_k which returns true if at least k inputs are true, then the generated function class is called **DLOGTIME-uniform** FTC_O and is denoted as U_D-FTC_O. Accordingly, U_D-TC_O is the class of decision problems decidable by a function in U_D-FTC_O.*

By definition, U_D-AC_O and U_D-FAC_O are subsets of U_D-TC_O and U_D-FTC_O, respectively. Consequently, U_D-FAC_O-reductions are suited well to define completeness in the class U_D-TC_O:

Definition 5.2. *A problem Π is U_D**-TC_O-complete** if Π belongs to U_D-TC_O and if for all $\Pi' \in U_D$-TC_O there is a function r in U_D-FAC_O such that for all $s \in \{0,1\}^*$ it is true $s \in \Pi' \Leftrightarrow r(s) \in \Pi$.*

Clearly, since U_D-FAC_O-reductions are *transitive*, it is sufficient for U_D-TC_O-completeness to find another problem Π' known to be U_D-TC_O-complete and then provide a function r in U_D-FAC_O such that Π' is reducible to Π via r. U_D-TC_O-completeness states that a problem belongs to the hardest in that class. A canonical U_D-TC_O-complete problem is MAJ, containing all strings over $\{0,1\}^*$ with a *majority* of positive bits [10].

First Order Logic

The complexity classes U_D-AC_O and U_D-TC_O can be characterized by first order logic. Particularly, both classes can be defined as languages $\Pi \subseteq \{0,1\}^*$ that can be described by certain *first order formulas.* A comprehensive introduction to first order logic and in particular the connection to circuit families is given by Vollmer [45]. This chapter applies the following fragment of first order logic:

Definition 5.3. *Let $\mathcal{V}$ be the set of* ***variables****, i. e., an infinite set of strings over some alphabet, and let* 0 *and* $(|s|-1)$ *be* ***constant symbols****. Then* $X[\mathsf{v}]$, $\mathrm{Bit}[\mathsf{u},\mathsf{v}]$ *and* $<[\mathsf{u},\mathsf{v}]$ *are (atomic)* ***first order formulas*** *for all* $\mathsf{u},\mathsf{v} \in \mathcal{V} \cup \{0,(|s|-1)\}$*. If* v *is a variable then it is a free variable of* $X[\mathsf{v}]$ *and if* u *or* v *are variables then they are free variables of* $\mathrm{Bit}[\mathsf{u},\mathsf{v}]$ *and* $<[\mathsf{u},\mathsf{v}]$*. The three atomic formulas contain no bound variables.*

Let ϕ_1 be a first order formula with ***free variables*** $\mathcal{V}(\phi_1)$ *and* ***bound variables*** $\mathcal{B}(\phi_1)$ *and let ϕ_2 be a first order formulas with free variables $\mathcal{V}(\phi_2)$ and bound variables $\mathcal{B}(\phi_2)$. Without loss of generality let (1) $\mathcal{B}(\phi_1) \cap \mathcal{B}(\phi_2) = \emptyset$, (2) $\mathcal{V}(\phi_1) \cap \mathcal{B}(\phi_1) = \emptyset$ and (3) $\mathcal{V}(\phi_2) \cap \mathcal{B}(\phi_2) = \emptyset$. Otherwise, replacing all bound variables in ϕ_1 and ϕ_2 by new variables achieves these properties. Then*

1. *the* conjunction $(\phi_1 \wedge \phi_2)$ *and the* disjunction $(\phi_1 \vee \phi_2)$ *are first order formulas with free variables $\mathcal{V}(\phi_1) \cup \mathcal{V}(\phi_2)$ and bound variables $\mathcal{B}(\phi_1) \cup \mathcal{B}(\phi_2)$,*

2. *the* negation $(\neg\phi_1)$ *is a first order formula with free variables $\mathcal{V}(\phi_1)$ and bound variables $\mathcal{B}(\phi_1)$ and*

3. *on condition that $\mathsf{v} \in \mathcal{V}(\phi_1)$ is a free variable $(\forall \mathsf{v}\ \phi_1)$ and $(\exists \mathsf{v}\ \phi_1)$ are first order formulas with free variables $\mathcal{V}(\phi_1) \setminus \{\mathsf{v}\}$ and bound variables $\mathcal{B}(\phi_1) \cup \{\mathsf{v}\}$.*

A first order formula ϕ without free variables, i. e., with $\mathcal{V}(\phi) = \emptyset$, is called a ***sentence****.*

First order formulas with ***majority quantifier*** *are an extension of the concept which is built on the recursive construction scheme above. If ϕ is a first order formula with a free variable v then $(Q\mathsf{v}\ \phi)$ is a* ***first order formula with majority quantifier*** *over free variables $\mathcal{V}(\phi) \setminus \{\mathsf{v}\}$ and bound variables $\mathcal{B}(\phi) \cup \{\mathsf{v}\}$.*

The interpretation of a formula ϕ is defined recursively over its construction:

Definition 5.4. *Let ϕ be a first order formula with free variables $\mathcal{V}(\phi)$. A* ***structure*** *for ϕ is given by a string $s \in \{0,1\}^*$ together with an* ***assignment*** *$a : \mathcal{V}(\phi) \to \{0,\dots,|s|-1\}$. For convenience let*

$$a'(\mathsf{v}) = \begin{cases} a(\mathsf{v}) & : \textit{if } \mathsf{v} \textit{ is a free variable,} \\ 0 & : \textit{if } \mathsf{v} \textit{ is the constant } 0 \textit{ and} \\ |s|-1 & : \textit{if } \mathsf{v} \textit{ is the constant } (|s|-1). \end{cases}$$

Then (s,a) is a ***model*** *of ϕ, denoted by $(s,a) \models \phi$, if and only if*

$\phi = X[\mathsf{v}]$ *and* $s(a'(\mathsf{v})) = 1$,

$\phi = \mathrm{Bit}[\mathsf{u},\mathsf{v}]$ *and* $b(a'(\mathsf{v})) = 1$ *for the binary representation* b *of the value* $a'(\mathsf{u})$ [1],

$\phi = <[\mathsf{u},\mathsf{v}]$ *and* $a'(\mathsf{u}) < a'(\mathsf{v})$,

$\phi = (\phi_1 \wedge \phi_2)$ *and it is true that* $(s, a_1) \models \phi_1$ *and* $(s, a_2) \models \phi_2$ *where* a_1 *and* a_2 *are the restrictions of assignment* a *to the free variables in* ϕ_1*, respectively* ϕ_2*,*

$\phi = (\phi_1 \vee \phi_2)$ *and it is true that* $(s, a_1) \models \phi_1$ *or* $(s, a_2) \models \phi_2$ *and* a_1 *and* a_2 *are defined as above,*

$\phi = (\neg\phi_1)$ *and* $(s, a) \not\models \phi_1$*,*

$\phi = (\forall \mathsf{v}\ \phi_1)$ *and for all* $i \in \{0, \dots, |s| - 1\}$ *it is true that* $(s, a_i) \models \phi_1$ *where* $a_i : \mathcal{V}(\phi) \cup \{\mathsf{v}\} \to \{0, \dots, |s| - 1\}$ *is the assignment defined as*

$$a_i(\mathsf{u}) = \begin{cases} i & : \textit{if } \mathsf{u} = \mathsf{v} \\ a(\mathsf{u}) & : \textit{otherwise} \end{cases}$$

for all $\mathsf{u} \in \mathcal{V}(\phi) \cup \{\mathsf{v}\}$ *or*

$\phi = (\exists \mathsf{v}\ \phi_1)$ *and there is at least one* $i \in \{0, \dots, |s| - 1\}$ *such that* $(s, a_i) \models \phi_1$ *where the assignment* a_i *is defined as above.*

The interpretation of first order formulas with majority quantifier works similarly. A string s *and assignment* a *model* $\phi = (Q\mathsf{v}\ \phi_1)$ *if and only if there are at least* $\left\lceil \frac{|s|}{2} \right\rceil$ *choices of* $i \in \{0, \dots, |s| - 1\}$ *such that* $(s, a_i) \models \phi_1$ *where the assignment* a_i *is defined as above.*

For the sake of straightforwardness the definition does not resolve the interpretation of formulas with respect to an empty string s. Particularly, in that case the universe is the empty set and there is no proper way of assigning values to variables and constants or interpreting the predicate symbols X, Bit and $<$. These difficulties can be overcome if a formula can identify a structure with an empty string by using an additional predicate. However, such an exception is never necessary for the formulas in this thesis and thus, it is simply assumed that structures consist of nonempty strings.

The meaning of a sentence ϕ, i. e., a formula without free variables, is defined relative to the string s only. This is denoted either by $s \models \phi$ or by $s \not\models \phi$. The next definition shows how to apply this property to use first order sentences for the description of decision problems.

[1]Notice that the value $a'(\mathsf{u})$ needs at most $log_2|s|$ bits. Nevertheless, by the use of leading zeros the string b consists of $|s|$ bits.

Definition 5.5. *If ϕ is a first order sentence (with majority quantifier) then $\Pi_\phi \subseteq \{0,1\}^*$ is the set of binary strings defined as*

$$\Pi_\phi = \left\{ s \;\middle|\; s \models \phi \right\}.$$

The class of all sets Π_ϕ that can be described by a first order sentence ϕ is denoted as FO. Moreover, $FO[Q]$ is the class of all sets Π_ϕ describable by first order sentences ϕ with majority quantifier.

The reason why first order logic can be applied in connection with circuit design is formulated by the following theorem:

Theorem 5.6 (Mix-Barrington, Immerman, Straubing [35])**.** *$FO = U_D\text{-}AC_O$ and $FO[Q] = U_D\text{-}TC_O$.*

By the use of these equalities it is possible to show the membership of a decision problem in $U_D\text{-}AC_O$ or $U_D\text{-}TC_O$ by expressing the problem in an appropriate logic.

The descriptions of problems, however, may involve very long formulas. To improve readability of such formulas this chapter applies a modular design principle.

Definition 5.7. *Let ϕ be a first order formula with free variables $\mathcal{V}(\phi) = (\mathsf{v}_1, \dots, \mathsf{v}_n)$ and bound variables $\mathcal{B}(\phi)$. Moreover let $\mathsf{u}_1, \dots, \mathsf{u}_n$ be n arbitrary variables in $\mathcal{V} \setminus \mathcal{B}(\phi)$. Then $\phi[\mathsf{u}_1, \dots, \mathsf{u}_n]$ is the formula gained by replacing in ϕ every occurrence of v_i by u_i for all $i \in \{1, \dots, n\}$.*

Hence, now the free variables of ϕ can be used as input (and output) parameters. This means that ϕ can be applied as a sub formula in a sentence that replaces $\mathsf{v}_1, \dots, \mathsf{v}_n$ from outside to pass arguments to ϕ. To demonstrate this technique consider the formula $\text{Plus}[\mathsf{u}, \mathsf{v}, \mathsf{w}]$ described by Vollmer [45], which contains three free variables $\mathsf{u}, \mathsf{v}, \mathsf{w}$. Using Plus as a sub formula it is convenient to express more complex settings in shortened formulas, like

$$\exists \mathsf{x}\ \exists \mathsf{y}\ X[\mathsf{x}] \wedge \text{Plus}[\mathsf{x}, \mathsf{x}, \mathsf{y}] \wedge X[\mathsf{y}].$$

The relation between the values of the variables $\mathsf{u}, \mathsf{v}, \mathsf{w}$ expressed in Plus are now true for the arguments x and y because they replace all occurrences of $\mathsf{u}, \mathsf{v}, \mathsf{w}$. Hence, every model s of this sentence contains at least two zero bits $s(x) = s(y) = 0$ such that $2x = y$.

Enhancing First Order Logic

In certain cases it is hard to find compact sentences to describe a problem. To resolve such difficulties this subsection discusses some known techniques used in first order logic. To remain self contained some of the statements are proven.

Firstly, this chapter applies the standard notation of $\Rightarrow$, i. e., for formulas ϕ and ϕ' the formula $\neg\phi \vee \phi'$ can be replaced by $\phi \Rightarrow \phi'$. Moreover, if ϕ'' is a formula, too, then the formula

$$(\phi \Rightarrow \phi') \wedge (\neg\phi \Rightarrow \phi'')$$

expressing the meaning of if-then-else, can be shortened as

$$\phi \Rightarrow \phi' \not\Rightarrow \phi''.$$

Secondly, first order formulas can express calculations involving the four basic integer operations. To use this efficiently the following notation is applied frequently:

Definition 5.8. *Every variable* v *and all constants* $k \in \mathbb{N}$ *are called* ***terms****. If* t_1 *and* t_2 *are terms with variables* $\mathcal{V}(t_1)$ *and* $\mathcal{V}(t_2)$ *then* $(t_1 + t_2)$, $(t_1 - t_2)$, $(t_1 \cdot t_2)$ *and* $\left\lfloor \frac{t_1}{t_2} \right\rfloor$ *are terms with variables* $\mathcal{V}(t_1) \cup \mathcal{V}(t_2)$ *and* $(t_1 < t_2)$, $(t_1 \leq t_2)$, $(t_1 = t_2)$, $(t_1 \neq t_2)$, $(t_1 \geq t_2)$ *and* $(t_1 > t_2)$ *are* ***arithmetic formulas*** *with free variables* $\mathcal{V}(t_1) \cup \mathcal{V}(t_2)$.

Let $\phi = (t_1 \star t_2)$ *be an arithmetic formula for* $\star \in \{<, \leq, =, \neq, \geq, >\}$ *where* t_1, t_2 *are terms over variables* $\mathcal{V}(\phi)$. *Moreover let* s *be a string and* $a : \mathcal{V}(\phi) \to \{0, \ldots, (|s| - 1)\}$ *be an assignment. The* ***value*** $a(t)$ ***of a term*** $t \in \{t_1, t_2\}$ *relative to* (s, a) *is*

$$a(t) = \begin{cases} a(\mathsf{v}) & : \text{ if } t = \mathsf{v} \text{ is a variable,} \\ \min\{k, |s| - 1\} & : \text{ if } t = k \text{ is a constant,} \\ \min\{a(t') + a(t''), |s| - 1\} & : \text{ if } t = (t' + t'') \text{ for terms } t', t'', \\ \max\{a(t') - a(t''), 0\} & : \text{ if } t = (t' - t'') \text{ for terms } t', t'', \\ \min\{a(t') \cdot a(t''), |s| - 1\} & : \text{ if } t = (t' \cdot t'') \text{ for terms } t', t'', \\ \left\lfloor \frac{a(t')}{a(t'')} \right\rfloor & : \text{ if } t = \left\lfloor \frac{t'}{t''} \right\rfloor \text{ for terms } t', t'' \text{ and } a(t'') \neq 0, \\ |s| - 1 & : \text{ if } t = \left\lfloor \frac{t'}{t''} \right\rfloor \text{ for terms } t', t'' \text{ and } a(t'') = 0. \end{cases}$$

Then $(s, a) \models \phi$ *if and only if* $a(t_1)$ *and* $a(t_2)$ *are related as described by* $\star$.

Theorem 5.9 (Proof in Section 5.4)**.** *Every arithmetic formula* ϕ *can be equivalently expressed as a first order formula.*

This chapter applies additional notation for terms, like for example computing the maximum $\max\{\cdot, \cdot\}$, the absolute value $|\cdot|$ etc. which are easily reduced to the basic cases introduced in the theorem.

In some cases it is difficult to express certain relations in first order sentences just because the values of variables are restricted to $\{0, \ldots, |s|-1\}$. This difficulty can be overcome by the use of long variables:

Definition 5.10. ***Long variables*** *are an extension of the notion of variables in first order formulas given in Definition 5.3. If ϕ is a first order formula with a free* long *variable $\mathbf{v}$, denoted* bold faced*, and (s, a) is a structure for ϕ then $a(\mathbf{v})$ is a value ranging in $\{-|s|^k + 1, \ldots, |s|^k - 1\}$ for some sufficiently large but fixed $k \in \mathbb{N}$. Universal and existential quantification over $\mathbf{v}$ as well as the interpretation of $<[\cdot, \cdot]$ works in a straightforward fashion. To define the meaning of $X[\mathbf{v}]$ let any structure (s, a) be a model for the formula if and only if $a(\mathbf{v}) \in \{0, \ldots, |s| - 1\}$ and $s(a(\mathbf{v})) = 1$. Moreover, if $\mathbf{u}$ is another long variable then any model (s, a) of a formula* $\mathrm{Bit}[\mathbf{u}, \mathbf{v}]$ *fulfills $a(\mathbf{v}) \geq 0$ and $b(a(\mathbf{v})) = 1$ where b is a binary string with $|s|^k - 1$ bits representation the value $|a(\mathbf{u})|$ by inserting leading zeros.*

The justification of using long variables is related to the common practice of ignoring constant factors in computational complexity. In fact, long variables can be established by using a constant number of standard variables to represent a value ranging in $\{-|s|^k + 1, \ldots, |s|^k - 1\}$ for some fixed $k \in \mathbb{N}$. Consequently, it is a standard technique to assume that variables in first order variables can take long values:

Theorem 5.11 (Without proof)**.** *Formulas ϕ containing long variables are first order expressible.*

The proof of the theorem works by encoding the value of a long variable into the values of multiple standard variables. Hence, according to the theorem it would be justified to use standard variables and assume them to take long values whenever necessary. However, to make this chapter's proofs easier to follow, formulas use the notion of bold faced long variables to indicate the expression of long or negative values. The values of standard variables remain in $\{0, \ldots, |s| - 1\}$.

Clearly, Theorem 5.9 and Theorem 5.11 can be combined to obtain that first order formulas can express the four basic integer operations even on long values. Notice however that Theorem 5.11 does not allow majority quantification over long variables.

A further inconvenience occurs by the use of the majority quantifier. To improve the usability the following definition introduces an additional quantifier:

Definition 5.12. *Let ϕ be a first order formula (with majority quantifier) and let $\mathcal{V}(\phi)$ and $\mathcal{B}(\phi)$ be the corresponding sets of free and bound variables. Moreover let t, n and v be free variables of ϕ. Then $\phi' = (Q^{\mathsf{t}}_{\mathsf{n}} \mathsf{v}\ \phi)$ is a first order formula with* ***counting quantifier*** *over free variables $\mathcal{V}(\phi') = \mathcal{V}(\phi) \setminus \{\mathsf{v}\}$ and bound variables $\mathcal{B}(\phi') = \mathcal{B}(\phi) \cup \{\mathsf{v}\}$.*

A string s and an assignment $a : \mathcal{V}(\phi') \to \{0, \ldots, |s| - 1\}$ are a model of ϕ', i. e., $(s, a) \models \phi'$, if and only if

1. $a(\mathsf{n}) < \left\lfloor \frac{|s|}{2} \right\rfloor$,
2. $a(\mathsf{t}) \leq a(\mathsf{n}) + 1$ *and*

3. *for exactly* $a(\mathsf{t})$ *choices of* $i \in \{0, \ldots, |s|-1\}$ *it is true that* $(s, a_i) \models \phi$ *where* $a_i : \mathcal{V}(\phi) \to \{0, \ldots, |s|-1\}$ *is the assignment defined as*

$$a_i(\mathsf{u}) = \begin{cases} i & : \text{ if } \mathsf{u} = \mathsf{v} \\ a(\mathsf{u}) & : \text{ otherwise} \end{cases}$$

for all $\mathsf{u} \in \mathcal{V}(\phi)$.

Theorem 5.13 (Proof in Section 5.4)**.** *Formulae with counting quantifier can be equivalently expressed by formulas with majority quantifier.*

5.2 Membership in TC_O

To show that Problem 2.13 is in U_D-FTC_O this section gives a first order formula with majority quantifier. The difficulty with this approach is that $FO[Q]$ contains decision problems whereas Problem 2.13 is an optimization problem. To overcome this the section proceeds in three steps.

Firstly, the image matching Problem 2.13 is described as a function IM^u, mapping unary encodings of two images A and B to strings encoding a transformation f that minimizes the distortion between $f(A)[m]$ and B. In the second step, the problem to decide if a given bit in the output of IM^u is positive is expressed using a $FO[Q]$ sentence. This implies that such a decision problem is in U_D-TC_O. The third step introduces the same function IM for a binary encoding of A and B and reduces the problem to decide output bits of IM to the unary version, showing that the problem is in U_D-TC_O, too. This makes the proof that Problem 2.13 is in U_D-FTC_O possible.

5.2.1 Image Matching with Unary Encoded Color Values

The following definition describes the unary encoding:

Definition 5.14. *Let* $\Pi^u_{in} \subseteq \{0,1\}^*$ *be the set of binary strings*

$$s = enc^u(n) \mid enc^u(m) \mid enc^u(A) \mid enc^u(B)$$

encoding two digital images A *and* B *unary by*

- $enc^u(n) = 1^n0$ *and* $enc^u(m) = 1^m0$*, the encoding of two numbers* n *and* m *giving the supports* $z(n)$ *and* $z(m)$ *of* A *and* B*,*

- $enc^u(A)$*, the encoding of image* A*'s color information as the Little Endian* $C \cdot (2n+1)^2$ *bits binary representation of the number*

$$\sum_{j=0}^{2n} \sum_{i=0}^{2n} 2^{C\cdot(j(2n+1)+i)+A(i-n,j-n)},$$

(possibly with leading zeros) where every block of C bits encodes one of the colors $\mathcal{C} = \{0, \ldots, C-1\}$ unary by setting the cth bit positive and all other bits zero, and

- *$enc^u(B)$, the analogous encoding of B as the Little Endian $C \cdot (2m+1)^2$ bits binary representation of the number*

$$\sum_{j=0}^{2m} \sum_{i=0}^{2m} 2^{C\cdot(j(2m+1)+i)+B(i-m,j-m)}.$$

Moreover, let $\Pi_{out} \subseteq \{0,1\}^*$ be the set of binary strings

$$z = enc^b(t_1) \mid \ldots \mid enc^b(t_8),$$

encoding eight integer numbers $t_1, \ldots, t_8 \in \{-L, \ldots, L\}$ with $L = 3 \cdot 10^{16} m^{13} n^6$ such that for all $r \in \{1, \ldots, 8\}$ it is true that $enc^b(t_r)$ is the $70\lceil \log_2 mn \rceil$ bit two's complement representation of t_i. Notice that $70\lceil \log_2 mn \rceil$ bits are enough to encode a number in $\{-L, \ldots, L\}$, since

$$\log_2 L = \log_2 3 \cdot 10^{16} + 13 \log_2 m + 6 \log_2 n \leq 55 + 13 \log_2 mn \leq 68 \lceil \log_2 mn \rceil.$$

Based on Π_{in}^u and Π_{out} and for any given distortion measure $\omega : \mathcal{C} \times \mathcal{C} \to \mathbb{N}$ projective image matching can be described as the function $IM_\omega^u : \Pi_{in}^u \to \Pi_{out}$, where every input string $s \in \Pi_{in}^u$ encodes an image A of size n and an image B of size m and the output string $z = IM_\omega^u(s)$ encodes eight integers $t_1, \ldots, t_8$ that define an inverse projective transformation $f^{-1} = \mathcal{P}(\mathcal{L}[m,n](t_1, \ldots, t_8))$ which minimizes $\delta(f(A)[m], B)$.

But there may be multiple transformations f that minimize the distortion. To make IM_ω^u a function it is necessary to fix one of them. Consequently, IM_ω^u returns an encoding z of eight integers $t_1, \ldots, t_8$ that, secondary to representing an optimal transformation, give the smallest number

$$\sum_{i=1}^{8} (t_i + L)(2L+1)^{i-1}.$$

Hence, IM_ω^u chooses the transformation with minimum parameter t_8. If this leaves ambiguity IM_ω^u selects from the remaining transformations the one with minimum parameter t_7 and so forth. This function leads to the following decision problem:

Problem 5.15. UNARY PROJECTIVE IMAGE MATCHING kTH BIT

Input: *A string $s \in \{0,1\}^*$.*

Constraints:

Image distortion measure $\omega : \mathcal{C} \times \mathcal{C} \to \mathbb{N}$.

Question: *Is string s of the form $s'1^k$ with $s' \in \Pi^u_{in}$ and $k \in \mathbb{N}$ such that the kth bit in string $z = IM^u_\omega(s')$ is positive?*

One can think of a family $\mathcal{N}$ of DLOGTIME-uniform constant depth polynomial size threshold circuits taking strings s as input by providing an input line for every bit in s and giving a single output line to decide Problem 5.15. To show the existence of such a family $\mathcal{N}$ this section develops a first order sentence with majority quantifiers.

Lemma 5.16. *For any distortion measure $\omega : \mathcal{C} \times \mathcal{C} \to \mathbb{N}$ Problem 5.15 can be expressed as a first order sentence ϕ with majority quantifiers. Thus, the problem is in U_D-TC_O.*

Proof. The central idea behind the proof is that all the points P from the lattice $\mathcal{L}[m, n]$ can be described by the sentence ϕ. For each lattice point P the sentence ϕ can also represent the transformed image $f(A)[m]$ given by the inverse projective transformation $f^{-1} = \mathcal{P}(P)$ corresponding to P. According to Corollary 4.5 and the Lattice Property 3.15 this gives the whole dictionary $\mathcal{D}[m](A)$. Finally ϕ formalizes for all elements of $\mathcal{D}[m](A)$ the notion $\delta(f(A)[m], B)$ and the conditions to find a minimum. The sentence ϕ is of the following form

```
$\phi = \exists \mathsf{n}\ \exists \mathsf{m}\ \exists \mathsf{k}$ (
    (DecodeParam[$\mathsf{n}, \mathsf{m}, \mathsf{k}$]) $\wedge$
    (ImageMatch[$\mathsf{n}, \mathsf{m}, \mathsf{k}$])
)
```

where DecodeParam[$\mathsf{n}, \mathsf{m}, \mathsf{k}$] decodes the values n, m and k from the string s into variables n, m and k of ϕ and ImageMatch[$\mathsf{n}, \mathsf{m}, \mathsf{k}$] describes the essential part of the image matching procedure.

The main task of DecodeParam is to check whether s properly encodes a string $s' \in \Pi^u_{in}$ and a natural number $k \in \mathbb{N}$. This means in particular that for all pairs (s, a) of strings s and assignments a with $a(\mathsf{n}) = n$, $a(\mathsf{m}) = m$ and $a(\mathsf{k}) = k$ it is true

$$\begin{aligned}&(s, a) \models \text{DecodeParam}[\mathsf{n}, \mathsf{m}, \mathsf{k}] \quad \Longleftrightarrow \\ &s = enc^u(n) \mid enc^u(m) \mid enc^u(A) \mid enc^u(B) \mid 1^k\end{aligned}$$

for some image A of size n and image B of size m. The first task of DecodeParam is to make sure that the value of n equals the number of positive bits before the first zero and that the value of m is equal to the number of positive bits between first and second zero. After this check, DecodeParam has the values of m and n and based on them it is possible to find out if the length of s is sufficiently large, i.e., if s encodes images A and B of size n, respectively m, then it must be the case that $|s| \geq n+m+2+C \cdot ((2n+1)^2+(2m+1)^2)$. Hence, after DecodeParam has validated the length of s it is able to locate 1^k in the string s by the values of n and m which are used to determine the encoding lengths of A and B and to skip $enc^u(A)$ and $enc^u(B)$. This enables the formula to check if the value of k is equal to the number of positive bits at the end of the string. Finally, the formula tests by the use of two sub formulas ImageA and ImageB, if every block of C bits in the representation of A and B properly encodes a color value from $\mathcal{C}$:

$$
\begin{aligned}
&\mathrm{DecodeParam}[\mathsf{n},\mathsf{m},\mathsf{k}] = \exists \mathsf{u}\ \exists \mathbf{v}\ \forall \mathsf{x}\ \forall \mathbf{i}\ \forall \mathbf{j}\ \exists \mathsf{a}\ \exists \mathsf{b}\ \Big(\\
&\qquad (\neg X[\mathsf{n}]) \wedge \\
&\qquad (\mathsf{x} < \mathsf{n} \ \Rightarrow\ X[\mathsf{x}]) \wedge \\
&\qquad (\neg X[\mathsf{u}]) \wedge \\
&\qquad (\mathsf{n} < \mathsf{x} < \mathsf{u} \ \Rightarrow\ X[\mathsf{x}]) \wedge \\
&\qquad (\mathsf{m} = \mathsf{u} - \mathsf{n} - 1) \wedge \\
&\qquad \big(\mathbf{v} = \mathsf{m} + \mathsf{n} + 2 + C \cdot ((2\mathsf{n}+1)^2 + (2\mathsf{m}+1)^2)\big) \wedge \\
&\qquad (\mathbf{v} \leq (|s|-1)+1) \wedge \\
&\qquad (\mathbf{v} \leq \mathsf{x} \leq (|s|-1) \ \Rightarrow\ X[\mathsf{x}]) \wedge \\
&\qquad (\mathsf{k} = (|s|-1) - \mathbf{v} + 1) \wedge \\
&\qquad ((|\mathbf{i}| \leq \mathsf{n}) \wedge (|\mathbf{j}| \leq \mathsf{n}) \ \Rightarrow\ \mathrm{ImageA}[\mathsf{n},\mathsf{m},\mathbf{i},\mathbf{j},\mathsf{a}]) \wedge \\
&\qquad ((|\mathbf{i}| \leq \mathsf{m}) \wedge (|\mathbf{j}| \leq \mathsf{m}) \ \Rightarrow\ \mathrm{ImageB}[\mathsf{n},\mathsf{m},\mathbf{i},\mathbf{j},\mathsf{b}]) \\
&\Big)
\end{aligned}
$$

If s is a model of sentence ϕ then (s, a) is a model of DecodeParam where a is an assignment with $a(\mathsf{n}) = n$, $a(\mathsf{m}) = m$ and $a(\mathsf{k}) = k$ and s is of the form $s = s'1^k$ with

$$s' = enc^u(n) \mid enc^u(m) \mid enc^u(A) \mid enc^u(B)$$

for some image A of size n and image B of size m. This implies that s has $|s| = n+m+2+C \cdot ((2n+1)^2+(2m+1)^2)+k$ bits. Since $m, n \geq 0$ this means that $|s| \geq C$ which implies also that the values of basic variables can describe all color values in $\mathcal{C}$. Moreover, $|s| \geq |z(m)|$ and $|s| \geq |z(n)|$ and thus, long variables can represent positive and negative numbers of absolute values polynomially in m and n.

The verification of a proper encoding of A and B is described by the help of ImageA and ImageB. In fact it is true that these sub formulas provide access to all color values of A and B. Particularly, if s is a model of ϕ and a an assignment giving n and m the values n and m encoded in s then ImageA fulfills

$$(s, a) \models \mathrm{ImageA}[\mathsf{n},\mathsf{m},\mathbf{i},\mathbf{j},\mathsf{c}] \quad \Longleftrightarrow \quad A(a(\mathbf{i}), a(\mathbf{j})) = a(\mathsf{c}).$$

The use of long variables $\mathbf{i}$ and $\mathbf{j}$ results from the existence of negative pixel coordinates. It would be possible to omit the usage of long variables at this point but they make the formula easier to follow. Then, based on the information on m

and n, the sub formula ImageA provides the above functionality by determining the offset of the color information about $A((a(\mathbf{i}), a(\mathbf{j}))$ in the string s. Finally, the formulas makes sure that the color value encoded at the corresponding position equals $a(\mathsf{c})$:

```
ImageA$[\mathsf{n}, \mathsf{m}, \mathbf{i}, \mathbf{j}, \mathsf{c}] = \exists \mathsf{u}\ \exists \mathsf{v}\ \forall \mathsf{x}\ \big($
    $(|\mathbf{i}| > \mathsf{n} \ \Rightarrow\ \mathsf{c} = 0) \wedge$
    $(|\mathbf{j}| > \mathsf{n} \ \Rightarrow\ \mathsf{c} = 0) \wedge$
    $(\mathsf{u} = \mathsf{m} + \mathsf{n} + 2 + ((\mathbf{j} + \mathsf{n}) \cdot (2\mathsf{n} + 1) + (\mathbf{i} + \mathsf{n})) \cdot C) \wedge$
    $(|\mathbf{i}| \leq \mathsf{n} \wedge |\mathbf{j}| \leq \mathsf{n}) \ \Rightarrow \big($
        $(\mathsf{u} \leq \mathsf{v} < \mathsf{u} + C) \wedge$
        $(X[\mathsf{v}]) \wedge$
        $(\mathsf{u} \leq \mathsf{x} < \mathsf{v} \ \Rightarrow\ \neg X[\mathsf{x}]) \wedge$
        $(\mathsf{v} < \mathsf{x} < \mathsf{u} + C \ \Rightarrow\ \neg X[\mathsf{x}]) \wedge$
        $(\mathsf{c} = \mathsf{v} - \mathsf{u})$
    $\big)$
$\big)$
```

Note that the formula does not only check whether the block in s encoding $A((a(\mathbf{i}), a(\mathbf{j}))$ contains a positive bit at position $a(\mathsf{c})$ but also that it contains zero bits at all other positions. In this way it is checked whether the respective block has a proper unary encoding of a color value. Consequently, all blocks of $enc^u(A)$ are verified by the application of ImageA in DecodeParam for all possible assignments a giving $\mathbf{i}$ and $\mathbf{j}$ a value in $z(n)$. Additionally, if the value of $\mathbf{i}$ or of $\mathbf{j}$ goes beyond support $z(n)$ then the formula ImageA expresses that the corresponding pixel value is zero.

Similar statements are true for the following sub formula ImageB:

```
ImageB$[\mathsf{n}, \mathsf{m}, \mathbf{i}, \mathbf{j}, \mathsf{c}] = \exists \mathsf{u}\ \exists \mathsf{v}\ \forall \mathsf{x}\ \big($
    $(|\mathbf{i}| > \mathsf{m} \ \Rightarrow\ \mathsf{c} = 0) \wedge$
    $(|\mathbf{j}| > \mathsf{m} \ \Rightarrow\ \mathsf{c} = 0) \wedge$
    $\big(\mathsf{u} = \mathsf{m} + \mathsf{n} + 2 + C(2\mathsf{n} + 1)^2 + ((\mathbf{j} + \mathsf{m}) \cdot (2\mathsf{m} + 1) + (\mathbf{i} + \mathsf{m})) \cdot C\big) \wedge$
    $(|\mathbf{i}| \leq \mathsf{m} \wedge |\mathbf{j}| \leq \mathsf{m}) \ \Rightarrow \big($
        $(\mathsf{u} \leq \mathsf{v} < \mathsf{u} + C) \wedge$
        $(X[\mathsf{v}]) \wedge$
        $(\mathsf{u} \leq \mathsf{x} < \mathsf{v} \ \Rightarrow\ \neg X[\mathsf{x}]) \wedge$
        $(\mathsf{v} < \mathsf{x} < \mathsf{u} + C \ \Rightarrow\ \neg X[\mathsf{x}]) \wedge$
        $(\mathsf{c} = \mathsf{v} - \mathsf{u})$
    $\big)$
$\big)$
```

After the verification formalisms above the development of sub formula ImageMatch will follow a top-down approach of describing the essential part of the matching process. This means that for every pair (s, a) of a string $s = s'1^k$ with $s' = enc^u(n)|enc^u(m)|enc^u(A)|enc^u(B)$ and an assignment $a(\mathsf{n}) = n$, $a(\mathsf{m}) = m$ and $a(\mathsf{k}) = k$ it is true

$$(s, a) \models \text{ImageMatch}[\mathsf{n}, \mathsf{m}, \mathsf{k}] \iff z = \text{IM}^u_\omega(s') \ \wedge \ z(k) = 1.$$

To achieve this behavior ImageMatch formalizes the sampling of the whole lattice $\mathcal{L}[m, n]$:

```
$\text{ImageMatch}[\mathsf{n},\mathsf{m},\mathsf{k}] = \exists\mathbf{L}\ \exists\mathbf{t}_1\ \dots\ \exists\mathbf{t}_8\ \exists\mathbf{T}\ \forall\mathbf{t}'_1\ \dots\ \forall\mathbf{t}'_8\ \forall\mathbf{T}'\ ($
    $\big(\mathbf{L} = 3 \cdot 10^{16}\mathsf{m}^{13}\mathsf{n}^6\big) \wedge$
    $((|\mathbf{t}_1| \le \mathbf{L}) \wedge \dots \wedge (|\mathbf{t}_8| \le \mathbf{L})) \wedge$
    $(\text{TestMatch}[\mathsf{n},\mathsf{m},\mathbf{L},\mathbf{t}_1,\dots,\mathbf{t}_8,\mathbf{T}]) \wedge$
    $((|\mathbf{t}'_1| \le \mathbf{L}) \wedge \dots \wedge (|\mathbf{t}'_8| \le \mathbf{L}) \wedge \text{TestMatch}[\mathsf{n},\mathsf{m},\mathbf{L},\mathbf{t}'_1,\dots,\mathbf{t}'_8,\mathbf{T}'])\ \Rightarrow ($
        $(\mathbf{T} \le \mathbf{T}') \wedge$
        $(\mathbf{T} = \mathbf{T}')\ \Rightarrow ($
            $(\mathbf{t}_8 \le \mathbf{t}'_8) \wedge$
            $(\mathbf{t}_8 = \mathbf{t}'_8)\ \Rightarrow ($
                $(\mathbf{t}_7 \le \mathbf{t}'_7) \wedge$
                $\vdots$
                $(\mathbf{t}_2 = \mathbf{t}'_2\ \Rightarrow\ \mathbf{t}_1 \le \mathbf{t}'_1)$
                $\vdots$
            $)$
        $)$
    $)\ \wedge$
    $(\text{TestBit}[\mathsf{n},\mathsf{m},\mathsf{k},\mathbf{t}_1,\dots,\mathbf{t}_8])$
$)$
```

The formula needs the sampling limit $L = 3 \cdot 10^{16} m^{13} n^6$ from Section 3.2. Since L is polynomial in m and n a long variable $\mathbf{L}$ has to be used to represent the value. Then, according to the Lattice Property 3.15 and Corollary 4.5 sampling all integer tuples within the limits L enables ImageMatch to specify all projective transformations needed to compute the dictionary $\mathcal{D}[m](A)$ of image A . The matching against image B, however, is described in another sub formula TestMatch. The correct minimum tuple is found by testing every candidate against all competitor tuples. Finally, sub formula TestBit is used to describe the output string of $z = \text{IM}^u_\omega(s')$ by the values of $\mathbf{t}_1, \dots, \mathbf{t}_8$ and to test whether $z(k) = 1$.

First consider the sub formula TestMatch$[\mathsf{n}, \mathsf{m}, \mathbf{L}, \mathbf{t}_1, \ldots, \mathbf{t}_8, \mathbf{T}]$. The use of TestMatch is to represent the image $f(A)[m]$ that is given by the inverse projective transformation $f^{-1} = \mathcal{P}(P)$ with $P = \mathcal{L}[m,n](a(\mathbf{t}_1), \ldots, a(\mathbf{t}_8))$. The main challenge realized in TestMatch is to measure $\delta(f(A)[m], B)$. Hence, for every pair (s, a) with string $s = enc^u(n)|enc^u(m)|enc^u(A)|enc^u(B)|1^k$ and assignment a with $a(\mathsf{n}) = n$, $a(\mathsf{m}) = m$, $a(\mathbf{L}) = 3 \cdot 10^{16} m^{13} n^6$ and $a(\mathbf{t}_r) \in \{-L, \ldots, L\}$ for all $r \in \{1, \ldots, 8\}$ the formula fulfills

$$\begin{aligned}&(s,a) \models \text{TestMatch}[\mathsf{n}, \mathsf{m}, \mathbf{L}, \mathbf{t}_1, \ldots, \mathbf{t}_8, \mathbf{T}] \iff \\ &\delta(f(A)[m], B) = a(\mathbf{T}) \text{ with } f^{-1} = \mathcal{P}\left(\mathcal{L}[m,n](a(\mathbf{t}_1), \ldots, a(\mathbf{t}_8))\right).\end{aligned}$$

To measure $\delta(f(A)[m], B)$ means to compute $f(A)[m](i,j)$ for all $(i,j) \in z(m)$ and then to determine $\sum_{(i,j)\in z(m)} \delta(f(A)[m](i,j), B(i,j))$. It is not possible to evaluate the sum simply by a repeated application of Plus because then ϕ would not remain of constant length. The solution to this issue is the only point that actually needs the application of the majority quantifier.

Expressing the sum of integers as a first order formula with majority quantification is still an involved task. Note first that for all $(i,j) \in z(m)$ it is true that $\delta(f(A)[m](i,j), B(i,j))$ is a natural number between zero and a constant $\Upsilon \in \mathbb{N}$. Hence, all possible outcomes can be represented using $\upsilon = \lceil \log_2 \Upsilon \rceil$ bits. The sum can be reformulated as follows:

$$\begin{aligned}\delta(f(A)[m], B)) &= \sum_{(i,j)\in z(m)} \delta(f(A)[m](i,j), B(i,j)) \\ &= \sum_{(i,j)\in z(m)} 2^0 bit(\delta(f(A)[m](i,j), B(i,j)), 0) + \ldots + \\ &\quad 2^{\upsilon-1} bit(\delta(f(A)[m](i,j), B(i,j)), \upsilon - 1) \\ &= \left(2^0 \cdot \sum_{(i,j)\in z(m)} bit(\delta(f(A)[m](i,j), B(i,j)), 0)\right) + \ldots + \\ &\quad \left(2^{\upsilon-1} \cdot \sum_{(i,j)\in z(m)} bit(\delta(f(A)[m](i,j), B(i,j)), \upsilon - 1)\right),\end{aligned}$$

where $bit(w, r)$ gives the rth bit in the binary representation of w for all $w, r \in \mathbb{N}$. The advantage of this reformulation is that for all $r \in \{0, \ldots, \upsilon - 1\}$ the sum $z_r = \sum_{(i,j)\in z(m)} bit(\delta(f(A)[m](i,j), B(i,j)), r)$ can be interpreted as counting positive bits. Moreover, z_r is a value in $\{0, \ldots, (2m+1)^2\}$ which can be easily represented by a basic variable. The value $\delta(f(A)[m], B))$, however, needs the long variable $\mathbf{T}$. Nevertheless, $\delta(f(A)[m], B))$ can be obtained from a constant number of arithmetic operations on the numbers z_0 to $z_{\upsilon-1}$. This implies that TestMatch can be realized as follows:

```
TestMatch$[\mathsf{n},\mathsf{m},\mathbf{L},\mathbf{t}_1,\ldots,\mathbf{t}_8,\mathbf{T}] = \exists \mathsf{z}_0 \ \ldots \ \exists \mathsf{z}_{\upsilon-1}$ (
        $(\mathrm{Count}_0[\mathsf{n},\mathsf{m},\mathbf{L},\mathbf{t}_1,\ldots,\mathbf{t}_8,\mathsf{z}_0]) \wedge$
        $\vdots$
        $(\mathrm{Count}_{\upsilon-1}[\mathsf{n},\mathsf{m},\mathbf{L},\mathbf{t}_1,\ldots,\mathbf{t}_8,\mathsf{z}_{\upsilon-1}]) \wedge$
        $\left(\mathbf{T} = 2^0 \cdot \mathsf{z}_0 + \ldots + 2^{\upsilon-1} \cdot \mathsf{z}_{\upsilon-1}\right)$
)
```

For all $r \in \{0,\ldots,\upsilon-1\}$ the function of sub formula $\mathrm{Count}_r[\mathsf{n},\mathsf{m},\mathbf{L},\mathbf{t}_1,\ldots,\mathbf{t}_8,\mathsf{z}_r]$ is to represent the value z_r, i. e., for every pair (s,a) with string $s = enc^u(n)|enc^u(m)|enc^u(A)|enc^u(B)|1^k$ and assignment a with $a(\mathsf{n}) = n$, $a(\mathsf{m}) = m$, $a(\mathbf{L}) = 3 \cdot 10^{16} m^{13} n^6$ and $a(\mathbf{t}_1),\ldots,a(\mathbf{t}_1) \in \{-L,\ldots,L\}$ it is true

$$(s,a) \models \mathrm{Count}_r[\mathsf{n},\mathsf{m},\mathbf{L},\mathbf{t}_1,\ldots,\mathbf{t}_8,\mathsf{z}_r] \quad \Longleftrightarrow$$
$$a(\mathsf{z}_r) = \sum_{(i,j)\in z(m)} bit(\delta(f(A)[m](i,j),B(i,j)),r) \text{ with } f^{-1} = \mathcal{P}\left(\mathcal{L}[m,n](a(\mathbf{t}_1),\ldots)\right).$$

The counting of single bits can be realized with majority quantification, particularly, in the form of the convenient counting quantifier. For all $r \in \{0,\ldots,\upsilon-1\}$ the sub formula is defined as follows:

```
$\mathrm{Count}_r[\mathsf{n},\mathsf{m},\mathbf{L},\mathbf{t}_1,\ldots,\mathbf{t}_8,\mathsf{z}_r] = Q^{\mathsf{z}_r}_{|z(m)|}\mathsf{x} \ \exists \mathbf{i} \ \exists \mathbf{j} \ \exists \mathsf{a} \ \exists \mathsf{b} \ \exists \mathbf{c}$ (
        $(\mathbf{i} = (\mathsf{x} \mod (2\mathsf{m}+1)) - \mathsf{m}) \wedge$
        $\left(\mathbf{j} = \left\lfloor \frac{\mathsf{x}}{2\mathsf{m}+1} \right\rfloor - \mathsf{m}\right) \wedge$
        $(\mathrm{ImageATrans}[\mathsf{n},\mathsf{m},\mathbf{L},\mathbf{t}_1,\ldots,\mathbf{t}_8,\mathbf{i},\mathbf{j},\mathsf{a}]) \wedge$
        $(\mathrm{ImageB}[\mathsf{n},\mathsf{m},\mathbf{i},\mathbf{j},\mathsf{b}]) \wedge$
        $(\omega[\mathsf{a},\mathsf{b},\mathbf{c}]) \wedge$
        $(\mathrm{Bit}[\mathbf{c},r])$
)
```

Since $2z_r \leq 2(2m+1)^2 = 2|z(m)| \leq |s|$ the counting quantifier is used correctly according to Theorem 5.13. The idea for Count_r is to accept if and only if the rth bit in the value of the distortion is positive for exactly z_r times.

To describe the distortion the formula needs access to the color values of the transformation of image A. This is realized by the sub formula ImageATrans. The sub formula $\omega[\mathsf{a},\mathsf{b},\mathbf{c}]$ accepts a pair (s,a) if and only if $\omega(a(\mathsf{a}),a(\mathsf{b})) = a(\mathbf{c})$. Since $\omega : \mathcal{C} \times \mathcal{C} \to \mathbb{N}$ maps from a finite domain, it is obvious that $\omega[\mathsf{a},\mathsf{b},\mathbf{c}]$ can be expressed as a first order formula.

The sub formula ImageATrans fulfills for every pair (s,a) with string $s = enc^u(n)|enc^u(m)|enc^u(A)|enc^u(B)|1^k$ and assignment a with $a(\mathsf{n}) = n$, $a(\mathsf{m}) = m$,

$a(\mathsf{L}) = 3 \cdot 10^{16} m^{13} n^6$ and $a(\mathsf{t}_1) \dots a(\mathsf{t}_8) \in \{-L, \dots, L\}$ that

$$(s,a) \models \mathrm{ImageATrans}[\mathsf{n}, \mathsf{m}, \mathsf{L}, \mathsf{t}_1, \dots, \mathsf{t}_8, \mathsf{i}, \mathsf{j}, \mathsf{c}] \iff$$
$$A([f^{-1}(a(\mathsf{i}), a(\mathsf{j}))]) = a(\mathsf{c}) \text{ with } f^{-1} = \mathcal{P}\left(\mathcal{L}[m,n](a(\mathsf{t}_1), \dots, a(\mathsf{t}_8))\right).$$

Hence, the formula provides access to the color value $A([f^{-1}(a(\mathsf{i}), a(\mathsf{j}))])$ of image A determined by the inverse transformation f^{-1} of the lattice point $\mathcal{L}[m,n](a(\mathsf{t}_1), \dots, a(\mathsf{t}_8))$. This functionality is realized as follows:

$$\begin{aligned}
&\mathrm{ImageATrans}[\mathsf{n}, \mathsf{m}, \mathsf{L}, \mathsf{t}_1, \dots, \mathsf{t}_8, \mathsf{i}, \mathsf{j}, \mathsf{c}] = \exists \mathsf{u}\ \exists \mathsf{x}\ \exists \mathsf{y}\ \exists \mathsf{z}\ \exists \mathsf{i}'\ \exists \mathsf{j}'\ \Big(\\
&\qquad \big(\mathsf{u} = 12 \cdot 10^9 \mathsf{m}^7 \mathsf{n}^3 \cdot \mathsf{L}^2 + 4 \cdot \mathsf{L}^2\big) \wedge \\
&\qquad \left(\begin{pmatrix} \mathsf{x} \\ \mathsf{y} \\ \mathsf{z} \end{pmatrix} = \begin{pmatrix} 8\mathsf{t}_1\mathsf{L}^2 + 1 & 8\mathsf{t}_2\mathsf{L}^2 & 8\mathsf{t}_3\mathsf{L}^2 \\ 8\mathsf{t}_4\mathsf{L}^2 & 8\mathsf{t}_5\mathsf{L}^2 + 4\mathsf{L}^2 & 8\mathsf{t}_6\mathsf{L}^2 \\ 8\mathsf{t}_7\mathsf{L}^2 & 8\mathsf{t}_8\mathsf{L}^2 & \mathsf{u} \end{pmatrix} \cdot \begin{pmatrix} \mathsf{i} \\ \mathsf{j} \\ 1 \end{pmatrix} \right) \wedge \\
&\qquad \big(\mathsf{i}' = \left[\tfrac{\mathsf{x}}{\mathsf{z}}\right]\big) \wedge \\
&\qquad \big(\mathsf{j}' = \left[\tfrac{\mathsf{y}}{\mathsf{z}}\right]\big) \wedge \\
&\qquad (\mathrm{ImageA}[\mathsf{n}, \mathsf{m}, \mathsf{i}', \mathsf{j}', \mathsf{c}]) \\
&\Big)
\end{aligned}$$

The sub formula prepares the projective matrix of f^{-1} and then transforms the point $(a(\mathsf{i}), a(\mathsf{j}))$ to obtain the corresponding index $(a(\mathsf{i}'), a(\mathsf{j}')) = [f^{-1}(a(\mathsf{i}), a(\mathsf{j}))]$. To reduce the matrix multiplication to integer arithmetic the whole projective matrix is multiplied with the factor $8L^2d^{-1}$ for the value $d = (1.5 \cdot 10^9 m^7 n^3 + 0.5)^{-1}$ defined in Section 3.2. Due to the use of homogeneous coordinates, however, this factor is eliminated by the division described in $\mathsf{i}' = \left[\frac{\mathsf{x}}{\mathsf{z}}\right]$ and $\mathsf{j}' = \left[\frac{\mathsf{y}}{\mathsf{z}}\right]$. Nevertheless, all values are polynomial in n and m and thus, can be represented by long variables. Notice that $a(\mathsf{z})$ is never zero which follows from the Lattice Property 3.15. Finally, sub formula ImageA is applied to find the color value $A(a(\mathsf{i}'), a(\mathsf{j}'))$.

It remains to specify the sub formula TestBit used in ImageMatch. The function of the formula is to accept every pair (s,a) with an assignment a having $a(\mathsf{n}) = n$, $a(\mathsf{m}) = m$ and $a(\mathsf{k}) = k$ if and only if the string $z = enc^b(a(\mathsf{t}_1)) \mid \dots \mid enc^b(a(\mathsf{t}_8)) \in \Pi_{out}$ fulfills $z(k) = 1$. Clearly, z is the concatenation of eight binary two's complement numbers each of $70\lceil \log_2 mn \rceil$ bits. It is easy to see that $70\lceil \log_2 mn \rceil$ bits are enough to encode a number in $\{-L, \dots, L\}$ and thus, z has $560\lceil \log_2 mn \rceil$ bits. The formula first determines $\lceil \log_2 mn \rceil$ as the value of log and then the index $r \in \{1, \dots, 8\}$ of the parameter with the encoding $enc^b(a(\mathsf{t}_r))$ containing the bit k. Afterwards the encoding of this parameter is described and the bit k is queried:

$$\begin{aligned}
&\mathrm{TestBit}[\mathsf{n}, \mathsf{m}, \mathsf{k}, \mathsf{t}_1, \dots, \mathsf{t}_8] = \exists \mathsf{log}\ \exists \mathsf{p}\ \exists \mathsf{k}'\ \exists \mathsf{e}\ \Big(\\
&\qquad (2^{\mathsf{log}} \geq \mathsf{mn}) \wedge \\
&\qquad \big(2^{\mathsf{log}-1} < \mathsf{mn}\big) \wedge
\end{aligned}$$

```
        $(\mathsf{k} < 560 \cdot \mathsf{log}) \wedge$
        $\big(\mathbf{p} = 2^{70 \cdot \mathsf{log}}\big) \wedge$
        $(0 \leq \mathsf{k} < 70\mathsf{log}) \ \Rightarrow \big($
            $(\mathsf{k}' = \mathsf{k}) \wedge$
            $(\mathbf{t}_1 \geq 0 \ \Rightarrow \ \mathbf{e} = \mathbf{t}_1) \wedge$
            $(\mathbf{t}_1 < 0 \ \Rightarrow \ \mathbf{e} = \mathbf{p} - \mathbf{t}_1) \wedge$
            $(\mathrm{Bit}[\mathbf{e}, \mathsf{k}'])$
        $\big) \wedge$
        $\vdots$
        $(490 \cdot \mathsf{log} \leq \mathsf{k} < 560 \cdot \mathsf{log}) \ \Rightarrow \big($
            $(\mathsf{k}' = \mathsf{k} - 490 \cdot \mathsf{log}) \wedge$
            $(\mathbf{t}_8 \geq 0 \ \Rightarrow \ \mathbf{e} = \mathbf{t}_8) \wedge$
            $(\mathbf{t}_8 < 0 \ \Rightarrow \ \mathbf{e} = \mathbf{p} - \mathbf{t}_8) \wedge$
            $(\mathrm{Bit}[\mathbf{e}, \mathsf{k}'])$
        $\big)$
    $)$
```

This completes the proof. □

5.2.2 Encoding Color Values in Binary

That the unary decision version of image matching is in U_D-TC_O implies a corresponding DLOGTIME-uniform family $\mathcal{N}$ of constant depth polynomial size circuits using threshold gates. However, natural image matching instances have binary encoded color values. Consequently, this section provides translation circuits to convert binary representations of images to unary encodings to make the natural problem solvable by the help of $\mathcal{N}$.

The encoding of n and m is kept unary because their lengths are negligible with respect to the image sizes. For the encoding of color values the following presents the same decision problem building on a binary encoding scheme:

Definition 5.17. *Let $\Pi_{in} \subseteq \{0,1\}^*$ be the set of binary strings*

$$s = enc^u(n) \mid enc^u(m) \mid enc^b(A) \mid enc^b(B)$$

encoding two digital images A and B by

- *$enc^u(n) = 1^n0$ and $enc^u(n) = 1^m0$, giving the supports $z(n)$ and $z(m)$ of A and B and*
- *$enc^b(A)$ and $enc^b(B)$, the encoding of color information as Little Endian*

binary representation of the two numbers

$$\sum_{j=0}^{2n}\sum_{i=0}^{2n} 2^{\lceil \log_2 C\rceil \cdot (j(2n+1)+i)} A(i-n, j-n) \text{ and}$$
$$\sum_{j=0}^{2m}\sum_{i=0}^{2m} 2^{\lceil \log_2 C\rceil \cdot (j(2m+1)+i)} B(i-m, j-m)$$

using $\lceil \log_2 C\rceil \cdot (2n+1)^2$*, respectively* $\lceil \log_2 C\rceil \cdot (2m+1)^2$*, bits and consisting of* $\lceil \log_2 C\rceil$ *bit blocks each encoding a color of* $\mathcal{C}$*.*

Based on Π_{in} *and* Π_{out} *and for any given distortion measure* $\omega : \mathcal{C}\times\mathcal{C} \to \mathbb{N}$ *define the function* $IM_\omega : \Pi_{in} \to \Pi_{out}$*, where for all* $s \in \Pi_{in}$ *the string* $z = IM_\omega(s)$ *encodes eight integers* $t_1, \ldots, t_8$ *which primary define* $f^{-1} = \mathcal{P}\left(\mathcal{L}[m,n](t_1,\ldots,t_8)\right)$ *that minimizes* $\delta(f(A)[m], B)$ *for the images* A *and* B *encoded by* s *and secondary minimize the value*

$$\sum_{i=1}^{8}(t_i + L)(2L+1)^{i-1}.$$

This leads to the following decision problem:

Problem 5.18. Projective Image Matching kth Bit

Input: *A string* $s \in \{0,1\}^*$.

Constraints:
Image distortion measure $\omega : \mathcal{C}\times\mathcal{C} \to \mathbb{N}$.

Question: *Is string* s *of the form* $s'|enc^b(k)$ *with* $s' \in \Pi_{in}$ *and* $k \in \mathbb{N}$ *encoded as a binary number* $enc^b(k)$ *such that the* k*th bit in string* $z = IM_\omega(s')$ *is positive?*

The following lemma states that even this aggravated version of the decision problem is in U_D-TC_O:

Lemma 5.19. *For any distortion measure* $\omega : \mathcal{C}\times\mathcal{C} \to \mathbb{N}$ *Problem 5.18 is in* U_D-TC_O*.*

Proof. By Lemma 5.16 it is known that the image matching problem can be decided in U_D-TC_O if the color values of A and B and the integer k are encoded unary. Consequently, there is a DLOGTIME-uniform family $\mathcal{N}$ of constant-depth, polynomial-size threshold circuits deciding Problem 5.15. Because U_D-FTC_O is transitive a simple way to prove the statement is to provide a translation DLOGTIME-uniform family $\mathcal{N}'$ of constant-depth, polynomial-size threshold circuits that transforms input strings s, encoding color values of A and B as well as the integer k in binary, into their unary counterparts, denoted here by $enc^u(s)$. Afterwards the application of $\mathcal{N}$ on $enc^u(s)$ provides the correct answer also for s.

The translation circuit $\mathcal{N}'$ works on every part of s individually. For the representations of n and m the translation simply copies input bits to the output. Translating the images $enc^b(A)$ and $enc^b(B)$ in s to their unary representations, $enc^u(A)$ and $enc^u(B)$, can be done easily by $\mathcal{N}'$, too. Both, binary and unary representation, store a sequence of color values, each of constant length, either $\lceil \log_2 C \rceil$ or C bits. For every color in the sequence the translation works by simply using the binary value to index the positive bit in the unary representation.

For $enc^b(k)$ however, the growth of $enc^u(k)$ may be exponential and by that infeasible. Anyway, if the absolute value of k is less than $560\lceil \log_2 mn \rceil$, then $enc^u(k)$ contains only a polynomial number of bits with respect to $|s|$. Thus, it can be translated by $\mathcal{N}'$. Otherwise, if $|k| \geq 560\lceil \log_2 mn \rceil$, then s does not belong to the decision Problem 5.18 independent of the actual contents of A and B. Consequently, in this case the value of k is simply translated to the unary representation of $560\lceil \log_2 mn \rceil$. □

5.2.3 Solving Image Matching in U_D-FTC_O

Based on Lemma 5.19 it is possible to state a complexity result for the optimization Problem 2.13:

Theorem 5.20. *The optimization version of the image matching Problem 2.13 for projective transformations is in U_D-FTC_O.*

Proof. A precise definition of the optimization problem's input and output can be obtained by considering the sets Π_{in} and Π_{out}. Then the solution for a given string $s \in \Pi_{in}$ is given by the string $z = \mathrm{IM}_\omega(s)$.

By Lemma 5.19 it is known that the image matching Problem 5.18 can be decided in U_D-TC_O if A, B and k are naturally encoded in binary form. Consequently, there is a DLOGTIME-uniform family $\mathcal{N}$ of constant-depth, polynomial-size threshold circuits deciding Problem 5.18.

Using $\mathcal{N}$, the image matching solution for a given string $s \in \Pi_{in}$ can be found by computing every bit k of $z = \mathrm{IM}_\omega(s)$ by the help of the circuit $N_{|s|+\lceil \log_2 k \rceil} \in \mathcal{N}$ deciding the string $s|enc^b(k)$. This means, a family $\mathcal{N}'$ of constant-depth, polynomial-size threshold circuits computing IM_ω is found by combining suitable circuits from $\mathcal{N}$.

Moreover, $\mathcal{N}'$ is DLOGTIME-uniform since every circuit $N'_{|s|} \in \mathcal{N}'$ can be verified by a $O(\log |s|)$ time Turing machine $M_{\mathcal{N}'}$ that uses $M_{\mathcal{N}}$ as a subroutine. In particular, if a circuit $N'_{|s|} \in \mathcal{N}'$ consists of the sub circuits $N_{r_1}, \ldots, N_{r_k} \in \mathcal{N}$ with $k = 560\lceil \log_2 mn \rceil$ then $M_{\mathcal{N}'}$ works as follows: It first detects the circuit N_{r_i} that is affected by the query. This is possible in $O(\log |s|)$ time since $|s| \leq r_i \leq |s| + 560\lceil \log_2 mn \rceil \leq 2|s|$. Then it starts $M_{\mathcal{N}'}$ as a subroutine with the query translated for N_{r_i} which finishes in $O(\log r_i) = O(\log |s|)$ time. □

Since the optimization Problem 2.13 is at least as hard as the decision Problem 2.14 the containment in U_D-TC_O follows also for this problem:

Corollary 5.21. *The decision Problem 2.14 for projective image matching is in* U_D-TC_O.

Proof. Again a precise definition of input can be obtained by considering the set Π_{in}. Particularly, for any distortion measure $\omega : \mathcal{C} \times \mathcal{C} \to \mathbb{N}$ let Π^t_ω be the set of strings s of the form $s = s'|enc^b(T)$ with $s' \in \Pi_{in}$ and $T \in \mathbb{N}$ such that the string $z = \mathrm{IM}_\omega(s')$ encodes $(t_1, \dots, t_8)$ that define an inverse projective transformation $f^{-1} = \mathcal{P}(t_1, \dots, t_8)$ with $\delta(f(A)[m], B) \leq T$ where A and B are the images of size n, respectively m, encoded by s'. The rest of the proof shows that Π^t_ω can be decided in U_D-TC_O.

Firstly, by the existence of sub formula DecodeParam[$\mathsf{n}, \mathsf{m}, \mathsf{T}$] described in the proof of Lemma 5.16 and the binary-to-unary translation described in the proof of Lemma 5.19 it is possible to decode a given string s to $s' \in \Pi_{in}$ and $T \in \mathbb{N}$ by a DLOGTIME-uniform family $\mathcal{N}_1$ of constant-depth, polynomial-size threshold circuits. Moreover, according to Theorem 5.20 there is a DLOGTIME-uniform family $\mathcal{N}_2$ of constant-depth, polynomial-size threshold circuits that is able to compute the string $z = \mathrm{IM}_\omega(s')$. Then, by the sub formula TestBit there is a threshold circuit family $\mathcal{N}_3$ to decode z into transformation parameters $t_1, \dots, t_8$ simply by taking the two's complement. And finally, sub formula TestMatch enables a threshold circuit family $\mathcal{N}_4$ that computes $T' = \delta(f(A)[m], B)$ by $f^{-1} = \mathcal{P}(t_1, \dots, t_8)$. It is straight that the test of $T' \leq T$ can be done by a very simple circuit family $\mathcal{N}_5$.

By a similar argumentation as in the proof of Theorem 5.20 a DLOGTIME-uniform family $\mathcal{N}$ of constant-depth, polynomial-size threshold circuits deciding Π^t_ω can be obtained by a series connection of $\mathcal{N}_1$ to $\mathcal{N}_5$. $\square$

The technique presented in this section can be easily transferred to the case of affine transformations as shown by the author in [21]. For linear transformations, rotations, scalings and combinations of scaling and rotation the case is more complicated. Although proper subclasses, their structure is geometrically harder than in the projective case. This results basically from the fact that the arrangements $\mathcal{H}_\chi[m, n]$ under $\chi \in \{\chi_{\mathtt{l}}, \chi_{\mathtt{r}}, \chi_{\mathtt{s}}, \chi_{\mathtt{sr}}\}$ contain faces that have no volume. Consequently, it is likely that they are missed during a sampling process as applied in this section. Although the author believes that image matching under each of these classes can be done in TC_O, it remains open whether this is true.

5.3 Completeness in TC_O

This section shows that the decision Problem 2.14 is complete in U_D-TC_O. Recall the U_D-TC_O-complete majority problem, i. e., the set $MAJ \subseteq \{0, 1\}^*$ of strings which contain at least $\lceil 0.5|s| \rceil$ positive bits. This section gives a U_D-FAC_O-reduction r of MAJ to the set of strings s encoding two images A, B over the color set $\mathcal{C} = \{0, \dots, C-1\}$ and $\mathcal{T}$ such that the minimum $\omega(f(A)[m], B)$ over all projective transformations f is at most $\mathcal{T}$. Hence, $r : \{0,1\}^* \to \{0,1\}^*$ is a function which maps strings $s \in \{0,1\}^*$ to a binary encoding of images A and B and an integer $\mathcal{T}$ such that $s \in MAJ$ if and only if there is a transformation $f \in \mathcal{F}_{\mathrm{p}}$ such that $\delta(f(A)[m], B) \leq \mathcal{T}$.

The foundation of the reduction is given by the following observation: Consider any string $s \in \{0,1\}^*$. Then define images A_s and B_s, both of size $n = m = \max\{30|s|, 60\}$, where $A_s(i,j) = B_s(i,j) = 3$ for all indices $(i,j) \in z(n)$ with $j \neq 0$. Additionally set

$$A_s(i,0) = \begin{cases} s(i), & \text{if } 0 \leq i < |s| \\ 2, & \text{otherwise} \end{cases} \quad \text{and} \quad B_s(i,0) = \begin{cases} 1, & \text{if } 0 \leq i < |s| \\ 2, & \text{otherwise} \end{cases}$$

for all $i \in \{-n, \dots, n\}$ and let $\mathcal{T}_s = \lfloor 0.5|s| \rfloor$. Obviously, A_s represents a copy of s and B_s a row of positive bits. Moreover $\mathcal{T}_s$ describes the maximum number of zero bits that s can contain to be still in MAJ. Then the majority of bits in s is positive if and only if $\delta(A_s, B_s) \leq \mathcal{T}_s$ for the distortion measure $\omega(a,b) = |a-b|$. Hence, if transformations were not allowed on A_s this approach would already be successful.

However, image matching allows any projective transformation to be applied on A_s before evaluating the distortion against B_s. This means that there may be transformations $f \in \mathcal{F}_{\mathrm{p}}$ with $\delta(f(A_s)[n], B_s) < \delta(A_s, B_s)$. Hence, f may destroy the synchronization between A_s and B_s and then the distortion value between $f(A_s)[n]$ and B_s does not necessarily allow inference on the containment of string s in MAJ. Particularly, this happens if the distortion does not count the number of zero bits in s. For example, if f describes a vertical shifting of A_s by at least one pixel then it happens that the synchronization of the centerlines in $f(A)[n]$ and B_s is lost and consequently, their mutual distortion allows no statement about $s \in MAJ$.

To still use a similar approach A_s and B_s are extended in such a way that the transformations that lead to the optimal match are close to identity and thus, still count the number of zeros in s. For this end, in A_s and B_s as specified before define now for all $k \in \{-n+14, \dots, n-14\}$

$$A_s(k, -n+13) = 4, A_s(k, n-13) = 5, A_s(-n+13, k) = 6, A_s(n-13, k) = 7$$

and for all $k \in \{-n-1, \dots, n+1\}$

$$B_s(k, -n) = 4, B_s(k, n) = 5, B_s(-n, k) = 6, B_s(n, k) = 7,$$

i. e., draw a multiple colored frame in A_s and a little bigger frame in B_s. An impression of both images is given in Figure 5.1. Clearly, A_s and B_s contain only $C = 8$ different colors, i.e, the cardinality of $\mathcal{C} = \{0, 1, \dots, 7\}$ is a power of two. Consider the following lemma:

Lemma 5.22 (Proof in Section 5.4)**.** *Let $s \in \{0,1\}^*$ and A_s and B_s be as defined above. Moreover, consider the projective transformation f_s with the following projection matrix M_s:*

$$M_s = \begin{pmatrix} v & 0 & 0 \\ 0 & v & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

where $v = \frac{n}{n-13}$. If $\omega(a,b) = |a-b|$ then $\delta(f_s(A_s)[n], B_s) = \sum_{k=0}^{|s|-1} 1 - s(i)$.

The transformation f_s guarantees that (1) the representation of string s remains unaltered and (2) the rest of $f_s(A_s)[n]$ looks like B_s. This means $\delta(f_s(A_s)[n], B_s)$ equals the number of zero bits in s. Then s contains a majority of positive bits if and only if $\delta(f_s(A_s)[n], B_s) \leq \mathcal{T}_s$.

A big challenge is to show that the transformation f_s is indeed the best one, i. e., that there is no transformation $f \in \mathcal{F}_{\mathrm{P}}$ that performs a better match of $f(A_s)[n]$ to B_s. Otherwise the solution of image matching would rather go with f than with f_s.

Lemma 5.23 (Proof in Section 5.4)**.** *For all $s \in \{0,1\}^*$ it is true that $\delta(f_s(A_s)[n], B_s)$ is minimum over all projective transformations in $\mathcal{F}_{\mathrm{P}}$ if $\omega(a,b) = |a-b|$.*

The basic idea behind the lemma's proof is that the frames in $f(A_s)[n]$ and B_s have to be aligned if f performs a good match of $f(A_s)[n]$ against B_s. Together with the necessary alignment of the row $j = 0$ this enforces f to be very similar to f_s. The smaller frame in A_s guarantees that f has to scale A_s up to match B_s's frame. This results in the effect that every index $(I, 0)$ in the center line of A_s is represented by at least one index $(i, 0)$ in $f(A_s)[n]$, i. e., $(I, 0) = [f^{-1}(i, 0)]$. Consequently, since all zero bits in s are represented in the center line of A_s they are copied to $f(A_s)[n]$. Thus, the distortion between $f(A_s)[n]$ and B_s counts all zero bits of string s. The concept behind the two lemma is presented in Figure 5.1.

The rest is to argue that for any string $s \in \{0,1\}^*$ the computation of $r(s)$, providing a string that encodes A_s, B_s and the threshold $\mathcal{T}_s$, can be accomplished in U_D-FAC_O. However, the hardest part of computing A_s and B_s is copying and filling in constants, namely, inserting the string s in A_s and preparing the frames in both images.

The proof of $r \in U_D$-FAC_O is again provided by expressing r as a first order sentence. This means that again r is described as a decision problem, just like the function IM_ω in the previous section.

Definition 5.24. *Let $\Pi_{out} \subseteq \{0,1\}^*$ be the set of binary strings*

$$z = enc^u(n) \mid enc^u(m) \mid enc^b(A) \mid enc^b(B) \mid enc^b(\mathcal{T})$$

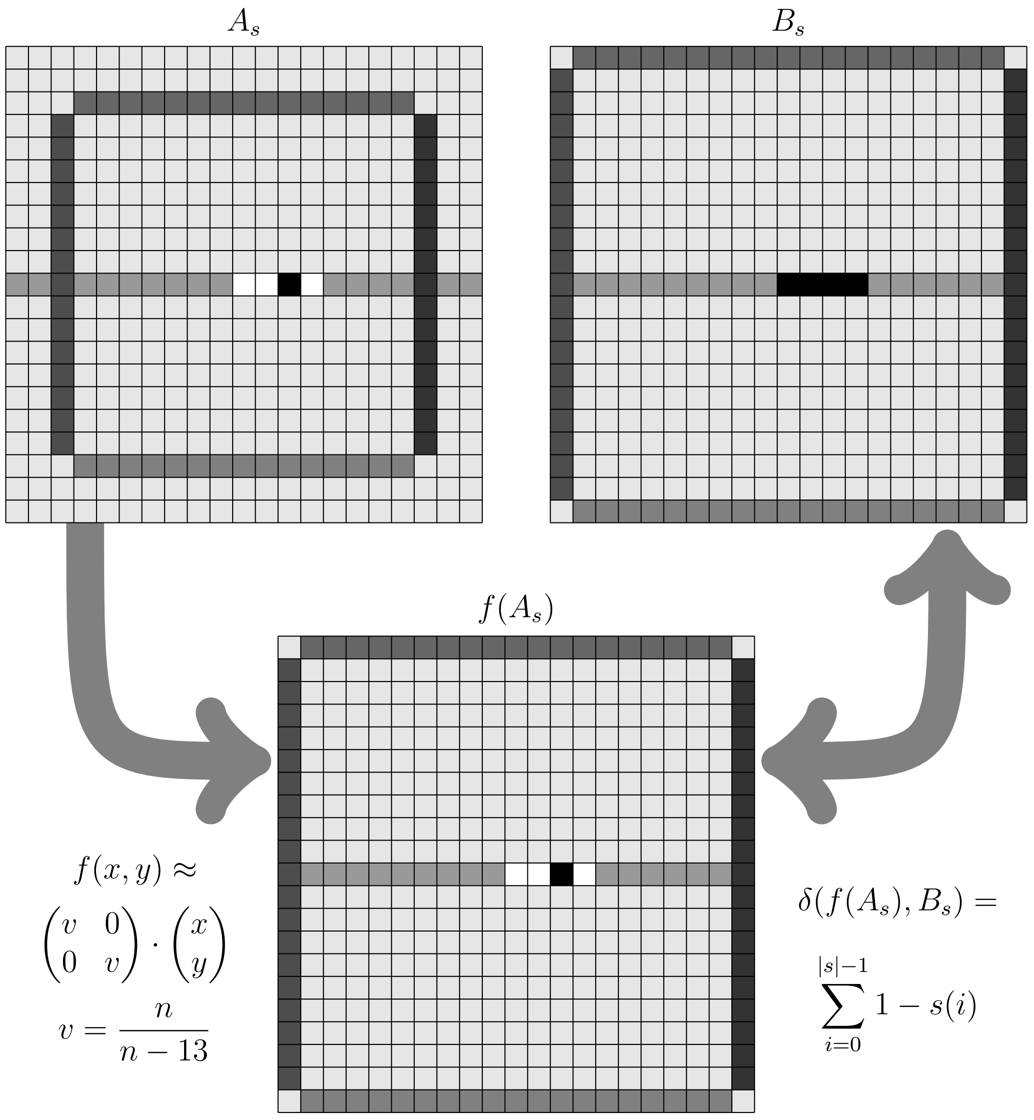

A_s
B_s
$f(A_s)$
$f(x,y) \approx \begin{pmatrix} v & 0 \\ 0 & v \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}$
$v = \frac{n}{n-13}$
$\delta(f(A_s), B_s) = \sum_{i=0}^{|s|-1} 1 - s(i)$

Figure 5.1: Images A_s and B_s for $s = 0010$. For perceivability, size and frame padding have been reduced. Every optimal f has to be nearly a v-scaling and consequently, distortion counts zeros in s.

encoding two digital images A and B of size n, respectively m, exactly like Π_{in} in Section 5.2 and an integer $\mathcal{T} \in \mathbb{N}$.

Then $r : \{0,1\}^* \to \Pi_{out}$ is the function that maps every string s to the string $z = r(s)$ encoding the images A_s and B_s of size $n = m = \max\{30|s|, 60\}$ described above and the integer $\mathcal{T}_s = \lfloor 0.5|s| \rfloor$.

According to this consider the following decision problem:

Problem 5.25. Majority to Projective Image Matching kth Bit

Input: *A string* $s \in \{0,1\}^*$.

Question: *Is string* s *of the form* $1^k 0 s'$ *with* $k \in \mathbb{N}$ *such that the* k*th bit in string* $z = r(s')$ *is positive?*

To show the existence of a DLOGTIME-uniform family $\mathcal{N}$ of constant depth polynomial size circuits this section develops a first order sentence for Problem 5.25. This implies that the function r is in U_D-FAC_O by combining circuits of $\mathcal{N}$ to compute every output bit.

Lemma 5.26. *Problem 5.25 can be expressed as a first order sentence* ϕ*. Thus, the problem is in* U_D-AC_O*.*

Proof. The sentence ϕ has a simply structure. First it makes sure that every model s is of the form $s = 1^k 0 s'$ with $k \in \mathbb{N}$. Then the sentence binds the value k to the variable k.

Afterwards ϕ consists of sub formulas to describe $enc^u(n)$, $enc^u(m)$, $enc^b(A_s)$, $enc^b(B_s)$ and $enc^b(\mathcal{T}_s)$ for $n = m = \max\{30|s|, 60\}$, the images A_s and B_s of size n and the integer $\mathcal{T}_s = \lfloor 0.5|s| \rfloor$. Based on the value of k, sentence ϕ knows the starting position of s' and also its length. Then the truth value of ϕ depends only on the sub formula responsible for the part of $r(s')$ which contains the kth bit:

$$
\begin{aligned}
\phi = \exists \mathsf{k}\ \exists \mathbf{n}\ \exists \mathcal{T}\ \exists \mathsf{log}\ \exists \mathbf{u}\ \exists \mathbf{v}\ \exists \mathbf{w}\ \Big(&\\
&(\mathrm{DecodeParam}[\mathsf{k}]) \wedge\\
&(\mathbf{n} = \max\{30((|s|-1)-\mathsf{k}), 60\}) \wedge\\
&(\mathcal{T} = \lfloor 0.5((|s|-1)-\mathsf{k}) \rfloor) \wedge\\
&(2^{\mathsf{log}} \geq \mathcal{T}) \wedge\\
&(2^{\mathsf{log}\text{-}1} < \mathcal{T}) \wedge\\
&\big(\mathsf{k} \leq 2\mathbf{n} + 2 + 6 \cdot (2\mathbf{n}+1)^2 + \mathsf{log}\big) \wedge\\
&(0 \leq \mathsf{k} \leq \mathbf{n} \ \Rightarrow\ \mathrm{EncodeN}[\mathsf{k}, \mathbf{n}]) \wedge\\
&(\mathbf{u} = 2\mathbf{n} + 2) \wedge\\
&(\mathbf{n} < \mathsf{k} < \mathbf{u} \ \Rightarrow\ \mathrm{EncodeN}[\mathsf{k} - \mathbf{n} - 1, \mathbf{n}]) \wedge\\
&\big(\mathbf{v} = \mathbf{u} + 3(2\mathbf{n}+1)^2\big) \wedge\\
&(\mathbf{u} \leq \mathsf{k} < \mathbf{v} \ \Rightarrow\ \mathrm{EncodeA}[\mathsf{k} - \mathbf{u}, \mathbf{n}, \mathsf{k}]) \wedge\\
&\big(\mathbf{w} = \mathbf{v} + 3(2\mathbf{n}+1)^2\big) \wedge\\
&(\mathbf{v} \leq \mathsf{k} < \mathbf{w} \ \Rightarrow\ \mathrm{EncodeB}[\mathsf{k} - \mathbf{v}, \mathbf{n}]) \wedge\\
&(\mathbf{w} \leq \mathsf{k} < \mathbf{w} + \mathsf{log} \ \Rightarrow\ \mathrm{Encode}\mathcal{T}[\mathsf{k} - \mathbf{w}, \mathcal{T}])\\
\Big)&
\end{aligned}
$$

The variables $\mathbf{n}$, $\mathbf{u}$, $\mathbf{v}$ and $\mathbf{w}$ have to be long because their values may be large compared to $|s'|$. The variable $\mathbf{n}$ for instance can take a maximal value of $\max\{30(|s'| - i - 1), 60\}$. Notice that ϕ checks if $k \leq |r(s')|$ by explicitly computing the length of $r(s')$, which depends only on the value of $\mathbf{n}$ and the value of

$\mathcal{T}$. To parse the string s and to represent the value of k the sentence applies the following sub formula:

$$
\begin{array}{l}
\text{DecodeParam}[\mathsf{k}] = \forall \mathsf{x}\ \Big(\\
\qquad (\neg X[\mathsf{k}]) \land \\
\qquad (\mathsf{x} < \mathsf{k}\ \Rightarrow\ X[\mathsf{x}]) \\
\Big)
\end{array}
$$

Next, to describe the encodings $enc^u(n)$ and $enc^u(m)$ the sentence applies the following very simple sub formula:

$$
\begin{array}{l}
\text{EncodeN}[\mathbf{i}, \mathbf{n}] =\ \Big(\\
\qquad (\mathbf{i} < \mathbf{n}) \\
\Big)
\end{array}
$$

Hence, in every model (s, a) with $a(\mathbf{n}) = n$ it is true that $a(\mathbf{i}) < n$ which states that in the encoding $enc^u(n)$ of n only the bits starting at index n, i. e., only the last one, are positive.

A more complex part of the sentence describes the encoding $enc^b(A_s)$:

$$
\begin{array}{l}
\text{EncodeA}[\mathbf{i}, \mathbf{n}, \mathsf{k}] = \exists \mathbf{i}'\ \exists \mathbf{j}'\ \exists \mathsf{bit}\ \exists \mathsf{c}\ \Big(\\
\qquad \left(\mathbf{i}' = \left\lfloor \frac{\mathbf{i} \mod 3(2\mathbf{n}+1)}{3} \right\rfloor - \mathbf{n}\right) \land \\
\qquad \left(\mathbf{j}' = \left\lfloor \frac{\mathbf{i}}{3(2\mathbf{n}+1)} \right\rfloor - \mathbf{n}\right) \land \\
\qquad (\mathsf{bit} = \mathbf{i} \mod 3) \land \\
\qquad (|\mathbf{i}'| \leq \mathbf{n} - 12 \land \mathbf{j}' = -\mathbf{n} + 13\ \Rightarrow\ \mathsf{c} = 4) \land \\
\qquad (|\mathbf{i}'| \leq \mathbf{n} - 12 \land \mathbf{j}' = \mathbf{n} - 13\ \Rightarrow\ \mathsf{c} = 5) \land \\
\qquad (\mathbf{i}' = -\mathbf{n} + 13 \land |\mathbf{j}'| \leq \mathbf{n} - 12\ \Rightarrow\ \mathsf{c} = 6) \land \\
\qquad (\mathbf{i}' = \mathbf{n} - 13 \land |\mathbf{j}'| \leq \mathbf{n} - 12\ \Rightarrow\ \mathsf{c} = 7) \land \\
\qquad (|\mathbf{i}'| = \mathbf{n} - 13 \land |\mathbf{j}'| = \mathbf{n} - 13\ \Rightarrow\ \mathsf{c} = 3) \land \\
\qquad ((|\mathbf{i}'| > \mathbf{n} - 13 \lor |\mathbf{j}'| > \mathbf{n} - 13) \land (\mathbf{j}' \neq 0)\ \Rightarrow\ \mathsf{c} = 3) \land \\
\qquad (|\mathbf{i}'| < \mathbf{n} - 13 \land |\mathbf{j}'| < \mathbf{n} - 13) \land \mathbf{j}' \neq 0\ \Rightarrow\ \mathsf{c} = 3) \land \\
\qquad (\mathbf{j}' = 0)\ \Rightarrow \Big(\\
\qquad\qquad (-\mathbf{n} \leq |\mathbf{i}'| < -\mathbf{n} + 13\ \Rightarrow\ \mathsf{c} = 2) \land \\
\qquad\qquad (-\mathbf{n} + 13 < |\mathbf{i}'| < 0\ \Rightarrow\ \mathsf{c} = 2) \land \\
\qquad\qquad (0 \leq \mathbf{i}' \leq (|s| - 1) - \mathsf{k} \land X[\mathbf{i}']\ \Rightarrow\ \mathsf{c} = 1) \land \\
\qquad\qquad (0 \leq \mathbf{i}' < (|s| - 1) - \mathsf{k} \land \neq X[\mathbf{i}']\ \Rightarrow\ \mathsf{c} = 0) \land \\
\qquad\qquad ((|s| - 1) - \mathsf{k} \leq \mathbf{i}' < \mathbf{n} - 13\ \Rightarrow\ \mathsf{c} = 2) \land \\
\qquad\qquad (\mathbf{n} - 13 < \mathbf{i}' \leq \mathbf{n}\ \Rightarrow\ \mathsf{c} = 2) \land \\
\qquad \Big)
\end{array}
$$

$$
\begin{aligned}
&\quad (\mathrm{Bit}[\mathsf{c}, \mathsf{bit}])\\
&)
\end{aligned}
$$

Hence, if (s, a) is a model, then the formula describes the index of the pixel encoded at the offset $a(\mathbf{i})$ with $(\mathbf{i'}, \mathbf{j'})$. Moreover, since every color value is built from three bits, the offset $a(\mathbf{i})$ gives also the queried bit in the respective pixel block, which is described by the variable bit. Afterwards, EncodeA describes conditions for the pixel being contained in A_s's frame, the centerline, the representation of s or somewhere else. Based on this it describes values for c being a color in $\mathcal{C} = \{0, \ldots, 7\}$. Finally, the formula makes sure that the value of bit describes a positive bit in the binary representation of the color value of c. Notice that the constant 3 in many arithmetic sub formulas comes from the number of bits, needed to encode a color value in $\mathcal{C}$.

Similarly, the sub formula EncodeB describes the encoding $enc^b(B_s)$:

$$
\begin{aligned}
&\mathrm{EncodeB}[\mathbf{i}, \mathbf{n}] = \exists \mathbf{i}'\ \exists \mathbf{j}'\ \exists \mathsf{bit}\ \exists \mathsf{c}\ \Big(\\
&\quad \left(\mathbf{i}' = \left\lfloor \tfrac{\mathbf{i} \mod 3(2\mathbf{n}+1)}{3} \right\rfloor - \mathbf{n}\right) \land\\
&\quad \left(\mathbf{j}' = \left\lfloor \tfrac{\mathbf{i}}{3(2\mathbf{n}+1)} \right\rfloor - \mathbf{n}\right) \land\\
&\quad (\mathsf{bit} = \mathbf{i} \mod 3) \land\\
&\quad (|\mathbf{i}'| < \mathbf{n} \land \mathbf{j}' = -\mathbf{n} \ \Rightarrow\ \mathsf{c} = 4) \land\\
&\quad (|\mathbf{i}'| < \mathbf{n} \land \mathbf{j}' = \mathbf{n} \ \Rightarrow\ \mathsf{c} = 5) \land\\
&\quad (\mathbf{i}' = -\mathbf{n} \land |\mathbf{j}'| < \mathbf{n} \ \Rightarrow\ \mathsf{c} = 6) \land\\
&\quad (\mathbf{i}' = \mathbf{n} \land |\mathbf{j}'| < \mathbf{n} \ \Rightarrow\ \mathsf{c} = 7) \land\\
&\quad (|\mathbf{i}'| = \mathbf{n} \land |\mathbf{j}'| = \mathbf{n} \ \Rightarrow\ \mathsf{c} = 3) \land\\
&\quad (|\mathbf{i}'| < \mathbf{n} \land 0 < |\mathbf{j}'| < \mathbf{n} \ \Rightarrow\ \mathsf{c} = 3) \land\\
&\quad (-\mathbf{n} < \mathbf{i}' < 0 \land \mathbf{j}' = 0 \ \Rightarrow\ \mathsf{c} = 2) \land\\
&\quad (0 \leq \mathbf{i}' < (|s| - 1) - \mathsf{k} \land \mathbf{j}' = 0 \ \Rightarrow\ \mathsf{c} = 1) \land\\
&\quad ((|s| - 1) - \mathsf{k} \leq \mathbf{i}' < \mathbf{n} \land \mathbf{j}' = 0 \ \Rightarrow\ \mathsf{c} = 2) \land\\
&\quad (\mathrm{Bit}[\mathsf{c}, \mathsf{bit}])\\
&\Big)
\end{aligned}
$$

Finally, the encoding of $\mathcal{T}_s$ is described by the following simple sub formula:

$$
\begin{aligned}
&\mathrm{Encode}\mathcal{T}[\mathbf{i}, \mathcal{T}] = \ \big(\\
&\quad (\mathrm{Bit}[\mathcal{T}, \mathbf{i}])\\
&\big)
\end{aligned}
$$

Hence, in ever model (s, a) it is true that $a(\mathbf{i})$ is the index of a positive bit in the binary representation of $a(\mathcal{T})$. $\square$

From Problem 5.25 being in U_D-AC_O it follows now that r is in U_D-FAC_O and thus, that it is justified to argue that the decision Problem 2.14 is complete in U_D-TC_O.

Lemma 5.27. *The function $r : \{0,1\}^* \to \Pi_{out}$ translating any string $s \in \{0,1\}^*$ into the encoding $z = r(s)$ of the instance $(A_s, B_s, \mathcal{T}_s)$ for the projective image matching Problem 2.14 is in U_D-FAC_O.*

Proof. By Lemma 5.26 it is known that Problem 5.25 is in U_D-AC_O. Consequently, there is a DLOGTIME-uniform family $\mathcal{N}$ of constant-depth, polynomial-size circuits deciding whether the kth bit in the output of r is positive.

Using $\mathcal{N}$, the function r can be evaluated for any given string $s' \in \{0,1\}^*$ by computing every bit k of $z = r(s')$ by the help of the circuit $N_{|k|+1+|s'|} \in \mathcal{N}$ deciding the string $1^k 0 s'$. This means, a family $\mathcal{N}'$ of constant-depth, polynomial-size circuits computing r is found by combining suitable circuits from $\mathcal{N}$.

Moreover, $\mathcal{N}'$ is DLOGTIME-uniform by a similar argumentations as in the proof of Theorem 5.20. □

The following is a straight implication of Corollary 5.21 and Lemma 5.27:

Corollary 5.28 (Without proof)**.** *Problem 2.14, the decision version of projective image matching, is U_D-TC_O-complete.*

Corollary 5.21 describes a reduction of the image matching Problem 2.14 for projective transformations to Problem 5.18, PROJECTIVE IMAGE MATCHING kTH BIT, which implies that this problem is U_D-TC_O-complete, too. This in turn allows to infer that the optimization Problem 2.13 belongs to the hardest in U_D-FTC_O, i. e., that every problem in this function class can be expressed in terms of projective image matching.

According to the previous section, showing the membership of projective subclasses in TC_O is not always trivial and, in fact, an open problem for linear transformations and its subclasses. However, the problem's hardness in TC_O follows easily for all considered subclasses of projective transformations, even for scalings or rotations, because the reduction builds mainly on δ and benefits from a restriction on the class of transformations.

5.4 Technical Proofs

The Proof of Theorem 5.9

Proof. Firstly, Vollmer [45] shows in his book that beside $<$ predicates for $\leq$, $=$, $\neq$, $\geq$ and $>$ can be expressed in first order formulas, too.

Moreover, he also shows that first order formulas can express the addition of variable values. Consequently, there is a first order predicate Plus$[\mathsf{u}, \mathsf{v}, \mathsf{w}]$ which accepts if and only if the values of u and v add up to the value of w. This implies straight the existence of $\mathrm{Sub}[\mathsf{u},\mathsf{v},\mathsf{w}] = \mathrm{Plus}[\mathsf{w},\mathsf{v},\mathsf{u}]$. Furthermore $\mathrm{Plus}[\mathsf{u},\mathsf{v},\mathsf{w}]$ enables the representation of arbitrary constant values. Particularly,

---

$$
\begin{aligned}
&\mathrm{Const}^k[\mathsf{v}] = \exists \mathsf{u}_1\ \forall \mathsf{w}\ \exists \mathsf{u}_2\ \ldots\ \exists \mathsf{u}_{k-1}\ \big(\\
&\qquad (<[0,\mathsf{u}_1]) \wedge\\
&\qquad (<[0,\mathsf{w}]\ \Rightarrow\ \leq[\mathsf{u}_1,\mathsf{w}]) \wedge\\
&\qquad (\mathrm{Plus}[\mathsf{u}_1,\mathsf{u}_1,\mathsf{u}_2]) \wedge\\
&\qquad (\leq[\mathsf{u}_2,(|s|-1)]) \wedge\\
&\qquad \vdots\\
&\qquad (\mathrm{Plus}[\mathsf{u}_{k-1},\mathsf{u}_1,\mathsf{v}])\\
&\big)
\end{aligned}
$$

---

binds the constant k to a given variable v.

Afterwards Vollmer [45] proves that first order formulas can express bit counting for variables, i. e., that there is a predicate $\mathrm{BitCount}[\mathsf{u},\mathsf{v}]$ which is true if and only if the value of v determines the number of positive bits in the binary representation of u's value. Due to Chandra et al. [10] it is true that multiplication reduces to bit counting and thus, it can be assumed that the predicate $\mathrm{Mul}[\mathsf{u},\mathsf{v},\mathsf{w}]$, which is true if the value of u times the value of v equals the value of w, can be expressed in a first order formula. Again, this implies the existence of

---

$$
\begin{aligned}
&\mathrm{DivF}[\mathsf{u},\mathsf{v},\mathsf{w}] = \exists \mathsf{u}'\ \exists \mathsf{u}''\ \big(\\
&\qquad (\mathrm{Mul}[\mathsf{w},\mathsf{v},\mathsf{u}']) \wedge\\
&\qquad (\mathrm{Plus}[\mathsf{u}',\mathsf{v},\mathsf{u}'']) \wedge\\
&\qquad (\leq[\mathsf{u}',\mathsf{u}]) \wedge\\
&\qquad (<[\mathsf{u},\mathsf{u}''])\\
&\big)
\end{aligned}
$$

---

expressing that rounding down the amount of dividing the values of u and v results in the value of w. Hence, a formula $\phi = (t_1 \star t_2)$ with terms t_1 and t_2 and $\star \in \{<,\leq,=,\neq,\geq,>\}$ can be translated into a first order formula. □

The Proof of Theorem 5.13

Proof. The counting quantifier is realized in three steps. Firstly, the following formula introduces the quantifier $\psi = Q_{\mathsf{n}}\mathsf{v}\ \psi'$. The formula ψ has a model (s,a) if and only if $(s,a_i) \models \psi'$ for at least $\left\lceil \frac{a(\mathsf{n})}{2} \right\rceil$ assignments a^i giving v values i in

$\{0,\dots,a(\mathsf{n})\}$:

$$
\begin{aligned}
&Q_{\mathsf{n}}\mathsf{v}\ \psi' = \exists \mathsf{u}\ Q\mathsf{v}\ \big(\\
&\qquad \Big(\mathrm{Bit}[\mathsf{n},0] \ \Rightarrow\ \mathsf{u} = \mathsf{n} + \Big\lfloor \tfrac{(|s|-1)-\mathsf{n}}{2} \Big\rfloor \not\Rightarrow \mathsf{u} = \mathsf{n} + \Big\lceil \tfrac{(|s|-1)-\mathsf{n}}{2} \Big\rceil\Big) \ \wedge\\
&\qquad (\mathsf{v} \le \mathsf{n} \ \Rightarrow\ \psi') \wedge\\
&\qquad (\mathsf{n} < \mathsf{v} \le \mathsf{u} \ \Rightarrow\ \mathsf{v} = \mathsf{v}) \wedge\\
&\qquad (\mathsf{u} < \mathsf{v} \le (|s|-1) \ \Rightarrow\ \mathsf{v} \ne \mathsf{v})\\
&\big)
\end{aligned}
$$

For all $i \in \{0,\dots,a(\mathsf{n})\}$ the formula simply queries ψ'. Then it devides the remaining assignments $a^i(\mathsf{v})$ with $i \in \{a(\mathsf{n})+1,\dots,|s|-1\}$ into two equal parts and makes sure that one half becomes a model of ψ and the other not. In this way, the formula realizes that the needless assignments have no impact.

The next step realizes the quantifier $\psi = G^{\mathsf{t}}_{\mathsf{n}}\mathsf{v}\ \psi'$. Such a formula ψ has a model (s,a) if and only if $2a(\mathsf{t}) \le 2a(\mathsf{n}) + 2 < |s|$ and $(s,a_i) \models \psi'$ for at least $a(\mathsf{t})$ assignments a^i giving v values i in $\{0,\dots,a(\mathsf{n})\}$:

$$
\begin{aligned}
&G^{\mathsf{t}}_{\mathsf{n}}\mathsf{v}\ \psi' = \exists \mathsf{u}\ \big(\\
&\qquad \Big(\mathsf{n} \le \Big\lfloor \tfrac{|s|-1}{2} \Big\rfloor\Big) \wedge\\
&\qquad (\mathsf{t} \le \mathsf{n}+1) \wedge\\
&\qquad \big(\mathsf{t} \le \lceil \tfrac{\mathsf{n}}{2} \rceil\big) \ \Rightarrow \big(\\
&\qquad\qquad (\mathsf{u} = 2(\mathsf{n}-\mathsf{t})+1) \wedge\\
&\qquad\qquad \big(Q_{\mathsf{u}}\mathsf{v}\ \big((\mathsf{v} \le \mathsf{n}) \Rightarrow \psi'\big)\big)\\
&\qquad \big) \not\Rightarrow \big(\\
&\qquad\qquad (\mathsf{u} = 2\mathsf{t}-1) \wedge\\
&\qquad\qquad \big(Q_{\mathsf{u}}\mathsf{v}\ \big((\mathsf{v} \le \mathsf{n}) \Rightarrow \psi' \wedge (\mathsf{u} \le \mathsf{n})\big)\big)\\
&\qquad \big)\\
&\big)
\end{aligned}
$$

In this formula the majority is shifted to a specific value $a(\mathsf{t})$, simply by forcing the formula to be true or false for values of v beyond $a(\mathsf{n})$. The last step applies the G-quantifier to establish counting:

$$
\begin{aligned}
&Q^{\mathsf{t}}_{\mathsf{n}}\mathsf{v}\ \phi' = \exists \mathsf{u}\ \big(\\
&\qquad \Big(\mathsf{n} \le \Big\lfloor \tfrac{|s|-1}{2} \Big\rfloor\Big) \wedge\\
&\qquad (\mathsf{t} \le \mathsf{n}+1) \wedge\\
&\qquad (G^{\mathsf{t}}_{\mathsf{n}}\mathsf{v}\ \phi') \wedge\\
&\qquad \big(G^{\mathsf{n}+1-\mathsf{t}}_{\mathsf{n}}\mathsf{v}\ \neg\phi'\big)\\
&\big)
\end{aligned}
$$

Clearly, the number of models (s, a_i) with $i \in \{0, \dots, a(\mathsf{n})\}$ is exactly $a(\mathsf{t})$ if at least $a(\mathsf{t})$ of them model ϕ' and $(a(\mathsf{n}) + 1 - a(\mathsf{t}))$ do not model ϕ'. □

The Proof of Lemma 5.22

Proof. The following shows first that for all $(i, j) \in z(n)$ with $j \neq 0$ it is true $f_s(A_s)[n](i, j) = B_s(i, j)$. Consider the inverse transformation f_s^{-1} given by the inverse projection matrix M_s^{-1}:

$$M_s^{-1} = \begin{pmatrix} v^{-1} & 0 & 0 \\ 0 & v^{-1} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

where $v^{-1} = 1 - \frac{13}{n}$. An index (i, j) is matched whenever $f_s(A_s)[n](i, j) = B_s(i, j)$.

Clearly

$$[f_s^{-1}(k, -n)] = (I, J) = \left[\left(k - \frac{13k}{n}, -n + 13\right)\right]$$

for all $k \in \{-n, \dots, n\}$. Then $f_s(A_s)[n](-n, -n) = A_s(-n + 13, -n + 13) = 3 = A_s(n - 13, -n + 13) = f_s(A_s)[n](n, -n)$. Moreover, since $n > 26$ it is true for all $k \in \{-n + 1, \dots, n - 1\}$ that $-n + 14 \leq I \leq n - 14$ and $J = -n + 13$ which implies $f_s(A_s)[n](k, -n) = A_s(I, J) = 4 = B(k, -n)$. Hence, all indices in the frame side with color 4 are matched. It is easily shown in an analogous way that the remaining indices of the frame are matched, too.

Notice for all $i \in \{-n + 1, \dots, n - 1\}$ and $j \in \{|s| + 1, \dots, n - 1\}$ that

$$[f_s^{-1}(i, j)] = (I, J) = \left[\left(i - \frac{13i}{n}, j - \frac{13j}{n}\right)\right].$$

Because $n > 26$ and $n \geq 30|s|$ it follows $-n+14 \leq I \leq n-14$ and $|s| < J \leq n-14$ which means again $f_s(A_s)[n](i, j) = A_s(I, J) = 3 = B(i, j)$. The same holds trivially for all $j \in \{-n + 1, \dots, -|s| - 1\}$.

For $i \in \{|s| + 1, \dots, n - 1\}$ and $j \in \{-|s|, \dots, |s|\}$ it is true that $[f_s^{-1}(i, j)] = (I, j)$ with $|s| < I \leq n - 14$ and analogously for $i \in \{-n + 1, \dots, -|s| - 1\}$ the similar statement of $[f_s^{-1}(i, j)] = (I, j)$ with $-|s| > I \geq -n + 14$ is true. Consequently, with the exception of indices (i, j) with $i, j \in \{-|s|, \dots, |s|\}$ all indices in $z(n)$ have been matched.

In this remaining square it is clearly true that $[f_s^{-1}(i, j)] = (i, j)$. This means for all such (i, j) with $j \neq 0$ that $f_s(A_s)[n](i, j) = A_s(i, j) = 3 = B(i, j)$.

Moreover a representation of the string s is exactly contained in $f_s(A_s)[n]$ and compared against a representation of positive bits in B_s and thus,

$$\begin{aligned}\delta(f_s(A_s)[n], B_s) &= \sum_{(i,j)\in z(n)} \omega(f_s(A_s)[n](i,j), B_s(i,j)) \\ &= \sum_{k=-n}^{n} |f_s(A_s)[n](k,0) - B_s(k,0)| \\ &= \sum_{k=0}^{|s|-1} |s(k) - 1|,\end{aligned}$$

which implies the lemma's statement. □

The Proof of Lemma 5.23

Proof. Notice first that $\delta(f_s(A_s)[n], B_s) \leq |s|$, i.e, extremely small compared to the number of indices ($> 3600|s|^2$). Even every side of the frame in B_s has roughly $60|s|$ indices. Hence, the idea behind the proof is that every good transformation f has to transform the frame in A_s such that it is approximately matched against the frame in B_s. Then, this approximate frame matching implies the image A_s to be spanned in such a way that the representation of s in $f(A_s)[n]$ is exactly positioned on row $j = 0$. This means that s is matched against a string of positive bits to count the number of zero bits in s, which actually realizes the idea of the reduction.

Assume the existence of a transformation $f \in \mathcal{F}_{\mathrm{p}}$, with $\delta(f(A_s)[n], B_s) < \delta(f_s(A_s)[n], B_s) \leq |s|$, i. e., one that performs a better match than f_s. According to the proof of Lemma 5.22 it is true that $f_s(A_s)[n](i,j) = B_s(i,j)$ for all $(i,j) \in z(n)$ with the exception of indices $(i,0)$ fulfilling $i \in \{0,\ldots,|s|-1\}$. Clearly, f cannot beat f_s in these regions. Consequently, $f(A_s)$ has to match the indices $(i,0)$ with $i \in \{0,\ldots,|s|-1\}$ better than f_s possibly by tolerating additional costs in other parts.

Since $B_s(i,0)$ contains color 2 for all $i \in \{-n,\ldots,-1,\}\cup\{|s|,\ldots,n\}$ a shifting along the i-coordinate gives no possibility for f to generate essential differences in row $j = 0$. Particularly, there are only two ways for f to change row $j = 0$ such that

$$\sum_{i=0}^{|s|-1} |f(A_s)[n](i,0) - 1| \neq \sum_{i=0}^{|s|-1} |f_s(A_s)[n](i,0) - 1|\,,$$

i. e., to realize a difference in the distortions between $f(A_s)$ and B_s and between $f_s(A_s)$ and B_s restricted to the small part of row $j = 0$. Firstly, it is possible to alter the j-coordinate of at least one index to make it map outside the center row of A_s, i. e., such that there is $i \in \{0,\ldots,|s|-1\}$ with $f^{-1}(i,0) = (i',j')$ and $j' \neq 0$. In that case the color $f(A_s)[n](i,0)$ is artificially changed to 3, 4, … or 7. However, since $B_s(i,0) = 1$ this introduces higher costs compared to

$\delta(f_s(A_s)[n], B_s)$. This means that f has to identify the center rows of A_s and B_s, i. e., at least for the indices $(i, 0)$ with $i \in \{0, \ldots, |s| - 1\}$ it is true that $f^{-1}(i, 0) = (i', j')$ has $j' = 0$.

The other way of modification is to change the scaling within row $j = 0$. Particularly, f_s realizes an identity mapping in the essential parts of the center row containing the representation of s. That means, for all $i \in \{0, \ldots, |s|-1\}$ it is true that $[f_s^{-1}(i, 0)] = (i, 0)$. In this way f_s makes sure that $f_s(A_s)[n]$ contains an exact copy of the representation of s. However, in terms of global costs it could be profitable for f to skip or repeat parts of the representation of s in $f(A_s)[n]$. Clearly, just repeating things cannot help improving the distortion because it leads to multiple counting of zero bits.

But skipping zero bits could save costs. In fact, it could be the case that there are indices $(I, 0)$ in A_s with $I \in \{0, \ldots, |s| - 1\}$ that are missed by the transformation f^{-1}. Hence, it may happen that no index $(i, 0)$ with $i \in \{-n, \ldots, n\}$ has $[f^{-1}(i, 0)] = (I, 0)$. In this case $f(A_s)[n]$ would no longer contain a proper copy of the string s, particularly it lacks A_s's information at index $(I, 0)$. Especially, if $A_s(I, 0) = s(I) = 0$ then $f(A_s)[n]$ would miss this zero bit of s. As a consequence, the distortion computation between $f(A_s)[n]$ and B_s could not perform a correct counting of the number of zero bits in s.

The following shows that a skipping as described above has to be payed by inadequately high costs introduced by distortion in other parts of $f(A_s)[n]$. For this purpose assume actually that there is $I \in \{0, \ldots, |s| - 1\}$ such that for all $i \in \{-n, \ldots, n\}$ it is true that $[f^{-1}(i, 0)] = (i', j')$ with $i' \neq I$.

The following argumentation builds basically on the line preserving property of projective transformations. Particularly, according to the discussion in Lemma 2.10 it is true for the transformation f and for every straight line L in $\mathbb{R}^2$ that $L' = \{f^{-1}(x) \mid x \in L\}$ is a straight line, too. Notice that the indices $(-n, -n), (-n + 1, -n), \ldots, (n, -n)$, matching against the frame side in B_s with color 4, are situated on a line. Consequently, it is true that $f^{-1}(-n, -n), f^{-1}(-n + 1, -n), \ldots, f^{-1}(n, -n)$ are a set of points on another line L'. The same happens at the other three sides of the frame.

If I's existence implies that $[f^{-1}(i, -n)] = (i', j')$ with (i', j') being outside the frame side in A_s with color 4, i. e., if $i' \in \{-n+13, \ldots, n-13\}$ implies $j' \neq -n+13$, it follows from the line preservation that either all $k \in \{-n, \ldots, i - 1\}$ or all $k \in \{i + 1, \ldots, n\}$ give indices $[f^{-1}(k, n)]$ not being part of that frame side in A_s, too. Hence, if $|i|$ is not too close to n, then quite a number of frame indices remain unmatched and cause distortion. The same is true for the other three sides of the frame.

Consequently, consider the eight indices

$$\begin{array}{llll} c_1 = (-0.5n, -n), & c_2 = (0.5n, -n), & c_3 = (-0.5n, n), & c_4 = (0.5n, n), \\ c_5 = (-n, -0.5n), & c_6 = (-n, 0.5n), & c_7 = (n, -0.5n), & c_8 = (n, 0.5n) \end{array}$$

where c_1, c_2 correspond to the frame side with color 4, c_3, c_4 to the frame side with color 5, c_5, c_6 to the side with color 6 and c_7, c_8 to the side with color 7. Moreover, let

$$\begin{array}{rclrcl} f^{-1}(c_1) &=& (X_1, Y_1), & f^{-1}(c_2) &=& (X_2, Y_2),\\ f^{-1}(c_3) &=& (X_3, Y_3), & f^{-1}(c_4) &=& (X_4, Y_4),\\ f^{-1}(c_5) &=& (X_5, Y_5), & f^{-1}(c_6) &=& (X_6, Y_6),\\ f^{-1}(c_7) &=& (X_7, Y_7), & f^{-1}(c_8) &=& (X_8, Y_8). \end{array}$$

Assume that, although skipping the index $(I, 0)$, f maps all eight indices into the correct frame sides. In fact, regard the following weaker assumption:

$$\begin{array}{rcccl} -n + 12.5 &\leq& Y_1, Y_2, X_5, X_6 &\leq& -n + 13.5 \quad \text{and}\\ n - 13.5 &\leq& Y_3, Y_4, X_7, X_8 &\leq& n - 12.5. \end{array}$$

The remainder of the proof shows that this leads to a contradiction with the precondition of f skipping $(I, 0)$. For this purpose remember that $f^{-1} \in \mathcal{F}_{\mathrm{p}}$. Consequently, for all $(x, y) \in \mathbb{R}^2$ it is true that $f^{-1}(x, y) = \left(\frac{x'}{z'}, \frac{y'}{z'}\right)$ with

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} p_1 & p_2 & p_3 \\ p_4 & p_5 & p_6 \\ p_7 & p_8 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

for some $p_1, \ldots, p_8 \in \mathbb{R}^8$. If the values of Y_1, Y_2, X_5, X_6 and Y_3, Y_4, X_7, X_8 were known exactly then p_1 to p_8 would be precisely determined by the following system of linear equations:

$$\begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \\ p_7 \\ p_8 \end{pmatrix} = \begin{pmatrix} -n & -0.5n & 1 & 0 & 0 & 0 & nX_5 & 0.5nX_5 \\ -n & 0.5n & 1 & 0 & 0 & 0 & nX_6 & -0.5nX_6 \\ n & -0.5n & 1 & 0 & 0 & 0 & -nX_7 & 0.5nX_7 \\ n & 0.5n & 1 & 0 & 0 & 0 & -nX_8 & -0.5nX_8 \\ 0 & 0 & 0 & -0.5n & -n & 1 & 0.5nY_1 & nY_1 \\ 0 & 0 & 0 & 0.5n & -n & 1 & -0.5nY_2 & nY_2 \\ 0 & 0 & 0 & -0.5n & n & 1 & 0.5nY_3 & -nY_3 \\ 0 & 0 & 0 & 0.5n & n & 1 & -0.5nY_4 & -nY_4 \end{pmatrix}^{-1} \cdot \begin{pmatrix} X_5 \\ X_6 \\ X_7 \\ X_8 \\ Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \end{pmatrix}.$$

Particularly, it is true that

$$\begin{aligned} p_1 &= \tfrac{X_5(X_6(-Y_1+3Y_2+Y_3-3Y_4)+X_8(3Y_1-5Y_2+5Y_3-3Y_4))+X_7(X_8(3Y_1-Y_2-3Y_3+Y_4)+X_6(-5Y_1+3Y_2-3Y_3+5Y_4))}{n(X_7-X_6)(5Y_1-3Y_2+3Y_3-5Y_4)+n(X_5-X_8)(3Y_1-5Y_2+5Y_3-3Y_4)} \\ p_3 &= \tfrac{X_8(X_7(3Y_1-Y_2-3Y_3+Y_4)+X_6(-6Y_1+6Y_2-2Y_3+2Y_4))+X_5(X_7(2Y_1-2Y_2+6Y_3-6Y_4)+X_6(Y_1-3Y_2-Y_3+3Y_4))}{(X_7-X_6)(5Y_1-3Y_2+3Y_3-5Y_4)+(X_5-X_8)(3Y_1-5Y_2+5Y_3-3Y_4)} \\ p_7 &= \tfrac{2(X_5-X_7)(Y_1-Y_2+3Y_3-3Y_4)+2(X_8-X_6)(3Y_1-3Y_2+Y_3-Y_4)}{n(X_7-X_6)(5Y_1-3Y_2+3Y_3-5Y_4)+n(X_5-X_8)(3Y_1-5Y_2+5Y_3-3Y_4)}. \end{aligned}$$

However, instead of exact values for Y_1, Y_2, X_5, X_6 and Y_3, Y_4, X_7, X_8 the proof assumes only narrow bounds, i. e., $-n + 12.5 \leq Y_1, Y_2, X_5, X_6 \leq -n + 13.5$ and $n - 13.5 \leq Y_3, Y_4, X_7, X_8 \leq n - 12.5$. Based on this bounds it is possible, yet technically involved, to derive the following worst case estimations for p_1, p_3 and p_7:

$$\begin{array}{rclcll} p_1 & \leq & 1 - \frac{64n^2-1930n+14375}{8n(n-13.5)(n-15)} & \leq & 1-\frac{7}{n} & \text{for } n \geq 29, \\ |p_3| & \leq & \frac{8n^2-208n+1353}{8n(n-13.5)(n-15)} & \leq & \frac{2}{n} & \text{for } n \geq 31 \text{ and} \\ |p_7| & \leq & \frac{2n-27}{n(n-13.5)(n-15)} & \leq & \frac{3}{n^2} & \text{for } n \geq 58. \end{array}$$

Hence, p_1 is close, yet below, one and p_3 and p_7 have absolute values close to zero.

The idea works as follows. If f skips $(I,0)$ then there is $i \in \{-n,\ldots,n\}$ such $f^{-1}(i,0) = (i_1', j_1')$ and $f^{-1}(i+1,0) = (i_1', j_1')$ fulfills $i_1' < I < i_2'$. This means that $(i,0)$ is the rightmost index in the center row of $f(A_s)[n]$ mapping left of I and already $(i+1,0)$ maps right of I. Clearly, as i_1, I and i_2 are integer it follows that $f^{-1}(i+1,0) - f^{-1}(i,0) \geq 1$.

Since the two indices belong to the center row $j = 0$ and based on the estimations on p_1, p_3 and p_7 it is also true that

$$\begin{aligned} f^{-1}(i+1,0) - f^{-1}(i,0) &= \frac{(i+1)p_1 + p_3}{(i+1)p_7+1} - \frac{ip_1+p_3}{ip_7+1} \\ &= \frac{p_1 - p_3p_7}{((i+1)p_7+1)(ip_7+1)} \\ &\leq \frac{1-\frac{7}{n}+\frac{6}{n^3}}{(1-\frac{3n}{n^2})(1-\frac{3(n-1)}{n^2})} \quad \text{for } n \geq 58 \\ &\leq \frac{n^3-7n^2+6}{n^3-6n^2+9n} \\ &< 1 \quad \text{for } n \geq 1. \end{aligned}$$

This is a contradiction. Hence, if f skips $(I,0)$ then the weak assumption of $-n+12.5 \leq Y_1, Y_2, X_5, X_6 \leq -n+13.5$ and $n-13.5 \leq Y_3, Y_4, X_7, X_8 \leq n-12.5$ fails. Consequently, at least one of the eight indices c_1 to c_8 is mapped outside the corresponding frame side. Without loss of generality let $c_1 = (-0.5n, -n)$ be that index and thus, either $Y_1 < -n+12.5$ or $Y_1 > -n+13.5$. But then it follows from the line preservation of f^{-1} that all indices $(i_1, -n)$ with $i < -0.5n$ or all indices $(i_2, -n)$ with $i_2 > -0.5n$ do not match with the 4-colored frame side either. This means, at least $14|s|$ indices do not match which causes a distortion of more than $|s|$.

Equivalently, c_2 to c_8 falling out of the frame implies inadequately high costs. Consequently, f cannot pay the advantage of skipping $(I,0)$ which finally contradicts the assumption of f being better than f_s. $\square$

Chapter 6

Conclusions and Further Research

The intention of this last chapter is to draw a wider picture of image matching. Section 6.1 extends the results of Chapter 4 to a related computational problem, *two-dimensional pattern matching*. Results for one problem can often be transferred to the other and accordingly some results for image matching are extended to the two-dimensional pattern matching problem.

Afterwards, Section 6.2 gives an overview on further research results and ongoing work. Particularly, one open problem is to determine eligible *lower bounds* on the cardinality of dictionaries because their values essentially determine the running times of the presented algorithms. This recent question has been answered by the author and Maciej Liśkiewicz for small transformation classes like rotation and scaling. Progress has also been made in implementing parts of the introduced algorithm which demonstrated its practical relevance. Moreover, latest studies provide evidence that image matching is not only feasible for nearest neighbor interpolation, which, although being a standard technique, is a bit outdated. Linear and higher order interpolation do not seem to make efficient image matching impossible.

Finally, Section 6.3 summarizes and concludes the findings of of the four previous chapters.

6.1 Pattern Matching via Generic Image Matching

Section 1.1 states that, when concentrating on the discrete nature of image transformations, two-dimensional combinatorial pattern matching [2, 3, 4, 5, 17, 18, 33, 40] is the origin of research in image matching. In fact, both settings are very closely related and many results can easily be transfered. The combinatorial pattern matching problem can be formally described as follows:

Problem 6.1. TWO-DIMENSIONAL PATTERN MATCHING

Input: *Digital image A of size n and image B of size m.*

Constraints:
Subclass $\mathcal{F}$ of projective transformations with $\mathcal{P}[\mathcal{F}] : \mathbb{R}^d \to \mathcal{F}$ and $d \leq 8$.

Output: *All translation vectors $(t_1, t_2) \in z(m)$ which admit parameters $P \in \mathbb{R}^d$ such that for $f^{-1} = \mathcal{P}[\mathcal{F}]\,(P)$ and for all (i, j) in support $z(m)$ with $(i', j') = [f^{-1}(i - t_1, j - t_2)] \in z(n)$ it is true that $A(i', j') = B(i, j)$.*

Hence, instead of searching for the best match between A and B, pattern matching looks for all positions (t_1, t_2) in B that contain an exact copy of A under some transformations $f \in \mathcal{F}$.

According to Section 1.1 previous work on this problem leads to combinatorial image matching algorithms for some very small subclasses of projective transformations, such as solely scalings or solely rotations. However, results for image matching can be led back to the two-dimensional pattern matching problem, too. In fact, it is possible to solve the problem for the same repertoire of transformation classes as for image matching, i. e., up to projective transformations. However, since the problem statement involves already integer translations, it would be superfluous to consider transformation classes involving continuous translations.

In principle the solution to two-dimensional pattern matching is simply found by calling the image matching algorithm in Figure 4.4 of Section 4.3 as a subroutine for every meaningful translation vector (t_1, t_2). Figure 6.1 shows an efficient polynomial time approach. The algorithm works as follows. First, it makes sure that A and B do not contain color zero. More precisely, color zero is used only to identify background indices, i. e., for transformations f, background indices (i, j) in $f(A)[2m]$ give $f^{-1}(i, j) \notin z(n)$. Then, it prepares a distortion measure that actually tolerates the background color zero. This way makes sure that the background of any image $f(A)[2m]$ has no impact on the distortion $\delta(f(A)[2m], B')$. Finally, it calls the image matching algorithm for all translations (t_1, t_2) using the background tolerant distortion measure. The translation (t_1, t_2) is realized in the copy B' of B. If $\delta(f(A)[2m], B') = 0$ then, clearly, B contains an exact copy of a transformation of A at position (t_1, t_2).

The following theorem summarizes the results:

Theorem 6.2. *Let A be an image of size n, B be an image of size m and let χ be any k-dimensional slice of $\mathbb{R}^8$. If every face φ in $\mathcal{H}_\chi[m, n]$ contains a valid point P that can be computed in at most T time, then the algorithm in Figure 6.1 determines in $O(|\mathcal{E}[m, n]|^k) \cdot O(m^2) \cdot T$ time a list L of translations (t_1, t_2) which admit transformations $f \in \mathcal{F}_\chi$ such that for all $(i, j) \in z(m)$ it is true that $(i', j') = [f^{-1}(i - t_1, j - t_2)] \in z(n)$ gives $A(i', j') = B(i, j)$.*

Algorithm `Generic Two-dimensional Pattern Matching`
Input: Image A of size n and image B of size m.
Parameter: A slice χ in $\mathbb{R}^8$ and a constant $e \geq 1$.
Output: List L of translations (t_1, t_2) which admit a transformation $f \in \mathcal{F}_\chi$ where all $(i, j) \in z(m)$ with $(i', j') = [f^{-1}(i - t_1, j - t_2)] \in z(n)$ give $A(i', j') = B(i, j)$

```
1.  Procedure MAIN()                                              /* Main Program */
2.      $L = \emptyset$;                                              /* initialize solution */
3.      determine $c$, largest color in $A$ and $B$;           /* preserve zero for background */
4.      for all $(i, j) \in z(n)$ do if $A(i, j) = 0$ then set $A(i, j) = c + 1$;
5.      for all $(i, j) \in z(m)$ do if $B(i, j) = 0$ then set $B(i, j) = c + 1$;
6.      let $\delta(a, b) = \begin{cases} 0 : \text{if } a = 0, b = 0, a = b, \\ 1 : \text{otherwise} \end{cases}$ ;  /* set background tolerant distortion */
7.      for all $(t_1, t_2) \in z(m)$ do                          /* iterate over all translations */
8.          for all $(i, j) \in z(2m)$ do $B'(i, j) = B(i - t_1, j - t_2)$;  /* double size copy of $B$ */
9.          call Generic Image Matching with $(A, B')$ and parameter $\chi$, $\delta$ and $e$;
10.         store result in $P$ and let $f = (\mathcal{P}[\mathcal{F}_{\mathtt{p}}]\,(P))^{-1}$;
11.         if $\delta(f(A)[2m], B') = 0$ then add $(t_1, t_2)$ to $L$;        /* verify exact copy */
12.     end for
13.     return $L$;
```

Figure 6.1: The projective two-dimensional pattern matching algorithm. The matching is realized by the projective image matching algorithm.

In particular, the algorithm solves the two-dimensional pattern matching problem in time

$O(m^{10}n^4)$ *for linear transformations* $\mathcal{F}_\ell$,

$O(m^6n^2)$ *for scalings and rotations* $\mathcal{F}_{\mathtt{sr}}$ *and*

$O(m^4n)$ *for scalings* $\mathcal{F}_{\mathtt{s}}$.

Proof. Since A is of size n and B' of size $2m$ it follows from Theorem 4.15 that a single call to the projective image matching subroutine takes $O(|\mathcal{E}[2m, n]|^k) \cdot T = O(|\mathcal{E}[m, n]|^k) \cdot T$ time. But there are $(4m + 1)^2 = O(m^2)$ calls to the subroutine and thus, the first part of the theorem follows.

According to this, the estimations for the concrete examples $\mathcal{F}_\ell$, $\mathcal{F}_{\mathtt{sr}}$ and $\mathcal{F}_{\mathtt{s}}$ are a straight implication of Corollary 4.21. □

Similar to the discussion in the previous section it is true that the proposed technique can also be used to solve two-dimensional pattern matching under rotations $\mathcal{F}_{\mathtt{r}}$. In fact, the author and Maciej Liśkiewicz [26] show that this approach runs in $O(m^2n^3)$.

The exact complexity of two-dimensional pattern matching remains an open question. It is straightforward that the problem is in TC_O for the class of projective transformations. The same is true for affine transformations. However, considering the pattern matching problem under this classes of transformations is not very meaningful as they feature translations. For such classes it is either the case that L is empty or that L consists of all possible translation vectors (t_1, t_2). However, as for image matching, the complexity of two-dimensional pattern matching remains open for other subclasses of projective transformations. In fact, it is reasonable to assume the containment of both problems in TC_O for all considered projective subclasses but a proof of the assumption remains future work.

6.2 Overview on Work Progress

This section gives an overview on other completed and ongoing research for the image matching problem. Firstly, analysis in Chapter 3 to 4 provided only upper bounds, on the cardinality of dictionaries. However, to get a realistic impression on the structure of $\mathcal{D}[\mathcal{F}_\chi, m](A)$ it is necessary to obtain lower bounds for all $\chi \in \{\mathbb{R}^8, \chi_{\mathtt{a}}, \chi_\ell, \chi_{\mathtt{t}}, \chi_{\mathtt{sr}}, \chi_{\mathtt{s}}\}$.

Remember that upper bounds on $|\mathcal{D}[\mathcal{F}_\chi, m](A)|$ for subclasses $\mathcal{F}_\chi$ of projective transformations defined by a slice χ are deduced from upper bounds on the number of faces in $\mathcal{H}_\chi[m, n]$. But this does not take into account the color information of A. For this reason, concluding lower bounds for $|\mathcal{D}[\mathcal{F}_\chi, m](A)|$ directly from lower bounds on $|\mathcal{H}_\chi[m, n]|$ is not correct. Particularly, it could by the case that many faces in $\mathcal{H}_\chi[m, n]$ stand for the same transformation of A. This may happen for example if A is uniformly colored.

However, Amir et al. [2] show for the setting of rotations that $\mathcal{H}_\lambda[m, n]$, the partition of the non-linear subspace λ of $\mathbb{R}^8$, contains at least $\Omega(n^3)$ faces. Together with the upper bound of $O(n^3)$ this gives an exact impression of the cardinality of $\mathcal{H}_\lambda[m, n]$. But, this does not mean for any image A that $|\mathcal{D}[\mathcal{F}_{\mathtt{r}}, m](A)| \in \Theta(n^3)$. Moreover, the author and Maciej Liśkiewicz [24, 25] provide that the set $\mathcal{H}_{\chi_{\mathtt{sr}}}[n, n]$ has a cardinality of at least $\Omega(n^6/\log n)$ which is close to the upper bound $|\mathcal{H}_{\chi_{\mathtt{sr}}}[n, n]| \in O(n^6)$. But again, this bound has no direct implication for the cardinality of $\mathcal{D}[\mathcal{F}_{\mathtt{sr}}, n](A)$, the set of all combined scalings and rotations with size-n of any given image A of the same size n.

The author and Maciej Liśkiewicz [24, 25] were the first who provide lower bounds by showing that there is a family $A_n, n \in \mathbb{N}$ of images for which $|\mathcal{D}[\mathcal{F}_{\mathtt{r}}, m](A_n)| \in \Omega(n^3)$ and $|\mathcal{D}[\mathcal{F}_{\mathtt{sr}}, n](A_n)| \in \Omega(n^6/\log n)$. This means, in worst case, e. g., for an image A_n, it happens that upper and lower bounds on the cardinalities of $\mathcal{D}[\mathcal{F}_{\mathtt{r}}, m](A)$ and $\mathcal{D}[\mathcal{F}_{\mathtt{sr}}, n](A)$ are nearly the same. Consequently, the standard approach of enumerating all images in the dictionary cannot lead to essentially faster algorithms and to obtain really improved running times requires completely new ideas. Remarkably, it is possible to deduce a lower worst case bound of $\Omega(n^{12}/\log^2 n)$ for $|\mathcal{D}[\mathcal{F}_\ell, n](A)|$, the dictionary of linear transformations, by the use of the lower bound on $|\mathcal{D}[\mathcal{F}_{\mathtt{sr}}, n](A)|$. However, it remains future work to reveal lower tight bounds on the dictionary cardinalities for higher transformation classes, like affine or projective transformations.

Another aspect of supplementary work are implementations of the proposed image matching algorithms. First impressions on the algorithm's performance have been gained by Ragnar Nevries' diploma thesis [39]. However, in addition to that, its Java™ implementation revealed new insights into the structural properties of $\mathcal{H}_{\chi_{\mathtt{sr}}}[m,n]$. Particularly, it was possible to determine some exact values of $|\mathcal{H}_{\chi_{\mathtt{sr}}}[m,n]|$ by the use of his program. The following table gives an impression of how $|\mathcal{H}_{\chi_{\mathtt{sr}}}[m,n]|$ grows with respect to m and n:

n/m	0	1	2	3	4	5	6	7	8	9
0	1	89	665	2537	7017	15497	30313	54105	88537	138313
1	1	321	2609	8697	25489	59169	107129	197041	334129	495193
2	1	697	5865	20721	60137	131841	247521	455801	769145	1164017
3	1	1217	10353	37737	108817	239657	455177	817017	1375617	2104857
4	1	1881	16121	56265	164537	369569	685529	1245073	2110793	3177961
5	1	2689	23201	82753	240769	540041	1011457	1832281	3092209	4683697
6	1	3641	31481	114105	330793	741441	1398233	2526929	4252105	6466329
7	1	4737	41073	144889	423473	943601	1759529	3213753	5440305	8210905
8	1	5977	51977	185825	541321	1206153	2262801	4123089	6960265	10541905
9	1	7361	64081	231625	672929	1499409	2826617	5139385	8656241	13148617

The fast growth demonstrates the high worst-case time complexity of image matching and also two-dimensional pattern matching.

Moreover, Joe Bandenburg [6] demonstrates the applicability of the new image matching technique in real applications. In his master thesis he uses the image matching algorithm to realize rotation aware motion compensation in MPEG video compression.

The last aspect of ongoing research presented in this section are extended methods of interpolation. Current polynomial time algorithms for image matching or similar problems model image transformations by the help of nearest neighbor interpolation, an out-dated standard notion of image processing. But, although it is possible to approximate image matching for nicer interpolation schemes by the use of nearest neighbor interpolation it is still worthy to look for exact algorithms working with the higher standards. The topic of Florian Wendland's master thesis [46] are some first perspectives of extending the approaches in this thesis to fit the demands of linear interpolation.

6.3 Summary

The main result of the thesis is a structural analysis of the image matching problem with respect to projective transformations and a number of interesting subclasses such as affine and linear transformations, translations, rotations, scalings and combinations of scaling and rotation. Taking advantage of the search space structure common among the covered classes of transformations a generic polynomial time image matching algorithm is introduced.

To provide precise bounds for the running time of the algorithm the complexity of the search space structure is examined for each class of transformations. Polynomial upper bounds are provided for all considered classes but it remains a challenging task to find corresponding lower bounds for affine and projective transformations. Implementations of the proposed algorithm show the potential of the new method but it still needs a lot of effort to make it practically relevant.

Afterwards, the structural complexity of projective image matching is analyzed. The existence of a first order sentence using the majority quantifier, that expresses this problem, is used to argue that projective image matching is contained in U_D-TC_O. Moreover, the problem is even complete in U_D-TC_O which follows from an U_D-FAC_O-reduction from majority.

Altogether, the author presents first step towards general image matching in real applications and hopes that it helps to initiate future work on the practical aspects of image matching.

Bibliography

[1] Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, 1974.

[2] Amihood Amir, Ayelet Butman, Maxime Crochemore, Gad M. Landau, and Mary Schaps. Two dimensional pattern matching with rotations. *Theoretical Computer Science*, 314(1–2):173–187, 2003.

[3] Amihood Amir, Ayelet Butman, Moshe Lewenstein, and Ely Porat. Real two dimensional scaled matching. *Algorithmica*, 53(3):314–336, 2009.

[4] Amihood Amir and Eran Chencinski. Faster two dimensional scaled matching. *Algorithmica*, 56(2):214–234, 2008.

[5] Amihood Amir, Oren Kapah, and Dekel Tsur. Faster two dimensional pattern matching with rotations. *Theoretical Computer Science*, 368(3):196–204, 2006.

[6] Joe Bandenburg. An empirical evaluation of fast rotational block matching algorithms for motion compensation. Master thesis, University of Bristol, 2010.

[7] Alan C. Bovik. *Handbook of Image and Video Processing.* Academic Press, 2000.

[8] Andreas Brandstädt, Van Bang Le, and Jeremy P. Spinrad. *Graph Classes: A Survey.* CRC Press, 1999.

[9] Lisa G. Brown. A survey of image registration techniques. *ACM Computing Surveys*, 24(4):325–376, 1992.

[10] Ashok K. Chandra, Larry Stockmeyer, and Uzi Vishkin. Constant depth reducibility. *SIAM Journal on Computing*, 13(2):423–439, 1984.

[11] Richard Courant and Fritz John. *Introduction to Calculus and Analysis I.* Springer-Verlag, 1999.

[12] Richard Courant and Fritz John. *Introduction to Calculus and Analysis II/1.* Springer-Verlag, 1999.

[13] Richard Courant and Fritz John. *Introduction to Calculus and Analysis II/2.* Springer-Verlag, 1999.

[14] Ingemar Cox, Matthew Miller, and Jeffrey Bloom. *Digital Watermarking.* Morgan Kaufmann, 2001.

[15] Herbert Edelsbrunner. *Algorithms in Combinatorial Geometry.* Springer-Verlag, 1987.

[16] Herbert Edelsbrunner, Joseph O'Rourke, and Raimund Seidel. Constructing arrangements of lines and hyperplanes with applications. *SIAM Journal on Computing*, 15(2):341–363, 1986.

[17] Kimmo Fredriksson, Gonzalo Navarro, and Esko Ukkonen. Optimal exact and fast approximate two dimensional pattern matching allowing rotations. In *Proceedings of the 13th Annual Symposium on Combinatorial Pattern Matching (CPM'02)*, volume LNCS 2373, pages 235–248. Springer-Verlag, 2002.

[18] Kimmo Fredriksson and Esko Ukkonen. A rotation invariant filter for two dimensional string matching. In *Proceedings of the 9th Annual Symposium on Combinatorial Pattern Matching (CPM'98)*, volume LNCS 1448, pages 118–125. Springer-Verlag, 1998.

[19] Martin Golumbic. *Algorithmic Graph Theory and Perfect Graphs: Second Edition.* Elsevier Ltd, 2004.

[20] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Prentice Hall, 2006.

[21] Christian Hundt. Affine image matching is uniform TC_O-complete. In *Proceedings of the 21th Annual Symposium on Combinatorial Pattern Matching (CPM'10)*, volume LNCS 6129, pages 13–25. Springer-Verlag, 2010.

[22] Christian Hundt and Maciej Liśkiewicz. Combinatorial bounds and algorithmic aspects of image matching under projective transformations. In *Proceedings of the 33rd International Symposium on Mathematical Foundations of Computer Science (MFCS'08)*, volume LNCS 5162, pages 395–406. Springer-Verlag, 2008.

[23] Christian Hundt and Maciej Liśkiewicz. Two dimensional pattern matching with combined scaling and rotation. In *Proceedings of the 19th Annual Symposium on Combinatorial Pattern Matching (CPM'08)*, volume LNCS 5029, pages 5–17. Springer-Verlag, 2008.

[24] Christian Hundt and Maciej Liśkiewicz. New complexity bounds for image matching under rotation and scaling. In *Proceedings of the 20th Annual Symposium on Combinatorial Pattern Matching (CPM'09)*, volume LNCS 5577, pages 127–141. Springer-Verlag, 2009.

[25] Christian Hundt and Maciej Liśkiewicz. New complexity bounds for image matching under rotation and scaling. *Journal of Discrete Algorithms*, 2010. Article in Press.

[26] Christian Hundt, Maciej Liśkiewicz, and Ragnar Nevries. A combinatorial geometric approach to two dimensional robustly pattern matching with scaling and rotation. *Theoretical Computer Science*, 410(51):5317–5333, 2009.

[27] Piotr Indyk. Algorithmic aspects of geometric embeddings. In *42nd Annual IEEE Symposium on Foundations of Computer Science (FOCS'01)*, pages 10–33. IEEE Computer Society Press, 2001.

[28] Piotr Indyk, Rajeev Motwani, and Suresh Venkatasubramanian. Geometric matching under noise: Combinatorial bounds and algorithms. In *Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'01)*, pages 354–360. Society for Industrial and Applied Mathematics, 1999.

[29] Rangachar Kasturi and Ramesh Jain. *Computer Vision: Principles.* IEEE Computer Society, 1991.

[30] Claire Kenyon, Yuval Rabani, and Alistair Sinclair. Low distortion maps between point sets. *SIAM Journal on Computing*, 39(4):1617–1636, 2009.

[31] Daniel Keysers and Walter Unger. Elastic image matching is NP-complete. *Pattern Recognition Letters*, 24(1–3):445–453, 2003.

[32] Walter G. Kropatsch and Horst Bischof. *Digital Image Analysis: Selected Techniques and Applications.* Springer-Verlag, 2001.

[33] Gad M. Landau and Uzi Vishkin. Pattern matching in a digitized image. *Algorithmica*, 12(4–5):375–408, 1994.

[34] J.B.Antoine Maintz and Max A. Viergever. A survey of medical image registration. *Medical Image Analysis*, 2(1):1–36, 1998.

[35] David A. Mix-Barrington, Neil Immerman, and Howard Straubing. On uniformity within NC_1. *Journal of Computer and System Sciences*, 41(3):274–306, 1990.

[36] Jan Modersitzki. *Numerical Methods for Image Registration.* Oxford University Press, 2003.

[37] Jan Modersitzki. *FAIR: Flexible Algorithms for Image Registration (Fundamentals of Algorithms).* Society for Industrial Mathematics, 2009.

[38] Thomas B. Moeslund, Adrian Hilton, and Volker Krüger. A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding*, 104(2):90–126, 2006.

[39] Ragnar Nevries. Entwicklung und Analyse eines beschleunigten Image Matching Algorithmus für natürliche Bilder. Diploma thesis, Universität Rostock, 2008.

[40] Bertrand Nouvel and Éric Rémila. Incremental and transitive discrete rotations. In *Proceedings of the 11th International Workshop on Combinatorial Image Analysis (IWCIA'06)*, volume LNCS 4040, pages 199–213. Springer-Verlag, 2006.

[41] Christos Papadimitriou and Shmuel Safra. The complexity of low-distortion embeddings between point sets. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'05)*, pages 112–118. Society for Industrial and Applied Mathematics, 2005.

[42] Yun Q. Shi and Huifang Sun. *Image and Video Compression for Multimedia Engineering: Fundamentals, Algorithms, and Standards.* CRC Press, 1999.

[43] Kaushal Solanki, Upamanyu Madhow, B.S. Manjunath, Shiv Chandrasekaran, and Ibrahim El-Khalil. A survey of medical image registration. *IEEE Transactions on Information Forensics and Security*, 1(4):464–478, 2006.

[44] Robert E. Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972.

[45] Heribert Vollmer. *Introduction to Circuit Complexity.* Springer-Verlag, 1999.

[46] Florian Wendland. Exact polynomial algorithms for image matching with bilinear interpolation. Master thesis, Universität Rostock, 2011.

List of Symbols

$<[\mathsf{u},\mathsf{v}]$	atomic first order formula; satisfied by (s,a) if $a(\mathsf{u}) < a(\mathsf{v})$	p. 84
$\\|\cdot\\|$	Euclidean norm	p. 9
$[\,\cdot\,]$	rounding vector components to integers	p. 9
Γ_χ	surjective mapping $\mathcal{H}_\chi[m,n] \to \mathcal{D}[\mathcal{F}_\chi, m](A)$ for given image A of size n	p. 55
δ	distortion between images A and B of size n, resp. m, by $\delta(A,B) = \sum_{(i,j)\in z(\max\{m,n\})} \omega(A(i,j), B(i,j))$	p. 13
λ	parameter subspace for rotations	p. 55
Π	decision problem; subset of $\{0,1\}^*$	p. 11
τ_χ	geometry preserving mapping $\mathcal{A}_\chi[m,n] \to \mathcal{H}_\chi[m,n]$	p. 60
ϕ	first order formula; ϕ without free variables is a sentence; modular formula $\phi[\mathsf{u}_1,\dots,\mathsf{u}_n]$ with free variables $\mathsf{u}_1,\dots,\mathsf{u}_n$ used for in- and output	p. 84
φ	face, a convex subspace of $\mathbb{R}^8$ defined by half space intersection; open if defined by $h^0(\cdot)$, $h^+(\cdot)$ and $h^-(\cdot)$ and closed if defined by $H^+(\cdot)$, $H^-(\cdot)$; interior $\downarrow\varphi$ is largest contained open face and $\hat{\varphi}$ is the boundary of φ	p. 23
χ	slice, a linear subspace of $\mathbb{R}^8$ defined by hyperplane intersection; $\chi_{\mathtt{a}}$, χ_ℓ, $\chi_{\mathtt{t}}$, $\chi_{\mathtt{s}}$, $\chi_{\mathtt{sr}}$ describe parameter subspaces for affine, linear transformations, translations, scalings and combinations of scaling and rotation	p. 21
ω	distortion between color values $\mathcal{C} \times \mathcal{C} \to \mathbb{N}$	p. 13
A, B	digital images	p. 12
A_I	nearest neighbor interpolation w. r. t. an image A	p. 12
$\mathcal{A}_\chi[\mathcal{E}]$	open arrangement given by a set $\mathcal{E}$ of linear polynomials; for short $\mathcal{A}_\chi[m,n] = \mathcal{A}_\chi[\mathcal{E}[m,n]]$	p. 24
$\mathcal{B}(\phi)$	set of bounded variables w. r. t. first order formula ϕ	p. 84
$\mathrm{Bit}[\mathsf{u},\mathsf{v}]$	atomic first order formula; satisfied by (s,a) if the $a(\mathsf{v})$th bit in binary representation of $a(\mathsf{u})$ is positive	p. 84

$\mathcal{C}$	colors, a set $\{0, \ldots, C-1\}$ of the first C natural numbers	p. 11
$\det(M)$	determinant of matrix M	p. 9
$\mathcal{D}[\mathcal{F}, m](A)$	dictionary of A, a set $\{f(A)[m] \mid f \in \mathcal{F}\}$ of all transformations of A by $f \in \mathcal{F}$	p. 15
e_1, e_2, $e[i, j]$	invalid polynomials; points P with $e_1(P) = 0$, $e_2(P) = 0$ or $e[i, j](P) = 0$ are invalid	p. 35
enc^u, enc^b	unary and binary encoding functions for image matching instances to binary strings	p. 89
$\mathcal{E}$	a set of linear polynomials; $\mathcal{E}[m, n]$ contains $\ell^{\mathtt{v}}_{iji'}$ and $\ell^{\mathtt{h}}_{ijj'}$ for all $(i, j) \in z(m)$ and $i', j' \in \{-n, \ldots, n+1\}$	p. 35
$E_{in}(v)$, $E_{out}(v)$	neighborhood of v in incidence graphs; $E^e_{in}(v)$, $E^e_{out}(v)$ are the restricted neighborhood of v w. r. t. constant e	p. 27
f	transformation $\mathbb{R}^2 \to \mathbb{R}^2$	p. 13
$f(A)[m]$	image transformation of size m w. r. t. transformation f and image A	p. 13
$\mathcal{F}$	set of transformations; $\mathcal{F}_\chi$ is the transformation set represented by points in χ	p. 15
FO	set of decision problems described by first order formulas; $FO = U_D\text{-}AC_O$	p. 86
$FO[Q]$	set of decision problems described by first order formulas with majority quantifier Q; $FO[Q] = U_D\text{-}TC_O$	p. 86
$\mathcal{F}_{\mathtt{p}}$	class of projective transformations; $\mathcal{F}_{\mathtt{a}}$, $\mathcal{F}_\ell$, $\mathcal{F}_{\mathtt{t}}$, $\mathcal{F}_{\mathtt{s}}$, $\mathcal{F}_{\mathtt{r}}$, $\mathcal{F}_{\mathtt{sr}}$ are the subclasses of affine, linear transformations, translations, scalings, rotations and combinations of scaling and rotation	p. 16
$\mathcal{G}_\chi[\mathcal{E}]$	incidence graph of open arrangement $\mathcal{A}_\chi[\mathcal{E}]$, for short $\mathcal{G}_\chi[m, n] = \mathcal{G}_\chi[\mathcal{E}[m, n]]$	p. 26
$h^0(\ell)$	surface in $\mathbb{R}^8$ defined by polynomial ℓ; subspace of all P with $\ell(P) = 0$	p. 21
$h^+(\ell)$,$h^-(\ell)$	half spaces in $\mathbb{R}^8$ defined by polynomial ℓ; subspace of all P with $\ell(P) > 0$, resp. < 0	p. 21
$\mathtt{h}^+_{j'}$, $\mathtt{h}^-_{j'}$	subspaces of $\mathbb{R}^2$ that consist of all pixels $pix(i, j) \in Pix[n]$ with $j \geq j'$, resp. $j < j'$	p. 34
$H^+(\ell)$, $H^-(\ell)$	half spaces in $\mathbb{R}^8$ defined by polynomial ℓ; $H^+(\ell) = h^0(\ell) \cup h^+(\ell)$, $H^-(\ell) = h^-(\ell)$	p. 21
$\mathcal{H}_\chi[\mathcal{E}]$	closed arrangement given by a set $\mathcal{E}$ of linear polynomials; for short $\mathcal{H}_\chi[m, n] = \mathcal{H}_\chi[\mathcal{E}[m, n]]$ and $\mathcal{H}_{\mathcal{Q}_{m,n}}[m, n] = \{\varphi \cap \mathcal{Q}_{m,n} \mid \varphi \in \mathcal{H}[m, n]\}$	p. 24

IM_ω	image matching function w. r. t. a distortion ω; IM^u_ω for unary encoding, IM^b_ω for binary encoding	p. 90
ℓ	polynomial; mostly linear like $\ell^{\mathrm{v}}_{iji'}$, $\ell^{\mathrm{h}}_{ijj'}$ in Definition 3.3	p. 10
$\ell(v)$	subset of linear polynomials in $\mathcal{E}[m,n]$, labels for every node v in the incidence graph	p. 27
$LOGSPACE$	set of problems decidable in logarithmic space	p. 81
$\mathcal{L}[m,n]$	lattice of points in $\mathbb{R}^8$; for $L = 3 \cdot 10^{16} m^{13} n^6$ and $d = (1.5 \cdot 10^9 m^7 n^3 + 0.5)^{-1}$ there are $2L+1$ points in every dimension with a mutual distance of d	p. 42
M	(square) matrix over $\mathbb{R}$	p. 9
$\mathcal{M}$	random access machine	p. 10
MAJ	subset of $\{0,1\}^*$ that consists of all strings s with at least $\lceil 0.5|s| \rceil$ positive bits	p. 83
N	circuit; N_n is a circuit with n input gates	p. 82
$\mathcal{N}$	family $\{N_0, N_1, N_2, \ldots\}$ of circuits, one for every input length	p. 82
$\mathbb{N}$	set $\{0,1,2,\ldots\}$ of nonnegative integers	p. 9
NC_1	set of problems decidable in logarithmic time by a polynomial size family of circuits of gates with fan-in two	p. 81
$pix(i,j)$	pixel at (i,j), a unit square area in $\mathbb{R}^2$ with center (i,j)	p. 12
P	vector in parameter space $\mathbb{R}^8$	p. 9
$P(v)$, $P'(v)$	points of $\mathbb{R}^8$, labels for every node v in the incidence graph; $P'(v)$ is a valid point in $\varphi(v)$	p. 27
$\mathcal{P}[\mathcal{F}]$	surjective mapping $\mathbb{R}^k \to \mathcal{F}$	p. 15
$Pix[n]$	set of pixels $pix(i,j)$ for all $(i,j) \in z(n)$	p. 12
$PTIME$	set of problems decidable in polynomial time	p. 81
Q	majority quantifier; $Q^{\mathrm{t}}_{\mathrm{n}}$ is the related counting quantifier	p. 84
$\mathcal{Q}_r$	hypercube in $\mathbb{R}^8$ with center $(0,\ldots,0)^T$ and diameter r; $\mathcal{Q}_{m,n} = \mathcal{Q}_{1.5 \cdot 10^7 m^6 n^3}$	p. 23
$\mathbb{R}$	set of real numbers; $\mathbb{R}^k$ forms sets of real vectors like the plane $\mathbb{R}^2$ and the projective parameter space $\mathbb{R}^8$	p. 9
s	string over $\{0,1\}^*$; s has length $|s|$, $s(i)$ gives ith bit and $s|s'$ is concatenation of two strings s and s'	p. 10
(s,a)	structure for formula ϕ; s is a string and a an assignment $a : \mathcal{V} \to \{0,1,\ldots,|s|-1\}$; $(s,a) \models \phi$ if it satisfies ϕ	p. 84
T	worst case time for computation of a valid point	p. 61
$\mathcal{T}$	threshold for image matching	p. 20

U_D-AC_O	set of problems decidable in constant time by a polynomial size DLOGTIME uniform family of unbounded fan-in circuits	p. 83
U_D-FAC_O	set of functions computable in constant time by a polynomial size DLOGTIME uniform family of unbounded fan-in circuits	p. 83
U_D-TC_O	set of problems decidable in constant time by a polynomial size DLOGTIME uniform family of unbounded fan-in circuits using threshold gates	p. 83
U_D-FTC_O	set of functions computable in constant time by a polynomial size DLOGTIME uniform family of unbounded fan-in circuits using threshold gates	p. 83
$\mathit{Update}(uv)$	set of tuples (i, j, i', j'), labels for edges $uv \in E^e$ in the incidence graph; traversing from u to v involves an update of pixel (i, j) with color $A(i', j')$	p. 63
$\mathsf{v}_{i'}^{+}$, $\mathsf{v}_{i'}^{-}$	subspaces of $\mathbb{R}^2$ that consist of all pixels $pix(i, j) \in Pix[n]$ with $i \geq i'$, resp. $i < i'$	p. 34
$\mathcal{V}$	set of variables in first order formulas	p. 84
$\mathcal{V}(\phi)$	set of free variables w. r. t. first order formula ϕ	p. 84
$X[\mathsf{v}]$	atomic first order formula; satisfied by (s, a) if $s(a(\mathsf{v})) = 1$	p. 84
$z(n)$	support with size n; images A of size n have $A(i, j) = 0$ for all $(i, j) \notin z(n)$	p. 12
$\mathbb{Z}$	set of integers; $\mathbb{Z}^2$ forms the set of integer pairs	p. 9